HPSS Installation Guide

High Performance Storage System, version 10.3.0.0.0, 29 September 2023

HPSS Installation Guide

High Performance Storage System, version 10.3.0.0.0, 29 September 2023

Table of Contents

List of Figures

List of Tables

About this book. The HPSS Installation Guide is for use both at system installation time as well as throughout the lifetime of the system. It will guide system administrators through the planning

and installation of a new HPSS system. It also guides system administrators through the conversion process to upgrade existing HPSS systems to Release 7.5. It serves as a reference whenever the system is reconfigured by the addition, deletion, or modification of hosts, tape libraries, devices, or other components.

Chapter 1 discusses HPSS changes for the latest release.

Chapter 2 gives an overview of HPSS technology.

Chapters 3-5 guide administrators of new HPSS systems through planning, system preparation, HPSS software installation, and configuration of the HPSS infrastructure.

Conventions used in this book. Example commands that should be typed at a command line will be proceeded by a percent sign ("%") and be presented in a courier font:

```
% sample command
```

Example command output and example contents of ASCII files will be presented in a courier font:

```
sample file line 1

sample file line 2
```

Any text preceded by a pound sign ("#") should be considered comment lines:

```
# This is a comment
```

Chapter 1. Release 10

This chapter summarizes HPSS changes for Release 10 into four categories: new features, retired features, deprecated features, and changed features. Changes since release 9.3 are described. For changes prior to 10.1, consult prior release notes.

1.1. New features in 10.3

New features for 10.3 are described below.

1.1.1. TS1170 / 3592-70F/S / Jag7 support

Support added for TS1170 / 3592-70F/S / Jag7 media/drive type.

1.1.2. dumpv_pvl can display the PVR name

dumppv_pvl can display the PVR name (or N/A) for each volume by specifying the -r option. See the man page for more information.

1.1.3. Migration and Purge State Changes

Migrations and purges will no longer always immediately try to start when the Migration/Purge Server (MPS) starts up.

When the MPS starts up, it will check for an environment variable called **HPSS_MPS_MIGR_PAUSE_SECS**. If it exists, it will use that value as the number of seconds for each storage class migration/purge run to wait before checking to see if it needs to run. If the environment variable is not set, it will use a default value of 300 seconds (5 minutes). Valid values for **HPSS_MPS_MIGR_PAUSE_SECS** are 0 to 2147483647. If the administrator wants to change the state of the active storage class migration/purge before the pause is expired, they may do so. For example, they may want to immediately start a migration.

Additionally the MPS will preserve the state of the storage class migration/purge between restarts of the MPS. For example, if *Storage Class 1* migration policy was `Suspended` and then the MPS shutdown, then *Storage Class 1* migration will remain `Suspended` when the MPS starts back up. There is one exception: if a storage class' migration or purge is `Running` when the MPS crashes, then the state will be changed to `Waiting` to wait for the next **Runtime Interval** before starting.

1.1.4. HPSS S3 interface

HPSS now offers an S3 (Simple Storage Service) interface that provides functionality similar to the AWS S3 interface. See the *Chapter 7, HPSS S3 interface* of the HPSS Install Guide for more information.

1.1.5. Read Queue APIs

A new set of APIs has been added to HPSS to manage the recall of large numbers of files. These APIs are collectively called "Read Queue APIs". See the *HPSS Programmers Reference* for details.

The read queue APIs enable HPSS client applications to send a large list of files to be read to HPSS in a "read queue". The read queue can be added to over time. The client application will request the core server to tell it when a file on its list is ready to run. This enables clients to simply manage very large lists of read requests.

The **lori** tool implements the read queue APIs described above. See the **lori** man page for details.

1.1.6. Persisted Read Queues

The read queue APIs are now persisted in HPSS. When a new read queue comes in, it is now persisted in Db2. When entries are added, those entries are persisted in Db2. This allows the core server to recover from downtime and continue processing read queue requests.

The **lori** tool now supports recovering a read queue in the case that the **lori** tool crashes using the read queue context ID. See the **lori** man page for details.

Additionally, there is a new tool to manage persisted read queues. This tool, **rq_cleanup**, will remove either a specific persisted read queue or all read queues. This can be useful in a case where a site goes down for an upgrade and does not want to recover read queue items upon startup. See the man page for **rq_cleanup** for details.

1.1.7. Updates to avoid server restarts

Updates have been made throughout HPSS to avoid most server restarts after making configuration changes. There is now an *Apply Config* button on the *HPSS Health and Status* window that becomes enabled when configuration changes have been made. After making configuration changes such as adding new migration policies, new purge policies, new storage classes, new hierarchies, and new classes of service, press the *Apply Config* button to apply the changes. Previously, the MPS and Core Servers would need to be restarted to apply these changes, but now this is no longer necessary. See the *HPSS Management Guide* for more information.

1.1.8. Added purge filters based on file size and age (CR 558)

HPSS now provides a capability to filter purge candidates based on the file size and the age of the file. Files are not subject to purging when two new conditions are met: file size is less than or equal to the specified size in bytes and file last access time is greater than or equal to the given time in minutes. The site administrator will determine the cutoff size and time in the purge policy.

When file size and last access time values are set to zero, the purge exemption is disabled, and the behavior is as it was in previous versions of HPSS.

1.2. New features in 10.2

New features for 10.2 are described below.

1.2.1. Purge on migrate (CR 231)

HPSS now provides a capability when creating files on disk to purge them as part of a successful migration. The new client function hpss_PurgeOnMigrate may be called with a flag of PURGE_ON_MIGRATE_SET to enable this functionality or called with a flag of PURGE_ON_MIGRATE_CLEAR to disable the functionality. By default, the functionality is disabled. When the PURGE_ON_MIGRATE_SET flag is set, a file that is at level 0 disk that has an associated purge policy will be purged following a successful migration.

1.2.2. Reinit SCSI PVR control paths (CR 609)

The SCSI PVR now supports rebuilding its control path list during reinitialization. This will drop any broken control paths from its list and add any newly available control paths. This will not impact operations for control paths which currently in use.

1.3. New features in 10.1

New features for 10.1 are described below.

1.3.1. HPSS visualization and monitoring (CR 410)

The approach that HPSS has taken to monitoring is to facilitate integration by providing a suite of tools which can be used by administrators to gather information about HPSS into their preferred workflow. A number of additional tools and logging information have been added, which are documented here, to present raw statistical information captured by HPSS.

A site which does not have existing visualization infrastructure for HPSS can use our provided Kibana dashboards to quickly obtain useful HPSS monitoring. However, a site which does have existing visualization infrastructure might benefit from additional sources of data and integrate it into their monitoring.

Beginning in 10.1, HPSS is delivering templates for use with Kibana to provide more trend views. This includes server status, core server statistics, file counts and bytes by class of service or storage class, tape mounts, transfers by device and mover, and trashcan, migration, and purge statistics over time.

See the *Chapter 6, Installation and configuration of the Elastic (ELK) Stack* section of the HPSS Install Guide for more information.

1.3.2. Restricted access (CR 625)

HPSS now provides finer-grained access controls than the "Restricted User" feature. The "Restricted User" feature was an on/off switch that granted or disabled access to HPSS APIs for particular users. Restricted Access replaces Restricted User as the method for controlling HPSS client access.

Restricted Access allows for a site to restrict access to specific operations by user. Restricted Access duplicates the behavior of the Restricted User feature, but also provides finer-grained control by operation such as restricting copies, creates, writes, and stages.

See the *HPSS Management Guide* for more information.

1.3.3. Files now show the type of media in extended attributes

Extended attribute calls (e.g. hpss_FileGetXAttributes) which retrieve level information now include the type of the media, for each tape reported. The media type can be converted to a more useful string with hpss_MediaTypeString(). See the *HPSS Programmers Reference* for more details.

1.3.4. Files now show current activity in extended attributes

Extended attribute calls (e.g. hpss_FileGetXAttributes) now report back current activity against the file. This includes reads, writes, migrates, purges, changecos, and stages. A value of -1 indicates that the file is not open. See the *HPSS Programmers Reference* for more details.

1.3.5. Low overhead read interface (lori - CR 562)

lori is a new tool which behaves similarly to quaid and shares many of the same options including organizing requests by tape, callouts upon read completion, filtering, and more. However, while quaid stages files up to the top level of the hierarchy, lori uses PIO to read data out of HPSS with minimal overhead and into the selected output directory. See the lori man page for more details.

In the future, lori will take advantage of upcoming changes to enable mass recall with read recovery.

1.3.6. dump_acct_sum will now dump the bandwidth table

dump_acct_sum will now present the bandwidth table (bytes read and written) by account/UID/GID/COS using the -b option. See the man page for more information.

1.3.7. dumpv_pvl now displays the HPSS label format

dumppv_pvl now outputs the tape label format for each volume. See the man page for more information.

1.3.8. lscos and lsvol now support JSON output

lscos and lsvol now support outputting in JSON format with -j. See the tools' respective man pages for details.

1.3.9. Toggle RAO on/off per tape device

RAO can now be toggled on/off for each tape device. The default is ON. See the *HPSS Management Guide* for more information.

1.3.10. Repack will log media access INFO logs

Repack now logs information about source and destination tapes involved in repack operations as INFO logs. Individual file access is not logged for repack.

1.3.11. HPSS server metrics tool

A new tool has been created, hpss_server_metrics. This tool provides system statistics which previously could only be gathered through a variety of other tools and APIs.

The server metrics tool provides statistics in both human-readable and JSON format, and can also reset statistics for certain reports to generate interval-based statistics.

See the hpss_server_metrics man page for more information.

1.3.12. HPSS DB metrics tool

A new tool has been created, hpss_db_metrics. This tool provides database statistics which previously could only be gathered through out of band database queries.

The db metrics tool provides statistics in both human-readable and JSON format.

See the hpss_db_metrics man page for more information.

1.4. Features retired in 10.1

HPSS features retired in 10.1 are described below.

1.4.1. Restricted user list

The Restricted Access feature replaces the Restricted User List feature. The Restricted User screens in the SSM have been removed.

The Restricted Access feature can replicate the behavior of the Restricted User List by denying access to the same set of users for "ALL" operations. See the Restricted Access documentation for more details.

1.5. Features deprecated in 10.1

There were no HPSS features deprecated in 10.1.

1.6. Changes to existing HPSS features in 10.1

HPSS features significantly altered in 10.1 are described below.

1.6.1. HPSS now supports 65535 alternate groups

HPSS now supports users belonging to 65535 alternate groups, up from 64.

1.6.2. hpssmsg now supports more fields

hpssmsg is a tool to log messages in the HPSS log message format. Previously, this tool could only log a message with a type and severity. Now, hpssmsg also allows return code, functions, request id, program name, and a message ID to be logged. See the hpssmsg man page for more information.

1.6.3. Migration/Purge server raw report files now using JSON Lines.

The Migration/Purge Server writes daily reports. The reports have been changed to newline JSON formatted records. The new format is using JSON Lines (https://jsonlines.org/) for each record. This new format allows other tools like ELK and Splunk to take advantage of JSON output.

The mps_reporter tool has been updated to work with the new format. However, to view older MPS reports, you must use the mps_reporter_legacy tool.

When upgrading to HPSS 10.1, archive all files in the MPS report path before starting the 10.1 Migration/Purge Server. The old format of the MPS Report files are not compatible with the new JSON Lines format.

1.6.4. JSON timestamps now in ISO 8601 format

Timestamps in HPSS output to JSON widely used "seconds since epoch" format. However, for ease of use with applications that process timestamp data, the standard ISO 8601 format has been selected.

For example:

Epoch timestamp: 1656091032

Equivalent ISO 8601 formatted date: 2022-06-24T12:17:12Z

This change in the JSON timestamp format does not impact non-JSON outputs for these tools.

The following tools were modified to use the ISO 8601 format:

- dump_acct_sum
- lsvol

- lshpss
- rtmu
- dump_sspvs
- lspvhist

1.6.5. Parallel quaid file scans

Quaid now performs its initial input selection scans in parallel. This allows quaid to organize the inputs and begin staging faster. Users can control the amount of parallelism of the scan using the -n option. See the quaid man page for more information.

1.6.6. Avoid purging newly staged data with Last Access Time purge policies

When purge records were created following a stage request they used the file access statistics to create the purge time. When combined with the last access time purge policy, this would mean that a staged file was immediately purgeable since the file access time had not been updated.

HPSS has been modified to use the internal statistics for the top level when creating the purge record. This timestamp will be the time that the data was staged, meaning that the data will not be eligible to be purged until it has reached the Last Access Time threshold.

1.7. API changes in HPSS 10.1

All API changes are documented in the *HPSS Programmers Reference*.

- The functions hpss_GetRestrictedUserList and hpss_ReadRestrictedUserFile were removed.
- A new function, hpss_GetUID, was added to get the current client thread’s UID.
- A new function, hpss_GetGID, was added to get the current client thread’s GID.
- A new function, hpss_SetApplicationName, allows an application to set its name. This name will be logged at the core server REQUEST logs.
- The sec_cred_t structure has been modified to include a dynamically allocated array of that supports 64K alternate groups. Functions which return sec_cred_t now return heap allocations that need to be freed. Uses of sec_cred_t in HPSS to deal with credentials should be handled with care when porting applications to HPSS 10.1 to deal with memory leaks. The functions hpss_SECFreeCreds and hpss_SECDuplicateCred have been added to facilitate these changes.
- A number of APIs have been added for "read queueing". These APIs are not currently implemented and return ENOTSUP.

Chapter 2. HPSS basics

2.1. Introduction

The High Performance Storage System (HPSS) provides hierarchical storage management and services for very large storage environments. HPSS may be of interest to organizations having present and future scalability requirements that are very demanding in terms of total storage capacity, file sizes, data rates, number of objects stored, and numbers of users. HPSS is part of an open, distributed environment based on remote procedure calls, Kerberos, LDAP directory systems and DB2 which form the infrastructure of HPSS. HPSS is the result of a collaborative effort by leading US Government supercomputer laboratories and industry to address real and urgent high-end storage requirements. HPSS is offered commercially by IBM.

HPSS provides scalable parallel storage systems for highly parallel computers as well as traditional supercomputers and workstation clusters. Concentrating on meeting the high end of storage system and data management requirements, HPSS is scalable and designed to store up to petabytes (10^{15}) of data and to use network-connected storage devices to transfer data at rates up to multiple gigabytes (10^{9}) per second.

HPSS provides a large degree of control for the customer to manage the hierarchical storage system. Using configuration information defined by the customer, HPSS organizes storage devices into multiple storage hierarchies. Based on policy information defined by the customer and actual usage information, data are then moved to the appropriate storage hierarchy and to appropriate levels in the storage hierarchy.

2.2. HPSS capabilities

A primary goal of HPSS is to move large files between storage devices and parallel or clustered computers at speeds many times faster than today's commercial storage system software products and to do this in a way that is more reliable and manageable than is possible with current systems. In order to accomplish this goal, HPSS is designed and implemented based on the concepts described in the following subsections.

2.2.1. Network-centered architecture

The focus of HPSS is the network, not a single processor as in conventional storage systems. HPSS provides servers that can be distributed across a high-performance network to provide scalability and parallelism. The basis for this architecture is the *IEEE Mass Storage System Reference Model, Version 5*.

2.2.2. High data transfer rate

HPSS achieves high data transfer rates by eliminating overhead normally associated with data transfer operations. In general, HPSS servers establish and control transfer sessions but are not involved in the actual transfer of data.

2.2.3. Parallel operation

The HPSS Client Application Program Interface (Client API) supports parallel or sequential access to storage devices by clients executing parallel or sequential applications. HPSS also provides a Parallel File Transfer Protocol (PFTP). HPSS can even manage data transfers in a situation where the number of data sources differs from the number of destination sources. Parallel data transfer is vital in situations that demand fast access to very large files.

2.2.4. Based on standard components

HPSS runs on UNIX and is written in ANSI C and Java. It uses remote procedure calls, a selectable security service (Kerberos or UNIX), UNIX or LDAP for user configuration information, and DB2 as the basis for its portable, distributed, transaction-based architecture. These components are offered on many vendors' platforms.

The full HPSS system has been implemented on the Linux platform, and some components of HPSS are available on other platforms. Refer to *Section 2.4, "HPSS hardware platforms"* and *Section 3.3, "Prerequisite software considerations"* for specific information.

To aid vendors and users in porting HPSS to new platforms, HPSS source code is available.

2.2.5. Data integrity through transaction management

Transactional metadata management, provided by DB2, enables a reliable design that protects user data both from unauthorized use and from corruption or loss. A transaction is an atomic grouping of metadata management functions managed so that either all of them take place together or none of them takes place. Journaling makes it possible to back out any partially complete transactions if a failure occurs. Transaction technology is common in relational data management systems but not in storage systems. It is the key to maintaining reliability and security while scaling upward into a large, distributed storage environment.

2.2.6. Multiple hierarchies and Classes of Services

Most other storage management systems support simple storage hierarchies consisting of one kind of disk and one kind of tape. HPSS provides multiple configurable hierarchies, which are particularly useful when inserting new storage technologies over time. As new disks or tapes are added, new classes of service can be set up. HPSS files reside in a particular class of service which users select based on parameters such as file size and performance. A class of service is implemented by a storage hierarchy which in turn consists of multiple storage classes, as shown in *Figure 2.2, "Class of Service/ hierarchy/storage class"*. Storage classes are used to logically group storage media to provide storage for HPSS files. A hierarchy may be as simple as a single tape, or it may consist of two or more levels of disk and local tape. The user can even set up classes of service so that data from an older type of tape is subsequently migrated to a new type of tape. Such a procedure allows migration to new media over time without having to copy all the old media at once.

2.2.7. Storage subsystems

Storage subsystems (or simply, "subsystems") may be used to increase the scalability of HPSS in handling concurrent requests or to meet local political needs. Each subsystem contains a single Core Server. If migration and purge are needed for the subsystem, then it will also contain a Migration/ Purge Server. All other servers are subsystem-independent.

Data stored within HPSS is assigned to different subsystems based on pathname resolution. A pathname consisting of a forward slash ("/") resolves to the root Core Server which is specified in the global configuration file. However, if the pathname contains junction components, it may resolve to a Core Server in a different subsystem. For example, the pathname `/JunctionToSubsys2/mydir` could lead to a fileset managed by the Core Server in subsystem 2. Sites which do not wish to partition their HPSS through the use of subsystems will run HPSS with a single subsystem.

2.3. HPSS components

The components of HPSS include files, filesets, junctions, virtual volumes, physical volumes, storage segments, metadata, servers, infrastructure, user interfaces, a management interface, and policies. Metadata is control information about the data stored under HPSS, such as location, access times, permissions, and storage policies. Most HPSS metadata is stored in tables in a DB2 relational database. Media and file metadata are represented by data structures that describe the attributes and characteristics of storage system components such as files, filesets, junctions, storage segments, and volumes. Servers are the processes that control the logic of the system and control movement of the data. The HPSS infrastructure provides the services that are used by all the servers for operations such as sending messages and providing reliable transaction management. User interfaces provide several different views of HPSS to applications with different needs. The management interface provides a way to administer and control the storage system and implement site policy.

These HPSS components are discussed below in the following sections.

Figure 2.1. File migration and stage operations

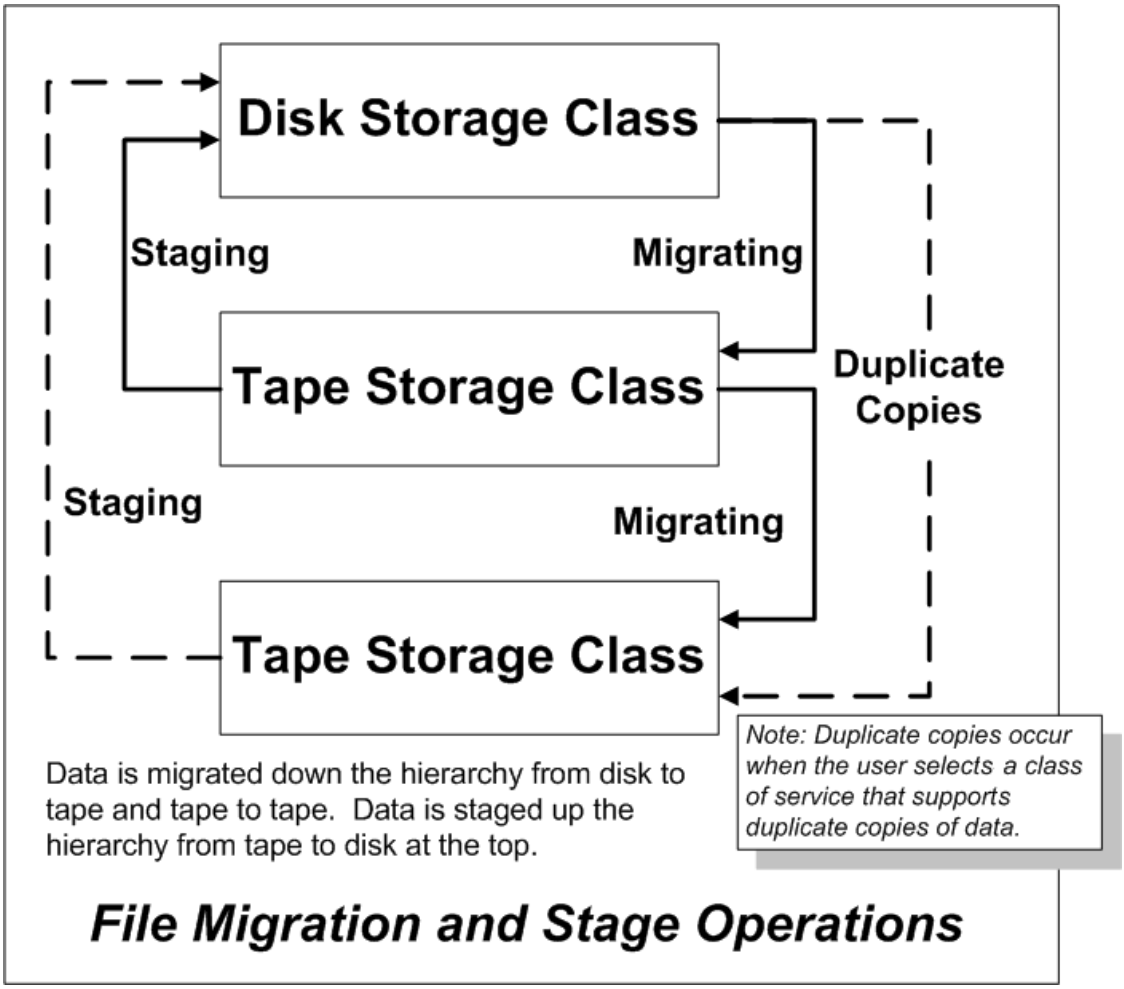
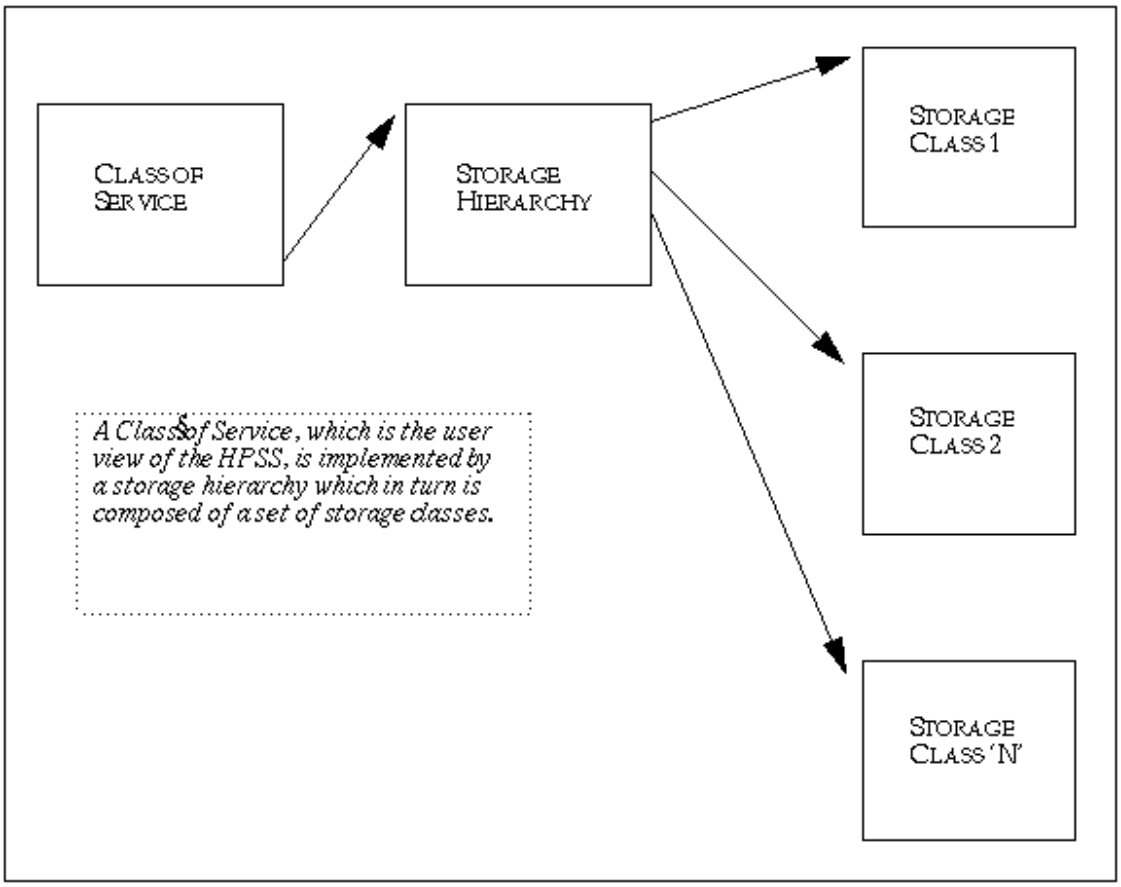


Figure 2.2. Class of Service/hierarchy/storage class



2.3.1. HPSS files, filesets, volumes, storage segments and related metadata

The various metadata constructs used to describe the HPSS name space and HPSS storage are described below:

Files (bitfiles)

Files in HPSS, called bitfiles in deference to IEEE Mass Storage Reference Model terminology, are logical strings of bytes, even though a particular bitfile may have a structure imposed by its owner. This unstructured view decouples HPSS from any particular file management system that host clients of HPSS might use. HPSS bitfile size is limited to 2^{64} - 1 bytes.

Each bitfile is identified by a machine-generated name called a bitfile ID. A bitfile may also have a human-readable name. It is the job of the HPSS Core Server (discussed in *Section 2.3.2, "HPSS servers"*) to map a human-readable name to a bitfile's ID.

Filesets

A fileset is a logical collection of files that can be managed as a single administrative unit, or more simply, a disjoint directory tree. A fileset has two identifiers: a human-readable name and a numeric fileset ID. Both identifiers are unique to a given realm.

Junctions

A junction is a Core Server object, much like a symbolic link to a directory, that is used to point to a fileset. This fileset may belong to the same Core Server or to a different Core Server.

When pointing to a different Core Server, junctions allow HPSS users to traverse to different subsystems.

File Families

HPSS files can be grouped into families. All files in a given family are recorded on a set of tapes assigned to the family. Only files from the given family are recorded on these tapes. HPSS supports grouping files on tape volumes only. A family can be selected at the time a file is created by supplying the appropriate family ID as one of the create parameters. All files created in the fileset belong to the family. When one of these files is migrated from disk to tape, it is recorded on a tape with other files in the same family. If no tape virtual volume is associated with the family, a blank tape is reassigned from the default family. The family affiliation is preserved when tapes are repacked.

Physical Volumes

A physical volume is a unit of storage media on which HPSS stores data. The media can be removable (for example, cartridge tape or optical disk) or non-removable (magnetic disk). Physical volumes may also be composite media, such as RAID disks, but must be represented by the host OS as a single device.

Physical volumes are not visible to the end user. The end user stores bitfiles into a logically unlimited storage space. HPSS, however, must implement this storage on a variety of types and quantities of physical volumes.

For a list of the tape physical volume types supported by HPSS, see *Section 3.4.4, "Tape devices"*.

Virtual Volumes

A virtual volume is used by the Core Server to provide a logical abstraction or mapping of physical volumes. A virtual volume may include one or more physical volumes. Striping of storage media is accomplished by the Core Servers by collecting more than one physical volume into a single virtual volume. A virtual volume is primarily used inside of HPSS, thus hidden from the user, but its existence benefits the user by making the user's data independent of device characteristics. Virtual volumes are organized as strings of bytes up to 2^{64} - 1 bytes in length that can be addressed by an offset into the virtual volume.

Storage Segments

A storage segment is an abstract storage object which is mapped onto a virtual volume. Each storage segment is associated with a storage class (defined below) and has a certain measure of location transparency. The Core Server (discussed in *Section 2.3.2, "HPSS servers"*) uses both disk and tape storage segments as its primary method of obtaining and accessing HPSS storage resources. Mappings of storage segments onto virtual volumes are maintained by the HPSS Core Servers.

Storage Maps

A storage map is a data structure used by the Core Server to manage the allocation of storage space on virtual volumes.

Storage Classes

A storage class defines a set of characteristics and usage parameters to be associated with a particular grouping of HPSS virtual volumes. Each virtual volume and its associated physical volumes belong to a single storage class in HPSS. Storage classes in turn are grouped to form

storage hierarchies (see below). An HPSS storage class is used to logically group storage media to provide storage for HPSS files with specific intended usage, similar size and usage characteristics.

Storage Hierarchies

An HPSS storage hierarchy defines the storage classes on which files in that hierarchy are to be stored. A hierarchy consists of multiple levels of storage, with each level representing a different storage class. Files are moved up and down the hierarchy via migrate and stage operations based on usage patterns, storage availability, and site policies. For example, a storage hierarchy might consist of a fast disk, followed by a fast data transfer and medium storage capacity robot tape system, which in turn is followed by a large data storage capacity but relatively slow data transfer tape robot system. Files are placed on a particular level in the hierarchy depending upon the migration levels that are associated with each level in the hierarchy. Multiple copies are controlled by this mechanism. Also data can be placed at higher levels in the hierarchy by staging operations. The staging and migrating of data is shown in *Figure 2.1, "File migration and stage operations"*.

Class of Service (COS)

Each bitfile has an attribute called Class Of Service. The COS defines a set of parameters associated with operational and performance characteristics of a bitfile. The COS results in the bitfile being stored in a storage hierarchy suitable for its anticipated and actual size and usage characteristics. *Figure 2.2, "Class of Service/hierarchy/storage class"* shows the relationship between COS, storage hierarchies, and storage classes.

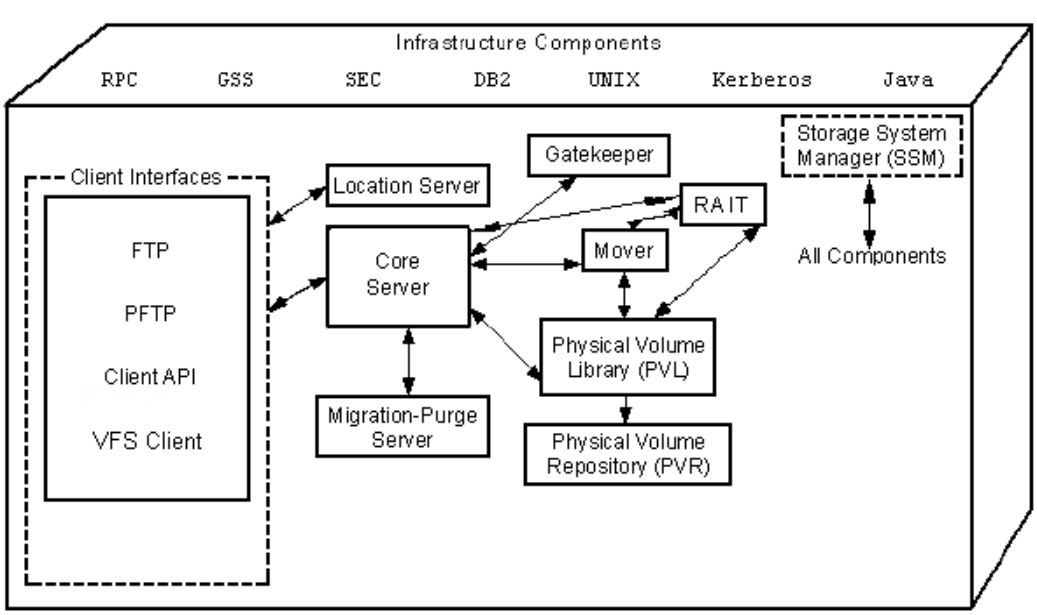
User-defined Attributes (UDAs)

User-defined Attributes in HPSS are client-created attributes containing additional metadata. UDAs can be associated with any name space object. UDAs are represented internally as a well-formed XML document. The XML document size limit is 2 GB. The maximum amount of data that can be returned at any one time is configurable via Core Server and API settings. See the *HPSS Management Guide* for more details.

2.3.2. HPSS servers

HPSS servers include the Core Server, Migration/Purge Server, Gatekeeper, Location Server, Physical Volume Library, Physical Volume Repository, Mover, Storage System Management System Manager, and Startup Daemon. *Figure 2.3, "HPSS components"* provides a simplified view of the HPSS system. Each major server component is shown, along with the basic control communication paths (thin arrowed lines). Infrastructure items (those components that "glue together" the distributed servers) are shown at the top of the cube. These infrastructure items are discussed in *Section 2.3.4, "HPSS infrastructure"*. HPSS user interfaces (the clients listed in the figure) are also discussed in *Section 2.3.5, "HPSS user interfaces"*.

Figure 2.3. HPSS components



Core Server (CS)

The Core Server provides several key sets of functionality.

First, the Core Server provides translation between human-oriented names and HPSS object identifiers. Name space objects managed by the Core Server are filesets, junctions, directories, files, hard links, and symbolic links. The Core Server provides access verification to objects and mechanisms for manipulating access to these objects via a Portable Operating System Interface (POSIX) view of the name space. This name space is a hierarchical structure consisting of directories, files, and links. These name space objects may exist within filesets that are connected via junctions.

Second, the Core Server provides the abstraction of logical bitfiles to its clients. A bitfile is identified by a Core Server generated name called a bitfile ID. Clients may reference portions of a bitfile by specifying the bitfile ID and a starting address and length. The Core Server supports random access to files and sparsely written files. It supports parallel reading and writing of data to bitfiles and performs the mapping of logical portions of bitfiles onto physical storage devices. The Core Server supports the migration, purging, and staging of data in a storage hierarchy (though the migration/purge policies are implemented through the Migration/Purge Server, a client to the Core Server).

Third, the Core Server provides a hierarchy of storage objects: storage segments, virtual volumes, and physical volumes. The Core Server translates storage segment references into virtual volume references and then into physical volume references, handles the mapping of physical resources into striped virtual volumes to allow parallel I/O to that set of resources, and schedules the mounting and dismounting of removable media through the Physical Volume Library (see below).

Migration/Purge Server (MPS)

The MPS allows a site to implement its storage management policies by managing the placement of data on HPSS storage media using site-defined migration and purge policies. By making appropriate calls to its Core Server, an MPS copies data to lower levels in the hierarchy

(migration), removes data from the current level once copies have been made (purge), or moves data between volumes at the same level (lateral move). Based on the hierarchy configuration, MPS can be directed to create duplicate copies of data when it is being migrated from disk or tape. This is done by copying the data to multiple lower levels in the storage hierarchy.

There are two types of migration: disk migration and tape file migration. The designation disk or tape refers to the type of storage class that migration is running against. See *Section 3.7.2, "Migration/Purge Server"* for a more complete discussion of the different types of migration.

MPS runs migration on each storage class periodically using the time interval specified in the migration policy for that class. See *Section 2.3.7, "HPSS policy modules"* for details on migration and purge policies. Migration runs can be started automatically when the warning or critical space thresholds for the storage class are exceeded. In addition, migration runs can be started manually by an administrator.

Purge runs are started automatically on each storage class when the free space in that class falls below the percentage specified in the purge policy. Purge runs may also be started manually.

Disk migration/purge:

The purpose of disk migration is to make one or more copies of disk files to lower levels in the hierarchy. The number of copies depends on the configuration of the hierarchy. For disk, migration and purge are separate operations. It is common for disk storage classes which have been configured for migration to also be configured for purge as well. Once a file has been migrated (copied) downwards in the hierarchy, it becomes eligible for purge, which subsequently removes the file from the higher level and allows the disk space to be reused.

Tape file migration:

The purpose of tape file migration is to make an additional copy (or multiple additional copies) of a file, in a tape storage class, to a lower level in the hierarchy. It is also possible to move files downwards instead of copying them. In this case, there is no duplicate copy maintained. There is no separate purge component to tape file migration. Empty volumes must be reclaimed using the **reclaim** utility.

Gatekeeper (GK)

The Gatekeeper provides two main services:

- It provides sites with the ability to schedule the use of HPSS resources using the Gatekeeping Service.
- It provides sites with the ability to validate user accounts using the Account Validation Service.

Both of these services allow sites to implement their own policy.

The default Gatekeeping Service policy is to not do any gatekeeping. Sites may choose to implement a policy for monitoring authorized callers, creates, opens, and stages. The Core Server will call the appropriate GK API depending on the requests that the site-implemented policy is monitoring.

The Account Validation Service performs authorizations of user storage charges. A site may perform no authorization, default authorization, or site-customized authorization depending on how the accounting policy is set up and whether or not a site has written site-specific account

validation code. Clients call this service when creating files, changing file ownership, or changing accounting information. If account validation is enabled, the Account Validation Service determines if the user is allowed to use a specific account or gives the user an account to use, if needed. The Core Server also calls this service to perform an authorization check just before account-sensitive operations take place.

Location Server (LS)
The Location Server acts as an information clearinghouse to its clients through the HPSS Client API to enable them to locate servers and gather information from both local and remote HPSS systems. Its primary function is to allow a client to determine a server's location and, by knowing other information about the server such as its object UUID, determine its server type or its subsystem id. This allows a client to contact the appropriate server. Usually the Location Server is only used by the Core Server or the Gatekeeper.

Physical Volume Library (PVL)
The PVL manages all HPSS physical volumes. It is in charge of mounting and dismounting sets of physical volumes, allocating drive and cartridge resources to satisfy mount and dismount requests, providing a mapping of physical volume to cartridge and of cartridge to Physical Volume Repository (PVR), and issuing commands to PVRs to perform physical mount and dismount actions. A primary function of the PVL is the support for atomic mounts of sets of cartridges for parallel access to data. Atomic mounts are implemented by the PVL, which waits until all necessary cartridge resources for a request are available before issuing mount commands to the PVRs.

Physical Volume Repository (PVR)
PVRs manage HPSS cartridges. Though an HPSS system may contain multiple PVRs, each cartridge is managed by only one. PVRs provide APIs for clients to request cartridge mounts and dismounts and query the status of cartridges. For convenience, PVRs are often configured in one-to-one correspondence to tape libraries.

For information on the types of tape libraries supported by HPSS PVRs, see *Section 3.4.2, "Robotically mounted tape"*.

An Operator PVR is provided for cartridges not under control of a robotic library. These cartridges are mounted on a set of drives by operators.

Mover (MVR)
The purpose of the Mover is to transfer data from a source device to a sink device. A device can be a standard I/O device with geometry (such as tape or disk) or a device without geometry (such as network or memory). The Mover's client (typically the Core Server) describes the data to be moved and where the data is to be sent. It is the Mover's responsibility to actually transfer the data, retrying failed requests and attempting to optimize transfers. The Mover supports transfers for disk devices, tape devices and a Mover protocol that can be used as a lightweight coordination and flow control mechanism for large transfers.

Storage System Management System Manager (SSMSM)
SSM, the Storage System Management subsystem, is the tool used by the system administrator to manage HPSS. SSM has three components, one of which is an HPSS server and two of which are user interface client programs. The server is:

SSM System Manager (SSMSM, or hpss_ssmsm)
Communicates with all other HPSS components requiring monitoring or control.

The user interface clients are:

SSM GUI (hpssgui)
Provides the HPSS administrator or operator the ability to configure or monitor the HPSS System through a graphical user interface.

SSM Command-Line Interface (hpssadm)
Provides the HPSS administrator or operator the ability to configure or monitor a subset of the HPSS system through a set of interactive or batch commands.

SSM enables the administrator to configure, monitor and control the resources (servers, devices, tape libraries, and media) of HPSS in ways that conform to the management policies of a given customer site.

Monitoring capabilities include the ability to query the values of important management attributes of storage system resources and the ability to receive notifications of alarms and other significant system events. Controlling capabilities include the ability to start up and shut down servers and the ability to set the values of management attributes of storage system resources and storage system policy parameters. Additionally, SSM can request that specific operations be performed on resources within the storage system, such as adding and deleting logical or physical resources.

Startup Daemon (SUD)
The Startup Daemon is called by the SSMSM to start each HPSS server except the SSMSM, the Startup Daemon itself, and the remote portion of the Mover. A Startup Daemon is required on each node where any HPSS Server executes, with the exception that no Startup Daemon is required on nodes where the only HPSS Server is the remote portion of the Mover.

2.3.3. HPSS storage subsystems

The goal of storage subsystems (or just "subsystems") is to increase the scalability of HPSS by allowing multiple Core Servers to be used within a single HPSS system. Every HPSS system is partitioned into one or more subsystems. Each subsystem contains a single Core Server. If migration and purge are needed, then the subsystem should contain a single Migration/Purge Server. Each Core Server and each Migration/Purge Server must exist within a storage subsystem. Each subsystem may optionally be serviced by a Gatekeeper which performs site-specific user-level scheduling of HPSS storage requests or account validation. Each Gatekeeper may service multiple subsystems. All other servers exist independently of storage subsystems. Sites which do not need multiple Core Servers use a single storage subsystem.

The computer that runs the Core Server for subsystem X is referred to as the "Subsystem X node" while the computer running the Root Core Server is known as the "Root Subsystem Node".

Each HPSS system consists of two types of DB2 databases. The global database contains subsystem-independent data, and a subsystem database contains subsystem-dependent data. An HPSS system has exactly one global database and one or more subsystem databases.

The definitions of classes of service, hierarchies, and storage classes apply to the entire HPSS system (they are subsystem-independent). All classes of service, hierarchies, and storage classes are known to all storage subsystems within HPSS. The level of resources dedicated to these entities by each storage subsystem may differ. It is possible to disable selected classes of service within given storage

subsystems. Although the class of service definitions are global, if a class of service is disabled within a storage subsystem then the Core Server in that subsystem never selects that class of service.

Since the Migration/Purge Server (MPS) is contained within the storage subsystem, migration and purge operate independently in each subsystem. Each Migration/Purge Server is responsible for migration and purge for those storage class resources contained within its particular storage subsystem. Migration and purge runs are independent and are not synchronized. Migration and purge for a storage class may be configured differently for each storage subsystem. It is possible to set up a single migration or purge policy which applies to a storage class across all storage subsystems (to make configuration easier), but it is also possible to control migration and purge differently in each storage subsystem.

Similarly, storage class thresholds may be configured differently for each storage subsystem. It is possible to set up a single set of thresholds which apply to a storage class across all storage subsystems, but it is also possible to control the thresholds differently for each storage subsystem.

2.3.4. HPSS infrastructure

The HPSS infrastructure items (see *Figure 2.3, "HPSS components"*) are those components and services used by the various HPSS servers. The HPSS infrastructure components common among servers are discussed below.

Remote Procedure Calls (RPC)

Most HPSS servers, with the exception of the MVR, PFTPD, and logging services (see below), communicate requests and status (control information) via RPCs. HPSS does not use RPCs to move user data. RPCs provide a communication interface resembling simple, local procedure calls.

Thread Services

HPSS uses a threads package for multitasking. The threads package is vital for HPSS to serve large numbers of concurrent users and to enable multiprocessing of its servers.

Transaction Management

Requests to perform actions, such as creating bitfiles or accessing file data, result in client-server interactions between software components. The HPSS Core Server performs most of the HPSS metadata changes using the transaction management tools provided by DB2. For the most part, these metadata transactions are managed entirely within the Core Server. Other servers such as MPS and PVL modify their metadata transactionally, and those transactions are entirely contained within those servers. A very small number of rarely performed operations require distributed transaction management, and these are handled by DB2 as well.

Transactional integrity to guarantee consistency of server state and metadata is required in HPSS in case a particular component fails. HPSS metadata updates utilize the transactional capability of DB2. The selection of DB2 was based on functionality and vendor platform support. It provides HPSS with an environment in which a job or action completes successfully or is aborted completely.

DB2 provides a full suite of recovery options for metadata transactions. Recovery of the database to a consistent state after a failure of HPSS or DB2 is automatic. A full suite of database backup and maintenance tools is provided as well.

Security

HPSS security software provides mechanisms that allow HPSS components to communicate in an authenticated manner, to authorize access to HPSS objects, to enforce access control on HPSS objects, and to issue log records for security-related events. The security components of HPSS provide authentication, authorization, enforcement, and audit capabilities for the HPSS components. Customer sites may use the default security policy delivered with HPSS or define their own security policy by implementing their own version of the security policy module.

- **Authentication** is responsible for guaranteeing that a principal (a customer identity) is the entity that is claimed, and that information received from an entity is from that entity.
- **Authorization** is responsible for enabling an authenticated entity access to an allowed set of resources and objects. Authorization enables end-user access to HPSS directories and bitfiles.
- **Enforcement** is responsible for guaranteeing that operations are restricted to the authorized set of operations.
- **Audit** is responsible for generating a log of security-relevant activity. HPSS audit capabilities allow sites to monitor HPSS authentication, authorization, and file security events. File security events include file creation, deletion, opening for I/O, and attribute modification operations.

HPSS components that communicate with each other maintain a joint security context. The security context for both sides of the communication contains identity and authorization information for the peer principals as well as an optional encryption key.

Access to HPSS server interfaces is controlled through an Access Control List (ACL) mechanism. Membership on this ACL is controlled by the HPSS administrator.

Logging

A logging infrastructure component in HPSS provides an audit trail of server events. Logged data may include alarms, events, requests, security audit records, info records, trace information, debug records, and accounting records. HPSS logs are logged to syslog, and Alarms and Event messages may be reflected to the SSM for display there. Log retention and archiving are accomplished using standard UNIX tools such as **logrotate**, along with HPSS tools like **hpss_log_archive**. See the **Logging** section of the *HPSS Management Guide* for more information.

Accounting

The HPSS accounting system provides the means to collect usage information in order to allow a particular site to charge its users for the use of HPSS resources. It is the responsibility of the individual site to sort and use this information for subsequent billing based on site-specific charging policies. For more information on the HPSS accounting policy, refer to *Section 2.3.7, "HPSS policy modules"*.

2.3.5. HPSS user interfaces

As indicated in *Figure 2.3, "HPSS components"*, HPSS provides the user with a number of transfer interfaces as discussed below.

File Transfer Protocol (FTP)

HPSS provides an industry-standard FTP user interface. Because standard FTP is a serial interface, data sent to a user is received serially. This does not mean that the data within HPSS is

not stored and retrieved in parallel; it means that the PFTP daemon within HPSS must consolidate its internal parallel transfers into a serial data transfer to the user. HPSS FTP performance in many cases will be limited not by the speed of a single storage device but by the speed of the data path between the HPSS PFTP daemon and the user's FTP client.

Parallel FTP (PFTP)
The PFTP supports standard FTP commands plus extensions and is built to optimize performance for storing and retrieving files from HPSS by allowing data to be transferred in parallel across the network media. The parallel client interfaces have a syntax similar to FTP but with numerous extensions to allow the user to transfer data to and from HPSS across parallel communication interfaces established between the PFTP client and the HPSS Movers. This provides the potential for using multiple client nodes as well as multiple server nodes. PFTP supports transfers via TCP/IP. The PFTP client establishes a **control** connection with the HPSS PFTP daemon and subsequently establishes TCP/IP **data** connections directly with HPSS Movers to transfer data at rates limited only by the underlying media, communications hardware, and software.

Client Application Program Interface (Client API)
The Client API is an HPSS-specific programming interface that mirrors the POSIX.1 specification where possible to provide ease of use to POSIX application programmers. Additional APIs are also provided to allow the programmer to take advantage of the specific features provided by HPSS (for example, storage/access hints passed on file creation and parallel data transfers). The Client API is a programming-level interface. It supports file open/create and close operations; file data and attribute access operations; file name operations; directory creation, deletion, and access operations; and working directory operations. HPSS users interested in taking advantage of parallel I/O capabilities in HPSS can add Client API calls to their applications to utilize parallel I/O. For the specific details of this interface see the *HPSS Programmer's Reference*.

HPSSFS-FUSE Interface
The HPSSFS-FUSE Interface presents a standard POSIX I/O interface to a user application. This obviates the need for a user application to be rewritten against the HPSS Client API and hence can be used "out of the box" as long as the user application is POSIX-compliant. A portion of an HPSS directory tree can be mounted on a client machine as if it were a local POSIX-compliant file system. See the *HPSSFS-FUSE Administrator's Guide* bundled with HPSSFS-FUSE for more information.

2.3.6. HPSS management interfaces

HPSS provides a graphical user interface, the SSM **hpssgui**, for HPSS administration and operations GUI. The **hpssgui** simplifies the management of HPSS by organizing a broad range of technical data into a series of easy-to-read graphic displays. The **hpssgui** allows monitoring and control of virtually all HPSS processes and resources from windows that can easily be added, deleted, moved, or overlapped as desired.

HPSS also provides a command-line SSM interface, **hpssadm**. This tool does not provide all the functionality of the **hpssgui**, but does implement a subset of its frequently used features, such as some monitoring and some control of servers, devices, storage classes, volumes, and alarms. It is useful for performing HPSS administration from remote locations where network traffic is slow or difficult. Additionally, **hpssadm** provides some rudimentary mass configuration support by means of the ability to issue configuration commands from a batch script.

In addition to SSM, HPSS provides a number of command-line utilities for specialized management purposes, such as listing the volumes managed by a particular PVR or core server. See the *Management tools* chapter of the *HPSS Management Guide* for more information. See also the HPSS man pages for descriptions of these utilities.

2.3.7. HPSS policy modules

There are a number of aspects of storage management that probably will differ at each HPSS site. For instance, sites typically have their own guidelines or policies covering the implementation of accounting, security, and other storage management operations. In order to accommodate site-specific policies, HPSS has implemented flexible interfaces to its servers to allow local sites the freedom to tailor management operations to meet their particular needs.

HPSS policies are implemented using two different approaches. Under the first approach, used for migration, purge, and logging policies, sites are provided with a large number of parameters that may be used to implement local policy. Under the second approach, HPSS communicates information through a well-defined interface to a policy software module that can be completely replaced by a site. Under both approaches, HPSS provides a default policy set for users.

Migration policy

The migration policy defines the conditions under which data is copied from one level in a storage hierarchy to one or more lower levels. Each storage class that is to have data copied from that storage class to a lower level in the hierarchy has a migration policy associated with it. The MPS uses this policy to control when files are copied and how much data is copied from the storage class in a given migration run. Migration runs are started automatically by the MPS based upon parameters in the migration policy.

Note that the number of copies which migration makes and the location of these copies is determined by the definition of the storage hierarchy and not by the migration policy.

Purge policy

The purge policy defines the conditions under which data that has already been migrated from a disk storage class can be deleted. Purge applies only to disk storage classes. It is common, but not necessary, for disk storage classes which have a migration policy to also have a purge policy. Purge runs are started automatically by the MPS based upon parameters in the purge policy.

Logging policy

The logging policy controls the types of messages to log. On a per-server basis, the message types to write to the HPSS log may be defined. In addition, for each server, options to send Alarm and Event messages to SSM may be defined.

Security policy

Security policy defines the authorization and access controls to be used for client access to HPSS. HPSS security policies are provided to control access (authentication) from FTP or Parallel FTP (or both) using **Username/Password** or **Kerberos** credentials.

HPSS provides facilities for recording information about authentication and object (file and directory) creation, deletion, access, and authorization events. The security audit policy for each server determines the records that each individual server will generate. All servers can generate authentication records.

Accounting policy

The accounting policy provides runtime information to the accounting report utility and to the account validation service of the Gatekeeper. It helps determine what style of accounting should be used and what level of validation should be enforced.

The two types of accounting are site-style and UNIX-style. The site-style approach is the traditional type of accounting in use by most mass storage systems. Each site will have a site-specific table (Account Map) that correlates the HPSS account index number with their local account charge codes. The UNIX-style approach allows a site to use the User Identifier (UID) for the account index. The UID is passed along in UNIX-style accounting just as the account index number is passed along in site-style accounting.

Account validation allows a site to perform usage authorization of an account for a user. It is turned on by enabling the Account Validation field of the *Accounting Policy* configuration screen. If account validation is enabled, the accounting style in use at the site is determined by the Accounting Style field. A site policy module may be implemented by the local site to perform customized account validation operations. The default account validation behavior is performed for any account validation operation that is not overridden by the site policy module.

Location policy

The location policy defines how Location Servers at a given site will perform, especially in regards to how often server location information is updated. All local, replicated Location Servers update information according to the same policy.

Gatekeeping policy

The Gatekeeper provides a gatekeeping service along with an account validation service. These services provide the mechanism for HPSS to communicate information though a well-defined interface to a policy software module that can be written by a site. The site policy code is placed in well-defined shared libraries for the gatekeeping policy and the accounting policy (`libgksite.[a|so]` for the gatekeeping policy and `libacctsite.[a|so]` for accounting) which are linked to the Gatekeeper. The Gatekeeper looks for these libraries in `/usr/local/lib64` first, and then `/opt/hpss/lib`. The gatekeeping policy shared library contains a default policy which does *no* gatekeeping. Sites will need to enhance this library to implement local policy rules if they wish to monitor and load-balance requests.

2.4. HPSS hardware platforms

2.4.1. Server platforms

HPSS requires at least one Linux node for the core server components. A server node must have sufficient processing power and memory to handle the workload.

2.4.2. Client platforms

The full-function Client API can be ported to any platform that supports UNIX.

The PFTP client code and Client API source code for platforms other than AIX and Linux are not on the HPSS distribution image. Maintenance of the PFTP and Client API software on platforms other

than AIX and Linux is the responsibility of the customer, unless a support agreement is negotiated with IBM. Contact HPSS support for information on how to obtain the needed software.

The following matrix illustrates which platforms support HPSS interfaces.

Table 2.1. HPSS client interface and Mover platforms

Platform	HPSS Mover	PFTP client	Client API	FTP clients *(see Note 1)*
IBM AIX		X	X	X
Oracle Solaris x86		X	X	X
RHEL (x86)	X	X	X	X
RHEL (Power PC)	X	X	X	X

Note 1: GUI-based clients may not function correctly for some commands.
Note 2: For compatibility of HPSS applications such as HSI/HTAR, HPSSFS-FUSE, and others, consult application documentation.

2.4.3. Mover platforms

Movers are used to control the logical network attachment of storage devices and are configured to run on one or more nodes. A Mover consists of two parts: The Mover administrative process that runs on the server node, and the remote Mover process that handles the HPSS devices and data transfers. See *Table 2.1, "HPSS client interface and Mover platforms"* above for a detailed list of supported platforms.

Chapter 3. HPSS planning

3.1. Overview

This chapter provides HPSS planning guidelines and considerations to help the administrator effectively plan, and make key decisions about, an HPSS system.

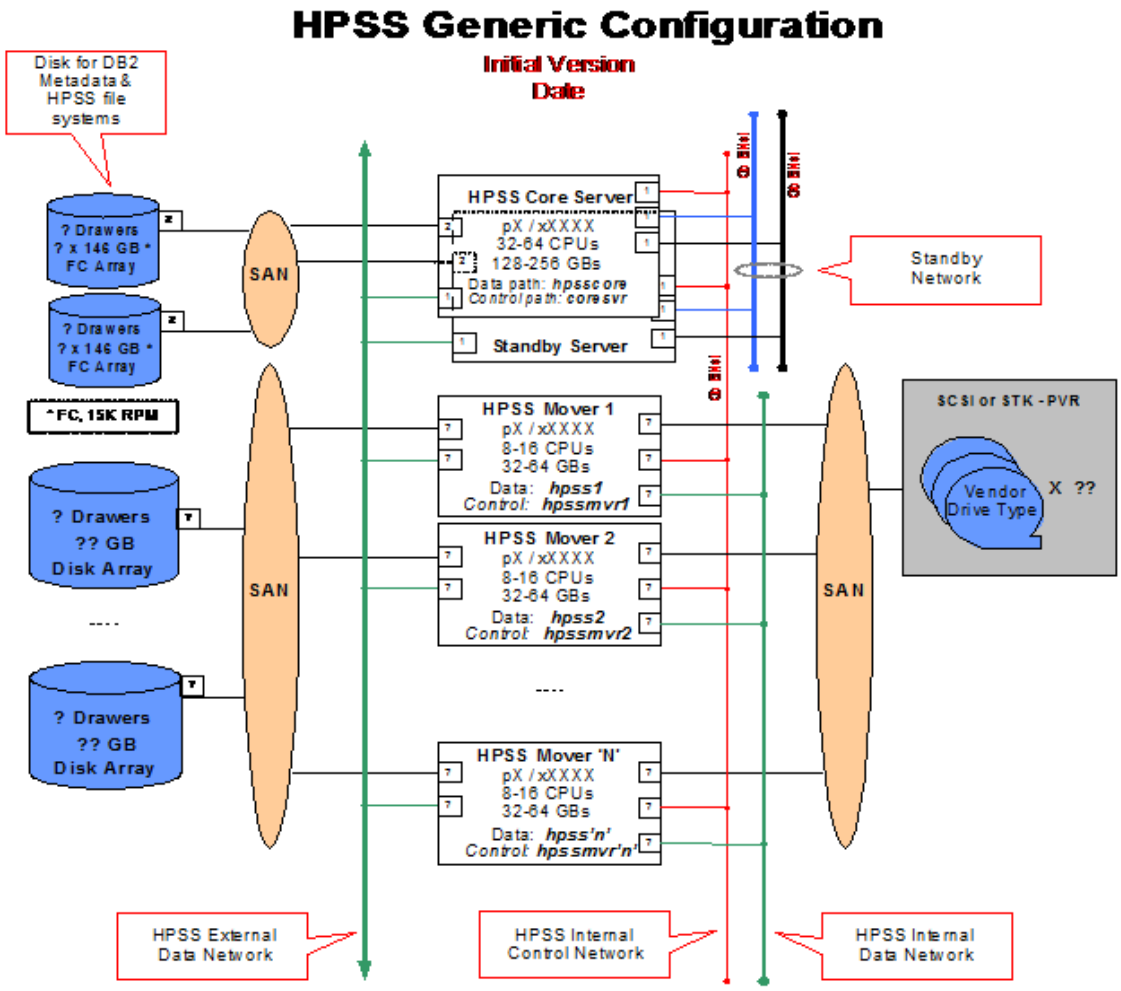
The planning process for HPSS must be done carefully to ensure that the resulting system satisfies the site's requirements and operates in an efficient manner. We recommend that the administrator read this entire chapter before planning the system.

The following paragraphs describe the recommended planning steps for the HPSS installation, configuration, and operational phases.

3.1.1. HPSS system architecture

Before getting into the details of storage sizing, it is best to understand the overall HPSS system and how the components are arranged to build the HSM. The following illustration shows the basic architecture of an HPSS system including disk and tape resources and their relationship to HPSS server nodes, Mover nodes, internal and external networks, and SAN interconnections. Specifics of this architecture for a given site are developed during the proposal and initial project planning stages of a deployment. Ideally, the required space is derived from requirements gathered from the HPSS Questionnaire document, known operational constraints, transfer and transaction rates, and anticipated storage capacity needs. Often the disk and tape resources are dictated by current equipment already available and budgetary constraints on what can be purchased. Specific quantities and sizing of these resource are beyond the scope of this planning document. These are largely defined by the above inputs and negotiations during the initial planning meeting in which the systems engineering team draws from experience and similar deployments to design a working architecture that balances the end-user requirements with the potential, or actual, resources available.

Figure 3.1. HPSS generic configuration



3.1.2. HPSS configuration planning

Before beginning the planning process, there is an important issue to consider. HPSS was designed to optimize the transfer of large files at the expense of some small file transfer performance. If at all possible, try to reduce the number of small files that are introduced into your HPSS system. For example, if you plan to use HPSS to backup all of the PCs in your organization, it would be best to aggregate the individual files into large individual files before moving them into the HPSS name space.

The following planning steps must be carefully considered for the HPSS infrastructure configuration and the HPSS configuration phases:

1. Identify the site's storage requirements and policies, such as the initial storage system size, anticipated growth, usage trends, average file size, expected throughput, backup policy, and availability. For more information, see *Section 3.2, "Requirements and intended uses for HPSS"*.

2. Define the architecture of the entire HPSS system to satisfy the above requirements. The planning should:

   - Identify the nodes to be configured as part of the HPSS system.

- Identify the type of network that will be used for the HPSS system. HPSS allows the administrator to configure the system to use IPv4-only (the default), IPv6 mixed-mode (where IPv6 is preferred), and IPv6-only mode.

  Note: The HPSS STK PVR is not supported on IPv6 due to the Oracle StorageTek CDK ACSAPI library not supporting it at this time. If you require this support, consult your Oracle representative about implementing this enhancement.

- Identify the disk and tape storage devices to be configured as part of the HPSS system and the nodes and networks to which each of the devices will be attached. Storage devices can be assigned to a number of nodes to allow data transfers to utilize the devices in parallel without being constrained by the resources of a single node. This capability also allows the administrator to configure the HPSS system to match the device performance with the performance of the network used to transfer the data between the HPSS Movers and the end users (or other HPSS Movers in the case of internal HPSS data movement for migration and staging). Refer to *Section 3.4, "Hardware considerations"* for more discussions on the storage devices and networks supported by HPSS.

- Identify the HPSS subsystems to be configured and how resources will be allocated among them. Refer to *Section 3.8, "Storage subsystem considerations"* for more discussion on subsystems.

- Identify the HPSS servers to be configured and the node where each of the servers will run. Refer to *Section 3.7, "HPSS server considerations"* for more discussions on the HPSS server configuration.

- Identify the HPSS user interfaces (such as FTP, PFTP, or Client API) to be configured and the nodes where the components of each user interface will run. Refer to *Section 3.6, "HPSS interface considerations"* for more discussion on the user interfaces supported by HPSS.

3. Ensure that the prerequisite software has been obtained, installed, and configured properly in order to satisfy the target HPSS architecture. Refer to *Section 3.3, "Prerequisite software considerations"* for more information on the HPSS prerequisite software requirements.

4. Determine the DB2 disk storage space needed to satisfy the requirements of the HPSS system, and verify there is sufficient free space in the file systems to meet those needs. Refer to *Section 3.5.2.2, "HPSS metadata space"* for more discussion of file system storage requirements for HPSS and DB2.

5. Verify that each of the identified nodes has sufficient resources to handle the workloads to be imposed on the node. Refer to *Section 3.5.3, "System memory and disk space"* for more discussions on the system resource requirements.

6. Plan the design of the HPSS storage characteristics and HPSS storage space to satisfy the site's requirements:

   - Plan for file families, if any. Refer to *Section 3.10.4, "File families"* for more information about configuring families.

   - Plan for filesets and junctions, if any. Refer to the *Filesets and junctions* chapter of the *HPSS Management Guide* for more information.

- Plan for HPSS storage classes. Refer to *Section 3.10.1, "Storage class"* for more information on the storage class configuration.
- Plan for HPSS storage hierarchies. Refer to *Section 3.10.2, "Storage hierarchy"* for more information on the storage hierarchy configuration.
- Plan for HPSS classes of service. Refer to *Section 3.10.3, "Class of Service"* for more information on the Class of Service configuration.
- Plan the migration and purge policy for each storage class. Refer to *Section 3.9.1, "Migration policy"* and *Section 3.9.2, "Purge policy"* for more information.
- Determine the amount of user data storage space needed for each storage class. Refer to *Section 3.5.1, "HPSS user storage space"* for more information on the HPSS storage space considerations.
- Identify the disk and tape media to be imported into HPSS.

7. Define the location policy to be used. Refer to *Section 3.9.6, "Location policy"* for more information.
8. Define the accounting policy to be used. Refer to *Section 3.9.3, "Accounting policy and validation"* for more information on the accounting policy configuration.
9. Define the logging policy for each of the HPSS servers. Refer to *Section 3.9.5, "Logging policy"* for more information on the logging policy configuration.
10. Define the security policy for the HPSS system. Refer to *Section 3.9.4, "Security policy"* for more information on the security policy for HPSS.
11. Determine if a Gatekeeper will be required. It is required if a site wants to do account validation or gatekeeping. Refer to *Section 3.9.3, "Accounting policy and validation"* and *Section 3.9.7, "Gatekeeping"* for more information.
12. Identify the site's need for User-defined Attributes, if any. Planning should include determining the amount of DB2 disk storage space needed to satisfy their UDA requirements, deciding which attributes should have indexes, and developing an XML schema if desired.

3.1.3. Purchasing hardware and software

It is recommended that hardware be purchased only after the HPSS configuration has been planned. Purchasing the hardware prior to the planning process may result in performance and utilization issues that could easily be avoided by advance planning.

When purchasing Linux servers for storage purposes, note that the operating system limitations will only allow a fixed number of raw devices to be configured per logical unit (disk drive or disk array). Linux operating system limits SCSI disks to 15 partitions and limits IDE disks to 63 partitions, unless LVM is used. These limits can potentially impact the utilization of a disk drive or disk array.

Refer to *Section 3.5, "HPSS sizing considerations"* for more information on calculating the number and size of devices that will be needed to meet your requirements.

Refer to *Section 3.3, "Prerequisite software considerations"* for more information on the required software that will be needed to run HPSS.

3.1.4. HPSS operational planning

The following planning steps must be carefully considered for the HPSS operational phase:

1. Define the site guidelines for the HPSS users and SSM users.

   - Each HPSS user who uses the storage services provided by HPSS should be assigned an Accounting ID and one or more appropriate Classes of Service (COS) to store files.

   - Each SSM user (administrator or operator) should be assigned an appropriate SSM security level. The SSM security level defines what functions each SSM user can perform on HPSS through SSM. Refer to the *SSM user security* section, the *Creating the SSM user accounts* section, and the *Add an SSM user ID* section of the *HPSS Management Guide* for more information on setting up the security level for an SSM user.

2. Define the site procedures for repacking and reclaiming HPSS tape volumes. Define the tape consolidation and reuse goals. For instance, define a tape utilization factor and plan to repack those tapes that fall below that limit. The limits can be set on a per storage class basis. Also, decide when or if empty tapes will be reclaimed. Refer to the *Repack Virtual Volumes window* section and the *Reclaim Virtual Volumes window* section (both in the *HPSS Management Guide*) for more information.

3. Define the site policy and procedure for generating the accounting reports. Take into consideration how often an accounting report needs to be generated, how the accounting information from the report will be used to produce the desired cost accounting, and whether the accounting reports need to be archived. Refer to *Section 3.9.3, "Accounting policy and validation"* and the *Accounting* section of the *HPSS Management Guide* for more information on defining an accounting policy and generating accounting reports.

4. Determine if gatekeeping (monitoring or load-balancing) will be required. If so, define and write the site policy code for gatekeeping. Refer to *Section 3.9.7, "Gatekeeping"* for more information on gatekeeping, and refer to the *HPSS Programmers Reference* for guidelines on implementing the Site Interfaces for the Gatekeeping Service.

5. Determine the desired site trashcan settings. Take into consideration the number of threads that will be devoted to the incineration process, how often the incinerator will run, the amount of time needed before a file becomes eligible for incineration, and how often the statistics thread will run. Refer to the *HPSS trashcans* section of the *HPSS Management Guide* for more information on trashcan settings.

3.1.5. HPSS deployment planning

The successful deployment of an HPSS installation is a complicated task which requires reviewing client and system requirements, integration of numerous products and resources, proper training of users and administrators, and extensive integration testing in the client environment.

To help the HPSS system administrators in all of these tasks, a set of procedures have been developed to supplement this document. The HPSS Deployment Process is a document maintained by the HPSS

deployment team and contains a detailed outline of what is required to bring up an HPSS system, from an initial introduction and review of the environment to production use. This document is provided to customers at the initial HPSS planning meeting. The deployment procedures include a timeline plus checklist that the HPSS customer installation/system administration team should use to keep the deployment of an HPSS system on track. This is the same guide that HPSS support uses to monitor and check the progress of an installation.

3.2. Requirements and intended uses for HPSS

This section provides some guidance for the administrator to identify the site's requirements and expectations of HPSS. Issues such as the amount of storage needed, access speed and data transfer speed, typical usage, security, expected growth, data backup, and conversion from an old system must be factored into the planning of a new HPSS system.

3.2.1. Storage system capacity

The amount of HPSS user data storage space the administrator must plan for includes the following considerations:

- The amount of user data storage space required to support new user accounts.
- The amount of user data storage space required to support expected growth of current accounts.
- The amount of user data storage space required to support storage management activities such as migration and repack.
- The amount of user data storage space required to support duplicate copies of user files.

Another component of storage space planning is the amount of space needed for HPSS system metadata. Refer to *Section 3.5.1, "HPSS user storage space"* and *Section 3.5.2.2, "HPSS metadata space"* for more information on determining the needed storage space and metadata space.

3.2.2. Required throughputs

Determine the required or expected throughput for the various types of data transfers that users will perform. Some users want quick access to small amounts of data. Other users have huge amounts of data they want to transfer quickly, but are willing to wait for tape mounts or other delays. In all cases, plan for peak loads that can occur during certain time periods. These findings must be used to determine the type of storage devices and network to be used with HPSS to provide the needed throughput.

3.2.3. Load characterization

Understanding the kind of load users are putting on an existing file storage system provides input that can be used to configure and schedule the HPSS system. What is the distribution of file sizes? How many files and how much data is moved in each category? How does the load vary with time (over a day, a week, a month)? Are any of the data transfer paths saturated?

Having this storage system load information helps to configure HPSS so that it can meet the peak demands. Also based on this information, maintenance activities such as migration, repack, and reclaim can be scheduled during times when HPSS is less busy.

3.2.4. Usage trends

To configure the system properly the growth rates of the various categories of storage, as well as the growth rate of the number of files accessed and data moved in the various categories must be known. Extra storage and data transfer hardware must be available if the amount of data storage and use are growing rapidly.

3.2.5. Duplicate file policy

The policy on duplicating user data files impacts the amount of data stored and the amount of data moved. If all user files are duplicated, the system will require twice as much tape storage. Users can be given control over duplication of their files by allowing them a choice between hierarchies which provide duplication and hierarchies which do not.

3.2.6. Charging policy

HPSS does not charge users for the use of storage system resources. Instead, it collects accounting information which a site can use to implement a charging policy. Even if sites do not charge for HPSS usage, the accounting information should be examined regularly in order to understand the system resource utilization. This information can assist sites in adjusting their system configuration to better handle their workload.

3.2.7. Security

Authentication and authorization between HPSS servers is done through the use of either UNIX or Kerberos security tools for authentication and either UNIX or LDAP for authorization services. By default, servers are authenticated using the Kerberos authentication service, and authorization information is obtained from the UNIX authorization service. The default protection level passes authentication tokens on the first remote procedure call to a server. The authentication service, authorization service, and protection level for each server can be configured to raise or lower the security of the system. Two cautions should be noted: (1) raising the protection level to packet integrity or packet privacy will require additional processing for each RPC, and (2) lowering the authentication service to none effectively removes the HPSS authentication and authorization mechanisms. Lowering the authentication service level should only be done in a trusted environment.

Each HPSS server authorizes and enforces access to its interfaces through access control lists stored in the AUTHZACL table. To modify the server state, control access is required. Generally, this is given only to the principal associated with the HPSS system administrative component, which by default is hpssssm. (This principal is defined by the HPSS_PRINCIPAL_SSM environment variable.) Additional principals can be allowed or denied access by setting permissions appropriately. See the *HPSS server security ACLs* section of the *HPSS Management Guide* for more information.

Security auditing in each server may be configured to record all, none, or some security events. Some sites may choose to log every client connection; every bitfile creation, deletion, and open; and every file management operation. Other sites may choose to log only errors. See the security

information fields in the general server configuration (in the *Server configuration* section of the *HPSS Management Guide*) for more details.

User access to HPSS interfaces depends on the interface being used. Access through the native Client API uses the UNIX or Kerberos authentication services and UNIX or LDAP authorization services described above. FTP or Parallel FTP access may utilize the HPSS password file, a configurable password file, or the Kerberos credentials. Additional FTP access is available using Kerberos GSS credentials. Refer to the *FTP* section of the *HPSS User's Guide* for additional details.

3.2.7.1. Cross realm access

Kerberos provides facilities for secure communication between multiple Kerberos realms (domains) referred to as Trusted "Cross Realm" access. These features use the Kerberos facilities to provide a trusted environment between cooperating locations. HPSS uses the Kerberos Cross Realm features for authentication. The procedures for interconnecting Kerberos realms are outlined in the *Security services* section of the *HPSS Management Guide*. The HPSS Parallel FTP program can utilize the Kerberos and HPSS Cross Realm access features.

The Generic Security Service (GSS) FTP, available from the Massachusetts Institute of Technology, and the HPSS Parallel FTP applications can take advantage of the Cross Realm access features for authentication and authorization (subject to certain caveats: see FTP documentation for details). The **pftp_client** binary must be built using the distributed source code. However, it is the site's responsibility to obtain the necessary Kerberos components.

ACLs entries in the AUTHZACL table and/or ACLs on HPSS directories and files may need to be added for appropriate foreign_user or foreign_group (or both) entries.

3.2.8. HPSS availability options

When configured with the appropriate redundant hardware, HPSS has some add-on options that allow a system to continue operations with a minimal amount of downtime: Automatic HA with Cluster Management Software, Core Server Manual Failover, and Core Server HADR Failover.

Automatic HA with Cluster Management Software
The Automated HA option for HPSS uses cluster management software to provide high availability. With RHEL 6, Red Hat's Cluster Suite Services is used to facilitate HA for HPSS. For RHEL 7, Tivoli System Automation for Multiplatforms (TSA/MP) is used to implement cluster services for the HA HPSS solution.

Core Server Manual Failover
A pair of identically-configured servers with a shared set of metadata and file system resources are used as the core server redundant pair. Only one node is active at any time, running the Core Services and DB2 components. A set of scripts allow the admin to start and stop service on each node, and the start script prevents the accidental startup of HPSS or DB2 if already active on the other node. This option protects against node loss, but not against the total loss of the shared metadata and file system resource.

Core Server HADR Failover
Like the Core Server Manual failover option, a pair of identically-configured servers are used; however, each node has its own set of metadata and file system resources. In normal operational

mode, a copy of DB2 runs on each node: one acting as the HADR primary copy, the other operating as the HADR secondary copy. The secondary server must be active and ready to receive logs and replay transactions from the primary. A set of scripts are used to start and stop HPSS on the primary copy node, and another set of scripts are used to switch HADR roles between the nodes. This option protects against the loss of one server node or one of the metadata and file system resources.

3.3. Prerequisite software considerations

This section defines the prerequisite requirements for HPSS. Some products must be obtained separately from HPSS and installed prior to the HPSS installation and configuration.

3.3.1. Prerequisite software overview

This section describes the prerequisite software packages required by HPSS and provides information to obtain them. Refer to the *HPSS Release Notes* for specific versions.

3.3.1.1. DB2

HPSS uses DB2 for Linux, UNIX and Windows by IBM Corporation to manage all HPSS metadata. DB2 software is included in the HPSS distribution. Refer to *Section 5.2.7, "Install DB2 and set up permanent license"* for more information. The required DB2 FixPak must be downloaded and installed after the DB2 base is installed. DB2 FixPaks can be downloaded from the DB2 for Linux, UNIX, and Windows webpage at http://www-01.ibm.com/support/docview.wss?rs=71&uid=swg27007053.

3.3.1.2. OpenSSL

HPSS uses the OpenSSL Project's cryptographic library for securing UNIX authentication. OpenSSL comes bundled with AIX and Linux. Sites using Kerberos with AIX will need an updated version of the standard AIX package of OpenSSL. Refer to http://www.openssl.org/ for additional information about OpenSSL.

3.3.1.3. Kerberos

HPSS uses the Massachusetts Institute of Technology (MIT) Kerberos to implement Kerberos authentication. MIT Kerberos is a network authentication protocol designed to provide authentication for client/server applications by using secret-key cryptography. A free implementation of this protocol can be downloaded from the MIT website (http://web.mit.edu/kerberos/). Refer to for more information.

For Linux, Kerberos is installed as part of the operating system. Sites running AIX will need to install the appropriate level of Kerberos as an additional step. If building Kerberos from source on AIX, contact HPSS support if you need additional guidance.

3.3.1.4. LDAP and IBM Kerberos

HPSS can be configured to use an LDAP directory to store its authorization information such as users' names, UIDs, GIDs, and home directories. The supported LDAP server products for this

release are IBM Tivoli Directory Server and OpenLDAP. TDS can be downloaded from IBM.com [http://www.ibm.com]. OpenLDAP can be downloaded from the OpenLDAP project page [http://www.openldap.org/].

Installing the IBM Kerberos client is only required if TDS is being used for authorization and the LDAP daemon will be used for Kerberos authentication. This option is supported only on AIX. The fileset for the IBM Kerberos client is located on the AIX Expansion Pack CD.

OpenLDAP is necessary for Kerberos authentication on Linux. OpenLDAP uses MIT Kerberos, even on AIX, so installing IBM Kerberos would be unnecessary for setting up an OpenLDAP server.

For either case, if Kerberos authentication to LDAP is required, OpenLDAP's client API will be required for compilation into HPSS.

3.3.1.5. Java

HPSS uses the Java Standard Edition to implement the SSM graphical user interface, **hpssgui**, the SSM command-line interface, and **hpssadm**.

The Java product required for the AIX and RHEL platforms can be downloaded from IBM's download webpages:

http://www.ibm.com/developerworks/java/jdk/aix/service.html

http://www.ibm.com/developerworks/java/jdk/linux/download.html.

3.3.1.6. Use of libTI-RPC

HPSS uses the transport-independent RPC (TI-RPC) library for server communications. This library is available via standard repositories and installation media for RHEL.

3.3.1.7. Jansson

Some HPSS servers make use of the Jansson library for converting output into the JSON format. This library is available via standard repositories and installation media for RHEL. For RHEL 8.x, the required version is 2.11.

3.3.1.8. STK Tools

For sites which use an STK PVR, a set of tools to enable communication between the STK PVR and the tape robot is needed, called "STK Toolkit". This toolkit will be provided by HPSS support.

3.3.2. Prerequisite summary By HPSS node type

This section provides a summary list of prerequisite software required for HPSS. It also lists the software versions which have been verified with HPSS 7.5.

3.3.2.1. HPSS server nodes

This section describes the prerequisite software required for each server node.

Linux requirements

- *See the release notes for the HPSS release you are using for the latest information on software prerequisites for the Linux server node.*
- Linux Kernel parameter changes:
  - ASLR or Address Space Layout Randomization is a feature activated by default on some of the newer Linux distributions. It is designed to load shared memory objects in random addresses. In DB2, multiple processes map a shared memory object at the same address across the processes. It was found that DB2 cannot guarantee the availability of the address for the shared memory object when ASLR is turned on.
  - Turn off the randomization by setting the following kernel parameter in `/etc/sysctl.conf`: `kernel.randomize_va_space=0`
  - The system must be rebooted for the change to be recognized.

HPSS Mover nodes

A Mover consists of two processes: the Mover administrative process that runs on the server node, and the remote Mover process that handles the HPSS devices and data transfers. To maximize performance, the remote Mover should not be placed on a node with DB2 and HPSS subsystem servers.

Since HPSS security, logging, and metadata services are performed on the server node, no additional software, like DB2 or HPSS servers, need be installed on the remote Mover node.

Linux requirements

See the release notes for the HPSS release you are using for the latest information on software prerequisites for the Linux Mover node.

3.3.2.2. HPSS client nodes

This section describes the prerequisite requirements for running HPSS clients.

SSM client requirements

The client node where the SSM **hpssgui** and **hpssadm** applications run must meet the following requirements:

- Supported platforms: AIX, Linux, Windows

Client API requirements

The client node where HPSS Client API applications run must meet the following requirements:

- For supported platforms, refer to *Section 2.4.2, "Client platforms"* for a complete list.

FTP/PFTP client requirements

The client node where HPSS FTP and PFTP run must meet the following requirements:

- For supported platforms, refer to *Section 2.4.2, "Client platforms"* for a complete list.

3.4. Hardware considerations

This section describes the hardware infrastructure needed to operate HPSS and includes considerations about infrastructure installation and operation that may impact HPSS.

3.4.1. Network considerations

Because of its distributed nature and high-performance requirements, an HPSS system is highly dependent on the networks providing connectivity among the HPSS servers and clients.

For control communications (that is, all communications except the actual transfer of data) among the HPSS clients and servers, HPSS requires TCP/IP services. Since control requests and replies are relatively small in size, a low-latency network usually is well suited to handling the control path.

The data path is logically separate from the control path and may also be physically separate, although this is not required. For the data path, HPSS supports the same TCP/IP networks as those supported for the control path. For supporting large data transfers, the latency of the network is less important than the overall data throughput.

HPSS provides support for IPv4 and IPv6 networking. By default, HPSS will only utilize IPv4 networks. If IPv6 is required, there are two new options: IPv6 mixed-mode (an IPv4/IPv6 environment that gives preferential treatment to IPv6 addresses) and IPv6-only mode that will only utilize IPv6 addresses.

The HPSS STK PVR is not supported on IPv6 due to the Oracle StorageTek CDK ACSAPI library not supporting it at this time. If you require this support, consult your Oracle representative about implementing this enhancement.

HPSS also supports a special data path option that may indirectly affect network planning because it may offload or shift some of the networking load. This option uses the shared memory data transfer method, which provides for intra-node transfers between either Movers or Movers and HPSS clients via a shared memory segment.

Along with shared memory, HPSS also supports a Local File Transfer (LFT) data path for client transfers that involve HPSS Movers that have access to the client's file system. In this case, the HPSS Mover can be configured to transfer the data directly to or from the client's file.

3.4.2. Robotically mounted tape

All HPSS PVRs are capable of sharing a library with other tape management systems but care must be taken when allocating drives among multiple library users. If it is necessary to share a drive between HPSS and another tape management system, the drive can be configured in the HPSS PVR but left in the LOCKED state until it is needed. To help the administrator remember why the drive is LOCKED, it is recommended that a text **Comment** be added to the drive metadata via the *PVL Drive Information* window or the *Tape Device Configuration* window to help the administrators recall why the drive is LOCKED; see the *HPSS Management Guide* for more information about device and drives management. When needed by HPSS, the drive should be set to UNLOCKED and should

not be used by any other tape management system while in this state. This is critical because HPSS periodically polls all of its unlocked drives even if they are not currently mounted or in use.

Generally, only one HPSS PVR is required per library. However, it is possible for multiple PVRs to manage a single library in order to provide drive and tape partitions within a library. The drives in the library must be partitioned among the PVRs and no drive should be configured in more than one PVR. Each tape is assigned to exactly one PVR when it is imported into the HPSS system and will only be mounted in drives managed by that PVR.

HPSS supports tape libraries from IBM, Spectralogic, and Oracle.

3.4.2.1. Drive-controlled LTO libraries (IBM, Spectralogic)

IBM and Spectralogic tape libraries and robots must be attached to a Linux workstation through a SCSI interface. In each case, the library shares a SCSI channel with one of the drives, so at least one of the drives in the library must be connected to the workstation. This workstation must be an HPSS node running the PVR. The SCSI PVR is used to communicate with these libraries.

The HPSS SCSI PVR is compatible with libraries which implement T10 SPC-3 and SMC-3 standards, and libraries which support a code set of 2 or 3.

3.4.2.2. Oracle StorageTek

The SCSI PVR may be used, as described above, for Oracle libraries which support a SCSI interface.

3.4.2.3. Oracle StorageTek tape libraries that support ACSLS

The STK PVR must be able to communicate with Oracle StorageTek's ACSLS server. HPSS requires any release of ACSLS version 7 or version 8. For example, the SL8500 supports ACSLS. For the STK PVR to communicate with the ACSLS server, it must have a TCP/IP connection to the server (for example, Ethernet) and the ACSLS Server System Interface (SSI) client software must be running on the node with the PVR. The client software, maintained by Oracle Corp., can be obtained through HPSS support; see *Section 3.7.6.1, "STK PVR"* and *STK PVR additional information* section in the *HPSS Management Guide* for additional information. Multiple Oracle StorageTek libraries can be connected via pass through ports and managed by a single ACSLS server. This collection of libraries can be managed by a single HPSS STK PVR.

3.4.2.4. ADIC AML

The AML PVR is supported by special bid only.

The Distributed AML Server (DAS) client components on the AIX workstations must be able to communicate (via a TCP/IP connected network) with DAS client components on the node controlling the robot in order to request DAS services. The AML PVR is used to communicate with the ADIC AML.

3.4.3. Manually mounted tape

An operator PVR is used to manage a homogeneous set of manually mounted drives. Multiple operator PVRs, with different drive types, can be configured without any additional considerations. Tape mount requests will be displayed on an SSM screen.

3.4.4. Tape devices

The tape devices/drives supported by HPSS are listed in the following table.

Table 3.1. Supported platform/driver/tape drive combinations

Platform	Driver	Devices
Linux	Native	3580 (Gen3, Gen4, Gen5, Gen6, Gen7, Gen8, Gen9), 3592 (Gen2, Gen3, Gen4, Gen5, Gen5A, Gen6, Gen7), 9840 (C, D), 9940 (A, B), T10000 (A, B, C, D)

The "Driver" column uses the following abbreviations:

Native Linux native SCSI Tape Device Driver

Older tape drives (3590, 3590E, 3590H, 9840 (A and B), 3580 (Gen1 and Gen2), 3592 (Gen1 and Gen2) will continue to be supported for existing HPSS sites until they can be upgraded.

3.4.4.1. Multiple media support

Certain drive types have the ability to mount multiple media formats. HPSS supports this ability by maintaining a drive preference table in the PVL for any media that can be mounted in multiple drive types. The drive preference table, drive availability, and mount operation are used in combination to determine what drive type to mount a particular media format in.

Idle drives are given priority over busy drives during the decision process.

The table below shows the drive preference for each media format that can be mounted in multiple drive types.

Table 3.2. Cartridge/drive affinity table

Cartridge type	Drive preference list	Operation supported
Single-Length 3590	Single-Length 3590 Double-Length 3590 Single-Length 3590E Double-Length 3590E Single-Length 3590H Double-Length 3590H	R/W R/W R R R R
Double-Length 3590	Double-Length 3590 Double-Length 3590E Double-Length 3590H	R/W R R
Single-Length 3590E	Single-Length 3590E Double-Length 3590E Double-Length 3590H	R/W R/W R
Double-Length 3590E	Double-Length 3590E	R/W

Cartridge type	Drive preference list	Operation supported
	Double-Length 3590H	R
Single-Length 3590H	Single-Length 3590H Double-Length 3590H	R/W R/W
Double-Length 3590H	Double-Length 3590H	R/W
3580 (LTO) Gen 1	3580 (LTO) Gen 1 3580 (LTO) Gen 2 3580 (LTO) Gen 3	R/W R/W R
3580 (LTO) Gen 2	3580 (LTO) Gen 2 3580 (LTO) Gen 3 3580 (LTO) Gen 4	R/W R/W R
3580 (LTO) Gen 3	3580 (LTO) Gen 3 3580 (LTO) Gen 4 3580 (LTO) Gen 5	R/W R/W R
3580 (LTO) Gen 4	3580 (LTO) Gen 4 3580 (LTO) Gen 5 3580 (LTO) Gen 6	R/W R/W R
3580 (LTO) Gen 5	3580 (LTO) Gen 5 3580 (LTO) Gen 6 3580 (LTO) Gen 7	R/W R/W R
3580 (LTO) Gen 6	3580 (LTO) Gen 6 3580 (LTO) Gen 7	R/W R/W
3580 (LTO) Gen 7	3580 (LTO) Gen 7 3580 (LTO) Gen 8	R/W R/W
3580 (LTO) Gen 7 M8	3580 (LTO) Gen 8	R/W
3580 (LTO) Gen 8	3580 (LTO) Gen 8 3580 (LTO) Gen 9	R/W R/W
3580 (LTO) Gen 9	3580 (LTO) Gen 9	R/W
3592 J1A JA/JW Tape 3592 J1A JJ/JR Tape	3592 J1A 3592 E05 (TS1120) 3592 E06 (TS1130) 3592 E07/EH7 (TS1140)	R/W R/W R R
3592 E05 JA/JW Tape 3592 E05 JB/JX Tape 3592 E05 JJ/JR Tape	3592 E05 (TS1120) 3592 E06 (TS1130) 3592 E07/EH7 (TS1140)	R/W R/W R
3592 E06 JA/JW Tape 3592 E06 JJ/JR Tape	3592 E06 (TS1130) 3592 E07/EH7 (TS1140)	R/W R
3592 E06 JB/JX Tape	3592 E06 (TS1130) 3592 E07/EH7 (TS1140)	R/W R/W

Cartridge type	Drive preference list	Operation supported
3592 E07/EH7 JC/JY Tape 3592 E07/EH7 JK Tape	3592 E07/EH7 (TS1140) 3592 E08/EH8 (TS1150) 3592 55E/F/G (TS1155) 3592 60E/F/G (TS1160)	R/W R/W R R
3592 E07/EH7 JB/JX Tape	3592 E07/EH7 (TS1140)	R/W
3592 E08/EH8 JC/JY Tape 3592 E08/EH8 JD/JZ Tape 3592 E08/EH8 JK Tape 3592 E08/EH8 JL Tape	3592 E08/EH8 (TS1150) 3592 55E/F/G (TS1155) 3592 60E/F/G (TS1160)	R/W R/W R/W
3592 55E/F/G JD/JZ Tape 3592 55E/F/G JL Tape	3592 55E/F/G (TS1155) 3592 60E/F/G (TS1160)	R/W R/W
3592 60E/F/G JE/JV Tape 3592 60E/F/G JM Tape	3592 60E/F/G (TS1160)	R/W
3592 70F/S JF Tape	3592 70F/S (TS1170)	R/W
STK 9840A STK 9840B	STK 9840A STK 9840B STK 9840C STK 9840D	R/W R/W R R
STK 9840C	STK 9840C STK 9840D	R/W R
STK 9840D	STK 9840D	R/W
STK 9940A	STK 9940A STK 9940B	R/W R
STK 9940B	STK 9940B	R/W
STK T10000A	STK T10000A STK T10000B STK T10000C STK T10000D	R/W R R R
STK T10000B	STK T10000B STK T10000C STK T10000D	R/W R R
STK T10000C	STK T10000C STK T10000D	R/W R
STK T10000D	STK T10000D	R/W

Note: HPSS generally refers to Oracle StorageTek media and hardware as STK. The STK tape cartridges listed above refer to the HPSS imported media type. So, an "STK T10000A cartridge" is a cartridge that was imported into HPSS as an "STK T10000A" cartridge and thus its label was written by an STK T10000A drive.

Additionally, for STK T10000 cartridges, Oracle supports different recording format density cartridges: T1 and T2. HPSS does not distinguish between these different cartridge density types. It is up to the administrator to import the cartridge using the correct and allowable media type for the cartridge. The following table is provided to help the administrator understand where these types can be used. *Consult the Oracle StorageTek documentation for the most up-to-date information.*

StorageTek T10000 tape drive	T10000 cartridge (T1)	T10000 T2 cartridge (T2)
T10000A	Read/Write	N/A
T10000B	Read/Write	N/A
T10000C	Read Only	Read/Write
T10000D	Read Only	Read/Write

3.4.5. Disk devices

HPSS supports locally-attached disk devices, including those devices attached via SCSI, SSA, or Fibre Channel. For these devices, operating system disk partitions of the desired size must be created (for example, a Linux disk partition), and the raw device name must be used when creating the Mover device configuration (see the *Configure a new device and drive* section of the *HPSS Management Guide* for details on configuring storage devices).

In addition, HPSS supports writing to sparse files residing on an underlying file system. This allows for the use of storage technologies that support file system interfaces, such as optical disks, or other types of disk appliances.

3.4.6. AWS Tape Gateway

The AWS Tape Gateway is one of several storage gateways provided by AWS. The AWS Tape Gateway provides cloud-based virtual tape storage for on-premises applications. This solution allows data to be stored in S3 in either Glacier Flexible Retrieval or Glacier Deep Archive tiers. HPSS can be used with the AWS Tape Gateway as a way to move HPSS data into cloud storage.

The AWS Tape Gateway provides an emulated tape library and tape drives. It is important to note that AWS Tape Gateway does not support tape zoning, logical block protection or SCSI reservations but otherwise can be configured just like a real tape library using the SCSI PVR. See the *HPSS Management Guide* for details on configuring an AWS Tape Gateway tape storage class or tape device.

Note that the data stored via AWS Tape Gateway will not be visible in AWS as files. The end result is a set of tape images that can be restored by HPSS.

3.4.7. Special bid considerations

The following options are available by special bid only:

- ADIC AML Tape Libraries
- Sony GY-8240
- HPSS High Availability

3.5. HPSS sizing considerations

There are two types of storage space that must be planned for: HPSS user storage space and HPSS infrastructure storage space.

HPSS user storage space is the disk and tape storage resources that will store user data. Typically, disk storage is allocated at the top level of storage hierarchies and is used as a disk cache for user data. Tape storage is usually allocated to lower levels of storage hierarchies and is used for the long-term, permanent storage of user data.

HPSS infrastructure storage space is the disk space allocated to file systems that contain executables, log files, server metadata (DB2 database), backups, and other HPSS supporting files and data. Tape resources outside of HPSS are usually required for backups of the operating system, HPSS specific file systems, and HPSS metadata unless other backup processes have been configured.

During the HPSS planning phase, it is important to assess how much disk space will be required to support the HPSS production environment. The first step in this process is to understand the various metadata tables managed by the HPSS system. The sections that follow explain the metadata table layout and how to best estimate disk space allocation to support these tables.

How these resources are interconnected to the overall system is just as important as the amount of disk or number of tape drives and cartridges allocated. For instance, if there are terabytes of disk storage in the form of several FC disk arrays and 50 enterprise type tape drives, but only one Mover and a couple of FC adapters, it is unlikely that the storage system will be able to adequately move data into, out of, and within the system to meet anyone’s demands and expectations. The "data pipes" between the storage resources must be adequately provisioned to allow for efficient transfer of data, including those times of peak demand. At the other extreme, one or both of the storage ends can be underallocated and waste the overall potential of the infrastructure. If there are too few tape drives, data stalls on the disk resources preventing new files from being transferred into the storage system, or from being staged back from tape media in a timely manner when the user requests access to it.

HPSS is a relational database application directly dependent on the speed and performance characteristics of the storage assigned to it. The following considerations must be given to this storage:

- Sufficient operational space to handle the expected amounts of metadata and user data (stored separately).
- Sufficient I/O capacity to handle the expected workload.

High-performance disk and tape systems for user data storage must be accompanied by high-performance storage supporting the HPSS database operations.

HPSS has the capability to take advantage of Storage Area Networks. Though separated in *Figure 3.1, “HPSS generic configuration”*, in reality there is usually only one SAN at an installation, and all the

resources are attached to it. Besides the HPSS Movers being connected to SAN, the end-user clients are often SAN-attached as well. The result is that the data paths take greater advantage of the SAN architecture, fewer store-and-forward operations are needed through Movers (that is, clients transfer data across SAN directly to disk resources, the Mover just manages the transfer), and less traffic across the TCP/IP network infrastructure. Adequately provisioning the "data pipes" is still critical, but the equation has changed to rely more heavily on the SAN to carry the data traffic.

3.5.1. HPSS user storage space

HPSS files are stored on the media that is defined and allocated to HPSS. Enough storage space must be provided to meet the demands of the user environment. HPSS assists in the management of space by providing SSM screens with information about total space and used space in all of the defined storage classes. In addition, alarms can be generated automatically based on configurable threshold values to indicate when space used in a given storage class has reached a threshold level. In a hierarchy where data is being migrated from one hierarchy level to a lower one, management of space in the storage class provided is done via the migration and purge policies that are provided. The basic factors involved are the total amount of media space available in the storage class being migrated and the rate at which this space is used. This will drive how the migration and purge policies are set up for the storage class. For more details on this, see *Section 3.9.1, "Migration policy"* and *Section 3.9.2, "Purge policy"*. Failure to provide enough storage space to satisfy a user request results in the user receiving a NO SPACE error. It is important to understand that the Core Server writes files only to the top level of the COS hierarchy. If the top level does not have sufficient free space for the write operation, it will fail, regardless of the amount of free space in lower levels.

3.5.2. HPSS infrastructure storage space

Figure 3.2, "HPSS Core Server and metadata resources" depicts a typical Core Server configuration and the metadata resources used by HPSS. The interconnect between the online storage resources and Core Servers varies on the specifics of a site, but usually falls into three categories: FC SAN, SAS, or Direct Connect. In all configurations, there is a certain amount of redundancy built into this configuration with the use of Dual Controllers in each storage array, multiple connection paths from the Core Servers, and in the case of an FC SAN - multiple SAN switches would be recommended. Additionally, key DB2 components like the DB2 log and DB2 logmirror would be separated onto different storage units. The goal of the configuration is to provide the greatest amount of protection and availability by using redundant components and strategically separating key resources across the hardware components. This is extended further with the use of a spare machine (for Manual Failover) or HADR system to provide redundancy for HPSS services as well.

Figure 3.2. HPSS Core Server and metadata resources

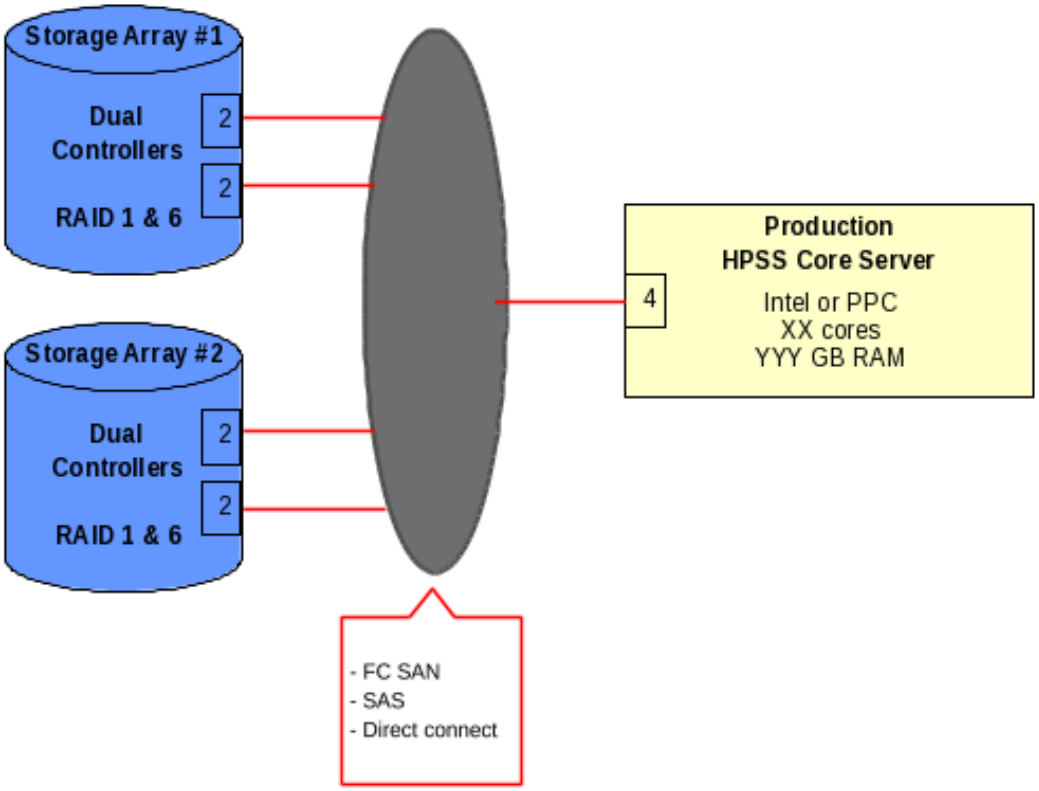


Figure 3.3, "Metadata disk layout - Rack 1" and *Figure 3.4, "Metadata disk layout - Rack 2"* show a possible allocation scheme using two E5624 with two trays of disks for each disk array.

Figure 3.3. Metadata disk layout - Rack 1

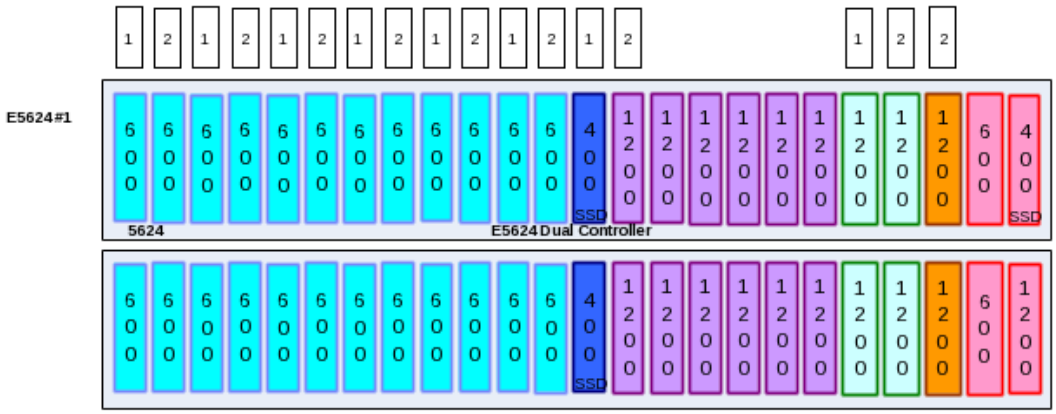
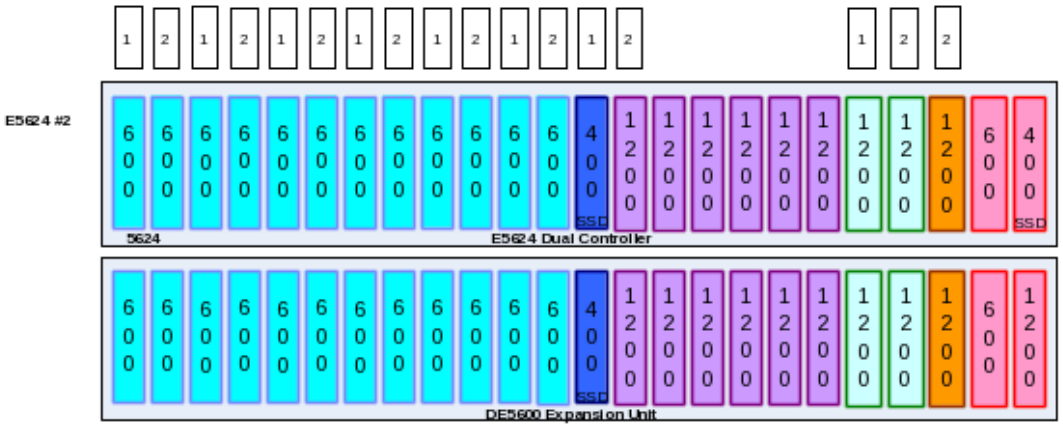


Figure 3.4. Metadata disk layout - Rack 2



Disk Array #1 configuration uses the following grouping of the disk resources:

- Twelve (1+1) RAID-1 arrays for the odd DB2 storage paths (light blue)
- One (1+1) RAID-1 array for the first copy of DB2 log (dark blue)
- One (10+2) RAID-6 for half of the DB2 backup space (purple)
- Two (1+1) RAID-1 arrays for HPSS file systems (green)
- One (1+1) RAID-1 array for DB2 log archive Mirror (brown)
- Four Hot Spare disks (red) (size and type vary)

Disk Array #2 configuration uses the following grouping of the disk resources:

- Twelve (1+1) RAID-1 arrays for the even DB2 storage paths (light blue)
- One (1+1) RAID-1 array for the second copy of DB2 log MIRROR (dark blue)
- One (10+2) RAID-6 for half of the DB2 backup space (purple)
- Two (1+1) RAID-1 arrays for HPSS file systems (green)
- One (1+1) RAID-1 array for DB2 log archive (brown)
- Four Hot Spare disks (red) (size and type vary)

The array model, disk sizes and types (spinning disk versus SSD), and grouping will vary with sites depending upon their operational requirements and available HW components. Specifics are developed with the HPSS Systems Engineering team and HPSS DB2 SMEs as needed. The above detail (and tables shown in Section 3.5.2.3, "HPSS file systems") are to only provide an example of how two storage arrays could be configured for a system that is designed to manage up to 500 million files. There are many considerations that must be taken into account to define the metadata storage which are well beyond the scope of this document. The examples, therefore, should only be used to understand the concept, and not used as an exact template for your site's situation.

The majority of the resources (light blue) will be allocated for DB2 metadata storage — this is storage for the HSUBSYS1 database. (The example does not cover the use of multiple subsystems nor the use of database partitioning.) Data is protected using a 1+1 RAID-1 configuration. Multiple LUNs are used, and reasonably high I/O performance can be attained across these resources; higher than what could be achieved if using multiple disks in a RAID 5 or RAID 6 configuration.

The DB2 log and logmirror also utilize a 1+1 RAID-1 configuration. While the RAID-1 provides data protection against a single disk failure, the log and logmirror are split between the two disk array units to provide additional protection against the loss of the entire disk array. This same arrangement is also in place for the first and second copy of the log-archives. This configuration redundancy eliminates a "single point of failure" in DB2 recovery objects (database backup images should have multiple copies as well) and allows the DB2 metadata databases to be recovered to the point of failure in the event of a complete disk array unit loss.

The remaining resources are allocated to the DB2 backup file system and HPSS supporting file systems (that is, where HPSS is installed). For the DB2 backup, the goal is to provide the maximum amount of space for the given set of resources, so a RAID-6 is used. Performance for this file system needs to be "good", but available space is more important. The HPSS supporting file systems have both lower space and performance requirements, but do need to be separate from other DB2 uses.

3.5.2.1. HPSS and DB2 file systems

The following subsections describe the various file systems used by HPSS.

/hpss_src

The HPSS software is installed in the `/hpss_src` directory. A symbolic link `/opt/hpss` will link to a subdirectory under `/hpss_src` where the appropriate HPSS version has been installed.

/var/hpss

See *Appendix F, The /var/hpss files* for a more detailed explanation of directories and files located in `/var/hpss`.

The `/var/hpss` directory tree is the default location of a number of HPSS configuration files and other files needed by the servers. It is recommended that this file system be at least 64 GB in size. A symbolic link `/var/hpss` references the `/varhpss` file system. Within the `/var/hpss` file system the following subdirectories exist:

- The `/var/hpss/etc` is the default directory where some additional UNIX configuration files are placed. These files are typically very small.
- The `/var/hpss/ftp` is the default directory where several PFTP daemon files are maintained. There are three subdirectories required: `adm` (contains the `hpss_ftpd.log` file), `daemon`, and `daemon/ftpd` where the `ftp.pids-hpss_class` file will be placed.
- The `/var/hpss/tmp` is the default directory where the Startup Daemon creates a lock file for each of the HPSS servers started on the node. HPSS may also write diagnostic log files and disk allocation maps in this directory, when configured to do so. The lock files are very small, but the logs and disk maps may be several tens of kilobytes, or larger.

It is up to the administrator to remove unneeded reports to prevent the `/var/hpss` *file system from filling.*

- The `/var/hpss/log` is the directory for HPSS log files. It is the recommended location for HPSS-specific log files; additional HPSS log files may also appear here.

- If the MPS is configured to produce a migration/purge report, it will generate a report every 24 hours. The size of these reports varies depending on the number of files being migrated and purged during the report cycle. These reports are placed in the `/var/hpss/mps` directory by default and will need to be removed when no longer needed.

  It is up to the administrator to remove unneeded reports to prevent the `/var/hpss` *file system from filling.*

- If the Gatekeeper is configured to do Gatekeeping Services, then the administrator may wish to create a site policy configuration file, usually named `/var/hpss/gk/gksitepolicy`. The size of this file depends on the site-implemented gatekeeping policy. If the Gatekeeper Service is not used, there is a minimal amount of disk space used in this directory.

- If an accounting report is requested, a report file and a checkpoint file are created in the directory specified in the accounting policy, usually `/var/hpss/acct`. The sizes of these files depend upon the number of files stored in HPSS.

  It is up to the administrator to remove unneeded reports to prevent the `/var/hpss` *file system from filling.*

- If SSM is configured to buffer alarm and event messages in a disk file, a file to store alarms and events will be created in `/var/hpss/ssm` directory. This alarm file is approximately 5 MB in size.

/hpss_corefile

is the default directory where HPSS servers put "core" files if they terminate abnormally. A symbolic link `/var/hpss/adm/core` is created to point to `/hpss_corefile`. Core files may be large, so it is recommended that there should be at least 64 GB reserved for this purpose on the server node and at least 4 GB on Mover nodes.

It is up to the administrator to remove unneeded core files to prevent the `/hpss_corefile` *file system from filling.*

/opt/ibm/db2

This is deployed as a symbolic link to a file system called `/optdb2`.

`/opt/ibm/db2` is the file system where the DB2 product is installed. Different versions of DB2 can be installed at the same time, and the size should be at least 64 GB.

The following DB2 file systems have a common base directory `/db2data`. This is a change from earlier releases of HPSS, but in an effort to help prevent accidental removal of critical DB2 files, we want the path to clearly identify this as DB2 data and users to be extra vigilant when operating in this part of the name space.

/db2data/db2_hpssdb

The `/db2data/db2_hpssdb` file system stores the DB2 instance configuration information and CFG database tables. It is also the HPSS instance home directory. Specifics on its size are described in the *CFG Database Allocation* section below. We recommend a file system size of 64 GB.

/db2data/p{0000, …, p-1}/stg{0001, …, t}

The DB2 storage path file systems are where DB2 stores the HPSS metadata. It takes the above form where "p" is the number of partitions, and "t" is the number of storage path file systems.

For most sites, the number of partitions will be 1, and there will be only one subsystem. The number of storage paths will be determined during the planning phase and the specifics of the metadata HW used for the site. The `/db2data/p####/stg####` file systems store the bulk of the HPSS metadata which is used to define the HPSS name space, bitfile information, disk and tape segments, disk and tape volumes, and activity information for migration/purge operations. When defining a new system, the HPSS Systems Engineering team needs to be consulted to help define or verify a configuration that will meet both the capacity and transaction requirements of the storage system.

Sites that have, or will have, hundreds of millions or billions of files, should consider the use of the DB2 Database Partitioning Feature (DPF), subsystems, or both. Planning for these features is beyond the scope of this document and will require additional consultation with the HPSS Systems Engineering and HPSS DB2 SME teams to customize a suitable configuration specific to the site's unique requirements.

/db2data/p{0000, …, p-1}/db2_backup1 & db2_backup2

The `/db2data/p####/db2_backup1` and `/db2data/p####/db2_backup2` file systems temporarily store backup images of the CFG and SUBSYS{X} databases. Typically subdirectories are used to segregate online versus offline backups, as well as separate subdirectories for each database. Something similar to the following:

```
…/online/cfg
…/online/subsys1
…/offline/cfg
…/offline/subsys1
```

Two file systems are used to store two complete backups on separate storage resources to protect the images from a single file system failure. After the backup images are generated, they are then transferred to long-term media, such as tape, using a backup file manager such as TSM. Details are described in *Section 3.5.2.2, "HPSS metadata space"*. Typically, backups are generated nightly, and at least a week's worth of backups should be kept on local file systems. Refer to the *IBM HPSS Operational Disaster Prevention Plan* document and consult with the HPSS Systems Engineering team to ensure enough space is reserved for this file system to support proper backups.

/db2data/p{0000, …, p-1}/db2_log & db2_logmirror

These two file systems contain the active transaction logs for the HPSS databases. The underlying storage for these file systems should have similar performance characteristics since DB2 transactions will wait for both log entries to be written. A separate subdirectory is defined under each file system for the CFG database and the SUBSYS databases:

```
…/cfg
…/subsys1
```

These file systems are typically backed by resources that support high-transaction I/O rates (for example, solid-state devices (SSD)). The amount of space required is usually small, a few tens of gigabytes, but I/O performance is critical to support HPSS metadata operations. For those sites with multiple subsystems, additional file systems (with separate storage resources) may be required for optimal performance. The "subdirectories" for the subsystems (subsys[2-5]) would then become mount points for these additional file systems.

Also, it is critical for the two file systems (`/db2data/db2_log` and `/db2data/db2_logmirror`) to be on separate metadata storage resources. Not only for performance but also for data protection. Loss of both file systems puts HPSS metadata at severe risk. Therefore, the "log" and "logmirror" are not only put on separate LUNs, but are required to be on separate storage arrays with dual paths to the controllers on each array to provide complete separation in the event of a catastrophic loss of a complete storage array.

/db2data/p{0000, …, p-1}/db2_logarchive1 & db2_logarchive2

Similar to DB2 log and logmirror file systems, two separate file systems are required to store completed ("archived") DB2 logs as they are generated. These files, in conjunction with the DB2 online backups, provide DB2 with the ability to recover DB2 to the point of failure in the event the primary DB2 resources are damaged, destroyed, or otherwise unavailable. The files in both file systems need to be safely copied to other media/storage like the DB2 backup images. The amount of storage for each file system should be adequate to store at least one week's worth of log archives, but more is recommended. It is up to the administrator to maintain the space in these file systems by periodically removing log archives that: 1) have been safely copied elsewhere, or 2) are no longer referenced by DB2 backup history. Refer to the *IBM HPSS Operational Disaster Prevention Plan* document for more information regarding the backup and restore process.

3.5.2.2. HPSS metadata space

During the HPSS planning phase, it is important to properly assess how much storage space and I/O capacity will be required by DB2 to manage HPSS metadata. The first step in this process is to understand the storage requirements for DB2. The sections that follow explain the various DB2 uses for storage and how best to estimate disk space allocation and capability to support DB2/HPSS operations.

CFG database allocation

By default, **mkhpss** will store the DB2-related files for HPSS in the `/db2data/db2_hpssdb` file system. As recommended above, this directory should be a separate file system of RAID disks. The amount of space needed will vary somewhat depending upon the number of subsystems in use. For a site with only one subsystem, the amount of space should be at least 64 GB.

SUBSYS database allocation

As a general rule of thumb, approximately 3500 bytes per file of storage should be reserved for the HPSS metadata storage paths. This general rule of thumb assumes:

- Table and index compression is enabled on large tables.

- Hierarchies with disk plus two levels of tape.
- A disk cache of no more than 10% of the total storage of the system.
  - With a reasonable/low number (less than 10 segments) of disk storage segments defined per file.
  - Systems that have a large number of disk segments or files with many holes may consume more storage space per file.
  - Systems that utilize UDAs may consume more storage space.

The calculations include additional space to allow for future in-place conversions and updates, since those may (or may not) utilize the same resources.

For sites with hundreds of millions or billions of files, the required disk space will require additional analysis since database partitioning will likely need to be utilized. The number of partitions will need to be discussed early in the planning stages of a new system, or a major upgrade. Consultation with the HPSS Systems Engineering and DB2 SME teams is required. Background information on database partitioning can be obtained from HPSS support, or it can be found by reviewing the DB2 information presented at the annual HPSS Users Forums, which are linked to from the HPSS Administrator Wiki.

DB2 log / DB2 log mirror

Recommended minimum: 200 GB for each file system per subsystem. Normally 10 log files (100 MB each) are required for each subsystem, but additional logs may be kept in the following situations:

- periods of heavy workloads
- long-running transactions
- when the log archive path is unavailable

DB2 log archives

Recommended minimum: 500 GB for each file system per subsystem, but this is dependent upon how busy the system is. Busier systems will need to have more space available. There should be at least enough space to keep one week's worth of log archive files (uncompressed) available for recovery. The HPSS Systems Engineering team will provide more guidance on appropriate sizing.

DB2 backup

Recommended minimum: 4 TB per subsystem for DB2 backup images that will be stored online; large subsystems will require additional space. Keep in mind this file system will contain separate nightly backup images for the last seven days (that is, a week; more is better).

Other file systems

Typically, the remaining file systems (`/var/hpss`, `/hpss`, `/opdb2`, `/hpss_corefile`, `/db2data/db2_hpssdb`) can be created out of shared LUN. There is no driving need for large amounts of space, nor high performance, for these file systems.

3.5.2.3. HPSS file systems

The following table provides a summary of what the file systems would look like (using the example of the E5624 disk arrays found earlier in this section).

Table 3.3. HPSS and DB2 file systems

HPSS and DB2 file systems There is one DB2 server				
LV name or device	**VG**	**Size ***	**File system mount point**	**Symlink to file system**
varhpss	hpss_vg	64 GB	/varhpss	/var/hpss
hpss_src	hpss_vg	64 GB	/hpss_src	
optdb2	hpss_vg	64 GB	/optdb2	/opt/ibm/db2
hpss_corefile	hpss_vg	64 GB	/hpss_corefile	/var/hpss/adm/core
db2_hpssdb	hpss_vg	64 GB	/db2data/db2_hpssdb	
db2_p0000_backup1	db2_p0000_backup1_vg	10.7 TB	/db2data/p0000/db2_backup1	
db2_p0000_backup2	db2_p0000_backup2_vg	10.7 TB	/db2data/p0000/db2_backup2	
db2_p0000_log	db2_p0000_log_vg	361 GB	/db2data/p0000/db2_log	
db2_p0000_logmirror	db2_p0000_logmirror_vg	361 GB	/db2data/p0000/db2_logmirror	
db2_p0000_logarch1	db2_logarch1_vg	1094 GB	/db2data/p0000/db2_logarchive1	
db2_p0000_logarch2	db2_logarch2_vg	1094 GB	/db2data/p0000/db2_logarchive2	
db2_p0000_stg0001	db2_p0000_stg0001_vg	545 GB	/db2data/p0000/stg0001	
db2_p0000_stg0002	db2_p0000_stg0002_vg	545 GB	/db2data/p0000/stg0002	
. . .	. . .	. . .	. . .	
db2_p0000_stg0024	db2_p0000_stg0024_vg	545 GB	/db2data/p0000/stg0024	

Mapping the LVs to LUN labels is shown in the following table:

Table 3.4. LV To LUN label mapping

LUN label	**VG**		**LUN label**	**VG**
db2_p0000_stg0001	db2_p0000_stg0001_vg		db2_p0000_stg0002	db2_p0000_stg0002_vg

LUN label	VG	LUN label	VG
db2_p0000_stg0003	db2_p0000_stg0003_vg	db2_p0000_stg0004	db2_p0000_stg0004_vg
. . .	. . .	. . .	. . .
db2_p0000_stg0023	db2_p0000_stg0023_vg	db2_p0000_stg0024	db2_p0000_stg0024_vg
db2_p0000_log	db2_log_vg	db2_p0000_logmirror	db2_p000_logmirror_vg
db2_p0000_backup1	db2_backup1_vg	db2_p0000_backup2	db2_p0000_backup2_vg
hpssfs1-1	hpss_vg	hpssfs2-1	hpss_vg
hpssfs1-2	hpss_vg	hpssfs2-2	hpss_vg
db2_p0000_logarch1	db2_p0000_logarch1_vg	db2_p0000_logarch2	db2_p0000_logarch2_vg

And LUN labels map back to the storage arrays given the following two tables:

Table 3.5. Storage Array #1

Array	Vol ID	Volume label	~Size	Config	Purpose
db2_p0000_stg0001	0	db2_p0000_stg0001	594 GB	1+1 RAID1	DB2 Storage
db2_p0000_stg0003	1	db2_p0000_stg0003	594 GB	1+1 RAID1	DB2 Storage
. . .	. . .	. . .	. . .	. . .	. . .
db2_p0000_stg0023	11	db2_p0000_stg0023	594 GB	1+1 RAID1	DB2 Storage
db2_p0000_log	12	db2_p0000_log	394 GB	1+1 RAID1	DB2 Log
db2_p0000_backup1	13	db2_p0000_backup1	11943 GB	10+2 RAID6	DB2 Backup
hpssfs1-1	14	hpssfs1-1	1194 GB	1+1 RAID1	HPSS File systems
hpssfs1-2	15	hpssfs1-2	1194 GB	1+1 RAID1	HPSS File systems
db2_p0000_logarch1	16	db2_p0000_logarch1	1194 GB	1+1 RAID1	DB2 Log Archive

Table 3.6. Storage Array #2

Array	Vol ID	Volume label	~Size	Config	Purpose
db2_p0000_stg0002	0	db2_p0000_stg0002	594 GB	1+1 RAID1	DB2 Storage
db2_p0000_stg0004	1	db2_p0000_stg0004	594 GB	1+1 RAID1	DB2 Storage

Array	Vol ID	Volume label	~Size	Config	Purpose
. . .	. . .	. . .	. . .	. . .	. . .
db2_p0000_stg0024	11	db2_p0000_stg0024	594 GB	1+1 RAID1	DB2 Storage
db2_p0000_log	12	db2_p0000_log	394 GB	1+1 RAID1	DB2 Log
db2_p0000_backup2	13	db2_p0000_backup2	11943 GB	10+2 RAID6	DB2 Backup
hpssfs2-1	14	hpssfs2-1	1194 GB	1+1 RAID1	HPSS File systems
hpssfs2-2	15	hpssfs2-2	1194 GB	1+1 RAID1	HPSS File systems
db2_p0000_logarch2	16	db2_p0000_logarch2	1117 GB	1+1 RAID1	DB2 Log Archive

3.5.3. System memory and disk space

The following sections discuss recommendations and requirements for disk space, system memory, and paging space.

3.5.3.1. Operating system disk spaces

It is recommended that all operating system logical volumes and partitions be mirrored. This is true of the HPSS server and Mover nodes.

3.5.3.2. System disk space requirements for running SSM

The SSM Graphical User Interface, **hpssgui**, and Command Line Interface, **hpssadm**, have an option to create session log files. **hpssgui** records all status bar and pop-up messages issued during the session in its log. **hpssadm** records all prompts, error and informational messages, and requested output (for example, lists, managed objects, and configuration structures) issued during the session in its log. Old session log files should be removed periodically to avoid filling the file system. This is typically done with a **cron** job. For example, the following command will remove all files from `/tmp` which have not been accessed within the previous seven days:

```
% find /tmp -atime +7 -exec rm {} \;
```

The creation of the session log files is controlled by the **-S** option to the **hpssgui** and **hpssadm** startup scripts. See their man pages for details.

3.5.3.3. System memory and paging space requirements

The memory and disk space requirements for the nodes where the HPSS servers will execute depends on the configuration of the servers, the nodes that each server will run on, and the amount of concurrent access they are configured to handle.

At least 2 GB of memory is recommended for nodes that will run one or more HPSS servers (and most likely a DB2 server), excluding the HPSS Movers. More memory is required for systems that run most of the servers on one node or support many concurrent users (or both). The memory available to HPSS and DB2 servers is critical to providing acceptable response times to end-user operations. Disk space requirements are primarily covered by *Section 3.5.2.2, "HPSS metadata space"* for DB2 space, and the preceding subsections under *Section 3.5.3, "System memory and disk space"* for the individual HPSS servers. Sufficient disk space should be allocated for the paging space, using recommendations in the system documentation for the amount of memory configured.

The amount of memory for nodes running HPSS Movers, and no DB2 servers, is dependent on the number and types of devices configured on the Mover node, the expected usages of those devices, and the configuration of the Movers. In general, Movers supporting disk devices will require more memory than Movers supporting tape devices because disk devices are likely to have more outstanding requests. At least 1 GB of memory should be configured on the Mover nodes. More memory is required for nodes that support many devices, especially disks, and have large numbers of concurrent end-user requests. Additionally, the size of the Mover's internal buffers impacts the Mover's memory requirements. Two buffers are used by each Mover process to handle I/O requests.

Paging space should be sized according to the following rules:

Table 3.7. Paging space info

Amount of physical memory	Minimum recommended amount of paging space
memory <= 256 MB	2.0 × amount of physical memory
256 MB < memory <= 1 GB	512 MB + ((amount of physical memory : 256 MB) × 1.25)
1 GB < memory <= 2 GB	1.5 × amount of physical memory
memory > 2 GB	1.0 × amount of physical memory

3.6. HPSS interface considerations

This section describes the user interfaces to HPSS and the various considerations that may impact the use and operation of HPSS.

3.6.1. Client API

The HPSS Client API provides a set of routines that allow clients to access the functions offered by HPSS. The API consists of a set of calls that are comparable to the file input/output interfaces defined by the POSIX standard (specifically *ISO/IEC 9945-1:1990* or *IEEE Standard 1003.1-1990*), as well as extensions provided to allow access to the extended capabilities offered by HPSS.

The Client API is built on top of the HPSS security layer (either UNIX or Kerberos). It must be run on a platform that supports the Core Server's security layer. For example, if the Core Server is using Kerberos authentication then users on the client platform must be able to authenticate themselves with the Core Server's Kerberos realm. To access HPSS from client platforms that do not support the Core Server's security layer, FTP or Parallel FTP must be used.

The Client API allows clients to specify the amount of data to be transferred with each request. The amount requested can have a considerable impact on system performance and the amount of metadata generated when writing directly to a tape storage class. See *Section 3.9.6, "Location policy"* and *Section 3.11, "HPSS performance considerations"* for further information.

The details of the Application Program Interface are described in the *HPSS Programmer's Reference*.

3.6.2. FTP

HPSS provides an FTP daemon that supports standard FTP clients. Extensions are also provided to allow additional features of HPSS to be utilized and queried. Extensions are provided for specifying Class of Service to be used for newly created files, as well as directory listing options to display Class of Service and accounting code information. In addition, the **chgrp**, **chmod**, and **chown** commands are supported as **quote site** options.

The FTP daemon is built on top of the Client API and must be run on a node that supports Kerberos clients. Note that FTP clients can run on computers that do not have Kerberos installed.

The size of the buffer used for reading and writing HPSS files can be specified in the FTP daemon configuration. The buffer size selected can have a considerable impact on both system performance and the amount of metadata generated when writing directly to a tape storage class. See *Section 3.9.6, "Location policy"* and *Section 3.11, "HPSS performance considerations"* for further information.

The GSSFTP from MIT is supported if the HPSS FTP daemon is appropriately configured. This client provides credential-based authentication and "Cross Realm" authentication to enhance security and "password-less" FTP features.

Refer to the *HPSS Management Guide* for details on configuring the FTP daemon.

Refer to the *HPSS User's Guide* for details of the FTP interface.

3.6.3. Parallel FTP

The FTP daemon also supports the HPSS Parallel FTP (PFTP) protocol, which allows the PFTP client to utilize the HPSS parallel data transfer mechanisms. This provides the capability for the client to transfer data directly to the HPSS Movers (that is, bypassing the FTP daemon), as well as the capability to stripe data across multiple client data ports (and potentially client nodes). Data transfers are supported through TCP/IP. Support is also provided for performing partial file transfers.

The FTP protocol is supported by the HPSS FTP daemon. Refer to the *FTP/PFTP daemon configuration* section of the *HPSS Management Guide* for configuration information. No additional configuration of the FTP daemon is required to support PFTP clients.

The client-side executable for PFTP is **pftp_client**, which supports TCP-based transfers. Because the client executable is a superset of standard FTP, standard FTP requests can be issued as well as the PFTP extensions. Authentication using either username/password or Kerberos credentials is configurable.

Refer to the *HPSS User's Guide* for details of the PFTP interface.

3.7. HPSS server considerations

Servers are the internal components of HPSS that provide the system's functionality. Each HPSS server executes as one or more UNIX processes. They must be configured correctly to ensure that HPSS operates properly. This section outlines key considerations that should be kept in mind when planning the server configuration for an HPSS system.

3.7.1. Core Server

The Core Server is responsible for managing the HPSS name space (such as files, directories, and links), bitfiles, and storage (such as physical volumes and virtual volumes) for a single subsystem. Each of these areas of responsibility are outlined in greater detail below.

Core Server at large.

The Core Server uses POSIX threads to service concurrent requests. The Core Server accepts requests from any authenticated client; however, certain Core Server functions can be performed only by trusted clients. Trusted clients are those for whom control permission has been set in the Core Server's ACL entry for the client. Higher levels of trust are granted to clients who have both control and write permission set in their ACL entry. Refer to the *HPSS server security ACLs* section of the *HPSS Management Guide* for information concerning the ACL for the Core Server.

The Core Server can be configured to allow or disallow super-user privileges (root access). When the Core Server is configured to allow root access, the UID of the super-user is configurable.

HPSS systems configured with multiple subsystems employ multiple Core Servers and multiple metadata databases. Though the servers are separate, each Core Server in a given HPSS realm must share the fileset global metadata table.

Name space.

The HPSS Core Server maintains the HPSS name space in system metadata. Refer to *Section 3.5, "HPSS sizing considerations"* for details on sizing the name space. Refer to the *DB2 space shortage* section of the *HPSS Management Guide* for information on handling a metadata space shortage. By utilizing multiple storage subsystems, it is possible to distribute large name spaces across multiple Core Servers. Junctions between the names spaces of these storage subsystems can be used to "join" these subsystems.

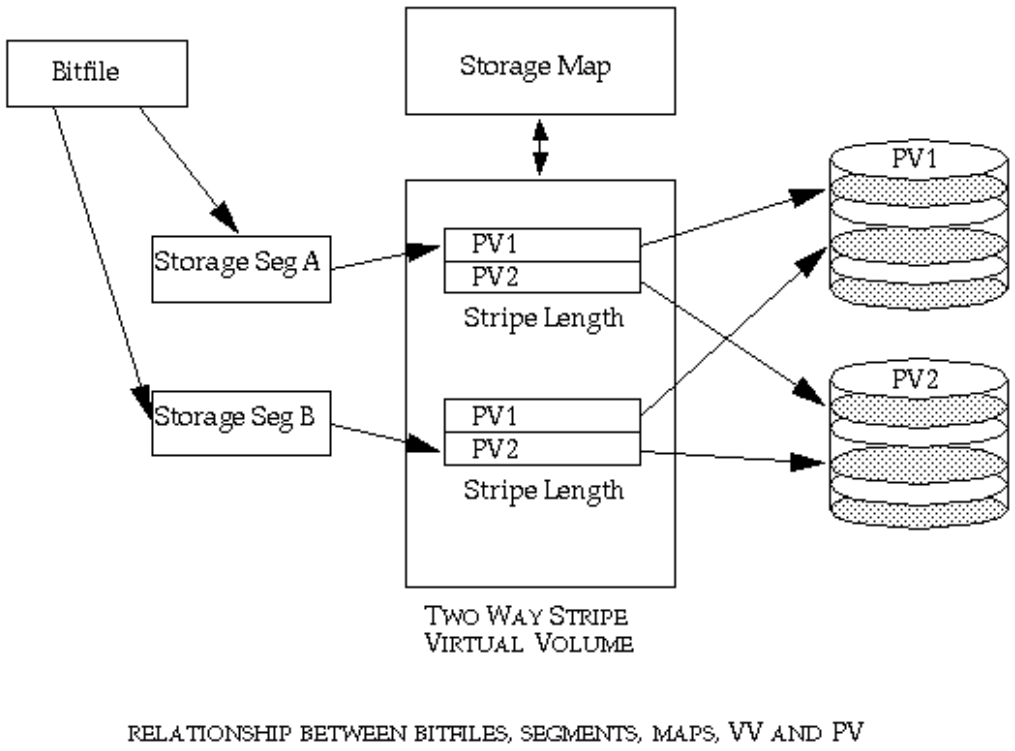
Bitfiles.

The Core Server provides a view of HPSS as a collection of files. It provides access to these files and maps the files onto underlying storage objects.

When a Core Server is configured, it is assigned a server ID. This value should never be changed because it is embedded in the ID of each bitfile and storage segments it uses. HPSS expects to be able to extract the server ID from any bitfile ID and connect to that server to access the file.

The Core Server maps bitfiles to their underlying physical storage by maintaining information that maps a bitfile to the storage segments that contain its data. For additional information, see *Section 3.7.1, "Core Server"*. The relationship of bitfiles to storage segments and other structures is shown in *Figure 3.5, "The relationship of various server data structures"*.

Figure 3.5. The relationship of various server data structures



Disk storage allocation.

Each Core Server manages disk storage units for HPSS. It maps each disk storage unit onto an HPSS disk Virtual Volume (VV) and records configuration data for the VV. The server also maintains a storage map for each VV that describes which portions of the VV are in use and which are free. *Figure 3.5, "The relationship of various server data structures"* shows the relationship of Core Server data structures such as VVs to other server data structures.

When one disk unit is mapped to a VV, the disk is said to be in a "1-wide stripe". Files are written to the disk in the conventional fashion. When two or more disks are grouped together in a VV, the disks are said to be in an "n-wide stripe". Files written to the VV are striped across the disks. All disks are written simultaneously with different portions of the files. Striping data this way increases throughput to the disks by almost a factor of N, where N is the width of the stripe.

The server can manage information for any number of disk VVs; however, because a copy of all of the VV and storage map information is kept in memory while the server runs, the size of the server will be somewhat proportional to the number of disks it manages.

The Core Server is designed to scale up its ability to manage disks as the number of disks increase. As long as sufficient memory and CPU capacity exist, threads can be added to the server to increase its throughput. Additional subsystems can also be added to a system, increasing concurrency even further.

Tape storage allocation.

Each Core Server manages magnetic tape storage media for HPSS. The server maps each tape cartridge onto an HPSS tape Physical Volume (PV) and records configuration data for the PV. Groups of one or more PVs are managed by the server as tape Virtual Volumes (VVs). The server maintains

a storage map for each VV that describes how much of each tape VV has been written and which storage segment, if any, is currently writable at the end of the VV. *Figure 3.5, "The relationship of various server data structures"* shows the relationship of data structures such as VVs to other server data structures.

When one tape cartridge is mapped to a VV, the tape is said to be in a "1-wide stripe". Files are written to the tape serially in the conventional fashion. When two or more tapes are grouped together in a VV, the tapes are said to be in an "n-wide stripe". Files written to the VV are striped across the tapes. All tapes are written simultaneously with different portions of the files. Striping data this way increases throughput to the tapes by almost a factor of N, where N is the width of the stripe.

RAIT tape VVs are special cases of striped tape VVs. In a RAIT VV, one or more of the PVs is used to store parity information for each stripe of data that is written. This allows the system to recover data from the RAIT VV if some number of the individual tapes become unreadable. It should be understood that the parity information rotates around the tapes, stripe by stripe, in roughly RAID 5 or 6 fashion, so it is not strictly correct to think of one of the PVs as the "parity tape".

The server can manage information for any number of tape VVs. It can also manage an unlimited number of tape VVs, maps, and storage segments without impacting its memory size.

The Core Server is designed to scale up its ability to manage tapes as the number of tapes increases. As long as sufficient memory and CPU capacity exist, threads can be added to the server to increase its throughput. Additional subsystems can also be added to a system, increasing concurrency even further.

Note that the number of tape drives the server manages has much more to do with the throughput of the server than the number of tape volumes the server manages. If the number of tape drives in the system needs to increase to meet workload demands, adding a new subsystem and redistributing the drives may be the best way to deal with the increased workload.

3.7.2. Migration/Purge Server

The Migration/Purge Server (MPS) reports storage class usage statistics and manages the amount of free space available in storage classes by performing periodic migration and purge runs on the storage classes. Migration copies data from the storage class on which it runs to one or more lower levels in the storage hierarchy. Once data has been migrated, a subsequent purge run will delete the data from the migrated storage class, if so configured. Migration is a prerequisite for purge, and MPS will never purge data which has not previously been migrated. It is possible, but not desirable, to assign only a migration policy and no purge policy to a disk storage class; however, this will result in data being copied (migrated) but never deleted (purged). It is important to recognize that migration and purge policies determine when data is migrated from a storage class and when the data is purged from that storage class; however, the number of copies and the location of those copies is determined by the storage hierarchy definition. The MPS uses the Core Server both to perform data movement between hierarchy levels and gather statistics. As such, the Core Server must be running in order for the MPS to complete its duties.

The MPS can exist only within a storage subsystem and a subsystem may be configured with no more than one MPS. Storage hierarchies are global across all storage subsystems within an HPSS system, but a given hierarchy may or may not be enabled within a given subsystem (a hierarchy is enabled within a subsystem by configuring the subsystem to enable one or more classes of service which reference that hierarchy). Note that a storage class may not be selectively migrated and purged

in different subsystems. If the hierarchy contains storage classes which require migration and purge, then an MPS must be configured to run against those storage classes in the subsystem. This MPS will manage migration and purge operations on only those storage resources within its assigned subsystem. Thus, for an HPSS system with multiple storage subsystems, there may be multiple MPSs, each operating on the resources within a particular subsystem.

Migration and purge operate differently on disk and tape storage classes. Disk migration and disk purge are configured on a disk storage class by associating a migration policy and a purge policy with that storage class. For tape storage classes, the migration and purge operations are combined and are collectively referred to as tape migration. Tape migration is enabled by associating a migration policy with a tape storage class. Purge policies are not needed or supported on tape storage classes.

Once migration and purge policies are configured for a storage class (and the MPS is restarted), the MPS will begin scheduling migration and purge runs for that storage class. Migration on both disk and tape is run periodically according to the runtime interval configured in the migration policy. Disk purge runs are not scheduled periodically, but rather are started when the percentage of space used in the storage class reaches the threshold configured in the purge policy for that storage class. It is critical that the hierarchies to which a storage class belongs be configured with proper migration targets in order for migration and purge to perform as expected.

The purpose of disk purge is to maintain a given amount of free space in a disk storage class by removing data of which copies exist at lower levels in the hierarchy. The order in which purge records are sorted, which determines the order in which files are purged, may be configured on the purge policy. It should be noted that all of the options except *Record Create Time* require additional metadata updates and can impose extra overhead on DB2. Also, unpredictable purge behavior may be observed if the purge record ordering is changed with existing purge records in the system until these existing records are cleared. A purge run ends when either the supply of purge records is exhausted or the purge target is reached.

Tape file migration is a file-based tape method which is able to make a single copy of tape files to the immediately lower level in the hierarchy. For details on tape migration options, see *Section 3.9.1, "Migration policy"*. If the MPS detects that the source tape volume has become active at any point during a tape file migration run, migration is abandoned on this volume until the next migration run. This is done in order to avoid competing with an HPSS system user for this volume. A tape volume is deemed to be active if any file it contains has been read or written within the access intervals specified in the migration policy.

The MPS provides the capability of generating migration/purge report files that document the activities of the server. The specification of the UNIX report file name prefix in the MPS server-specific configuration enables the server to create these report files. It is suggested that a complete path be provided as part of this file name prefix. Once reporting is enabled, a new report file is started every 24 hours. The names of the report files are made up of the UNIX file name prefix from the server-specific configuration, plus a year-month-day suffix. With reporting enabled, MPS will generate file-level migration and purge report entries in real time. These report files can be interpreted and viewed using the **mps_reporter** utility. Since the number and size of the report files grow rapidly, each site should develop a **cron** job that will periodically remove the reports that are no longer needed.

In order to efficiently perform disk migration, the MPS parallelizes the migration of files from disk to tape. The number of bytes that the MPS migrates simultaneously is user configurable via the Migration Stream Count in the *Disk Migration Policy* screen. For example, if the Migration Stream

Count is set to one, then the MPS will serially migrate aggregates. If the Migration Stream Count is set to four, then the MPS will attempt to have four aggregates migrating to four different virtual volumes at one time.

Tape aggregation, the default behavior for the disk migration policy, is the process of aggregating (that is, bundling together) many files on tape. This can dramatically increase migration performance, especially for sites that tend to ingest many relatively small files. When an individual file is written to tape, its data is flushed before the next file is written. When files are aggregated on tape, the time to flush the data to tape is only incurred once for the entire aggregate.

The Ordered Migration feature orders files being migrated from disk to tape by directory or by creation time. Tape aggregates can be used to group files according to their directories. This is especially pertinent when considering how to employ the Full Aggregate Recall feature. The objective of Ordered Migration is to co-locate data on tape, with either the directory of the files or the files' create times being the co-location criteria. A site may want to combine this feature with Full Aggregate Recall to potentially minimize the number of tape operations required to bring a directory's files back from tape. Of course, the migration policy will continue to honor the migration policy settings for when a file is eligible for migration and how often migration will run. Refer to the *HPSS Management Guide* **Classes of Service** and **Migration Policies** sections for further details on this feature.

As previously indicated, the MPS provides the information displayed in the *HPSS Active Storage Classes* window in SSM. Each MPS contributes storage class usage information for the resources within its storage subsystem. MPS accomplishes this by polling the Core Server within its subsystem at the interval specified in the MPS server-specific configuration. The resulting output is one line for each storage class for each storage subsystem in which that class is enabled. The MPS for a subsystem does not report on storage classes which are not enabled within that subsystem. The warning and critical storage class thresholds are also activated by the MPS.

3.7.3. Gatekeeper

Each Gatekeeper may provide sites with the ability to:

- Monitor or control the use of HPSS resources using Gatekeeping Services.
- Validate user accounts using the Account Validation Service.

If the site doesn't want either service, then it is not necessary to configure a Gatekeeper into the HPSS system.

Sites can choose to configure zero (0) or more Gatekeepers per HPSS system. Gatekeepers are associated with storage subsystems. Each storage subsystem can have zero or one Gatekeeper associated with it and each Gatekeeper can support one or more storage subsystems. Gatekeepers are associated with storage subsystems using the *Storage Subsystem Configuration* screen (see the *Storage subsystems* section of the *HPSS Management Guide*). If a storage subsystem has no Gatekeeper, then the Gatekeeper field will be blank. A single Gatekeeper can be associated with every storage subsystem, a group of storage subsystems, or one storage subsystem. A storage subsystem can *not* use more than one Gatekeeper.

Every Gatekeeper has the ability to supply the Account Validation Services. A bypass flag in the accounting policy metadata indicates whether or not account validation for an HPSS system is on or

off. Each Gatekeeper will read the accounting policy metadata file, so if multiple Gatekeepers are configured and account validation has been turned on, then any Gatekeeper can be chosen by the Location Server to fulfill account validation requests.

Every Gatekeeper has the ability to supply the Gatekeeping Service. The Gatekeeping Service provides a mechanism for HPSS to communicate information through a well-defined interface to a policy software module to be completely written by the site. The site policy code is placed in a well-defined site shared library for the gatekeeping policy (either `/usr/local/lib64/libgksite.[a|so]` or `/opt/hpss/lib/libgksite.[a|so]`) which is linked to the Gatekeeper. The gatekeeping policy shared library contains a default policy which does *no* gatekeeping. Sites will need to enhance this library to implement local policy rules if they wish to monitor or load-balance requests.

The gatekeeping site policy code will determine which types of requests it wants to monitor (authorized caller, create, open, and stage). Upon initialization, each Core Server will look for a Gatekeeper in the storage subsystem metadata. If no Gatekeeper is configured for a particular storage subsystem, then the Core Server in that storage subsystem will not attempt to connect to any Gatekeeper. If a Gatekeeper is configured for the storage subsystem that the Core Server is configured for, then the Core Server will query the Gatekeeper asking for the monitor types by calling a particular Gatekeeping Service API. This API will then call the appropriate Site Interface which each site can provide to determine which types of requests are to be monitored. This query by the Core Server will occur each time the Core Server (re)connects to the Gatekeeper. The Core Server will need to (re)connect to the Gatekeeper whenever the Core Server or Gatekeeper is restarted. Thus if a site wants to change the types of requests it is monitoring, then it will need to restart the Gatekeeper and Core Server.

If a Gatekeeper is configured, then it will either need to be running or marked non-executable for HPSS Client API requests to succeed in the Core Server (even if no gatekeeping or account validation is occurring); this is due to the HPSS Client API performing internal accounting initialization.

If multiple Gatekeepers are configured for gatekeeping, then the Core Server that controls the files being monitored will contact the Gatekeeper that is located in the same storage subsystem. Conversely, if one Gatekeeper is configured for gatekeeping for all storage subsystems, then each Core Server will contact the same Gatekeeper.

A Gatekeeper registers five different interfaces: Gatekeeper Service (also know as the *Functional Interface*), Account Validation Services, Administrative Services, Connection Manager Services, and Real-Time Monitoring Services. When the Gatekeeper initializes, it registers each separate interface. The Gatekeeper-specific configuration will contain any pertinent data about each interface.

The Gatekeeper Service interface provides the Gatekeeping APIs which call the site-implemented Site Interfaces. The Account Validation Service interface provides the Account Validation APIs. The Administrative Service provides the server APIs used by SSM for viewing, monitoring, and setting server attributes. The Connection Manager Service provides the HPSS connection management interfaces. The Real-Time Monitoring Service interface provides the Real-Time Monitoring APIs.

The Gatekeeper Service Site Interfaces provide a site the mechanism to create local policy on how to throttle or deny create, open and stage requests and which of these request types to monitor. For example, it might limit the number of files a user has opened at one time; or it might deny all create requests from a particular host or user. The Site Interfaces will be located in a shared library that is linked into the Gatekeeper.

It is important that the Site Interfaces return a status in a timely fashion. Create, open, and stage requests from MPS are timing sensitive, so the Site Interfaces won't be permitted to delay or deny these requests; however, the Site Interfaces may choose to be involved in keeping statistics on these requests by monitoring requests from authorized callers.

If a Gatekeeper should become heavily loaded, additional Gatekeepers can be configured (maximum of one Gatekeeper per storage subsystem). In order to keep the Gatekeepers simple and fast, they do not share state information. Thus if a site wrote a policy to allow each host a maximum of 20 creates, then that host would be allowed to create 20 files on each storage subsystem that has a separate Gatekeeper.

The Gatekeeper's Real-Time Monitoring Interface supports clients such as a Real-Time Monitoring utility which requests information about particular user files or HPSS Request IDs.

3.7.4. Location Server

All HPSS client API applications, which includes all end-user applications, will need to contact the Location Server at least once during initialization and usually later during execution in order to locate the appropriate servers to contact. If the Location Server is down for an extended length of time, these applications will eventually give up retrying their requests and become non-operational. Consider increasing the automatic restart count for failed servers in SSM.

If any server is down for an extended length of time it is important to mark the server as non-executable within SSM. As long as a server is marked executable the Location Server continues to advertise its location to clients which may try to contact it.

The Location Server must be reinitialized or recycled whenever the location policy or its server configuration is modified. Note that it is not necessary to recycle the Location Server if an HPSS server's configuration is added, modified, or removed since this information is periodically reread.

3.7.5. PVL

The PVL is responsible for mounting and dismounting PVs (such as tape and magnetic disk) and queuing mount requests when required drives and media are in use. The PVL usually receives requests from Core Server clients. The PVL accomplishes any physical movement of media that might be necessary by making requests to the appropriate Physical Volume Repository (PVR). The PVL communicates directly with HPSS Movers in order to verify media labels. The PVL will manage the creation, deletion, and update of drives/devices within the HPSS system.

Only one PVL per HPSS system is supported.

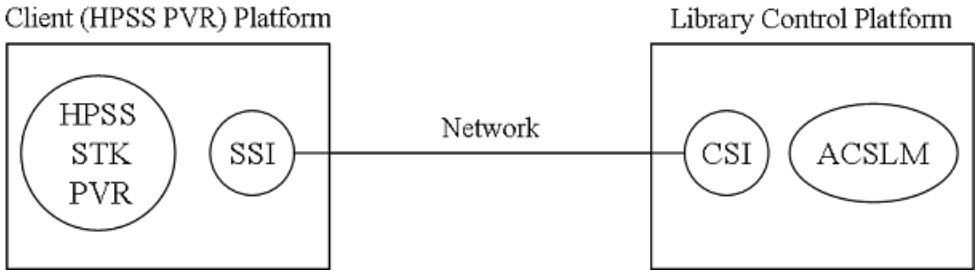
3.7.6. PVR

The PVR manages a set of imported cartridges and mounts and dismounts them when requested by the PVL. It is possible for multiple HPSS PVRs to manage a single library. This is done if it is necessary to organize the tape drives in the library into partitions. Each tape drive in the library is assigned to exactly one PVR. Additionally, each cartridge is assigned to only one PVR. The PVRs can be configured identically and can communicate with the library through the same interface.

The following sections describe the considerations for the various types of PVRs supported by HPSS.

3.7.6.1. STK PVR

The STK PVR communicates to the Oracle StorageTek ACSLS server via its Storage Server Interface (SSI) client software. The client software, maintained by Oracle Corp. can be obtained through HPSS support; see below and the *STK PVR additional information* section of the *HPSS Management Guide* for additional information. The SSI client software must be running on the same machine as the HPSS STK PVR. The SSI client communicates with the Client System Interface (CSI) server via RPCs.



The SSI must be started before the PVR. If the SSI is started after the PVR, the PVR should be stopped and restarted.

If multiple Oracle StorageTek libraries are managed, SSIs that communicate with each of the library units should be configured on separate CPUs. A PVR can be configured on each of the CPUs that is running an SSI. If multiple Oracle StorageTek robots are connected and are controlled by a single Library Management Unit (LMU), a single PVR can manage the collection of robots. The PVR can be configured on any CPU that is running an SSI.

ACSLS should be running on the platform directly connected to the Oracle StorageTek library. The HPSS STK PVR can run on any platform that has a TCP/IP connection to the ACSLS platform. The platform running the HPSS STK PVR must also be running Oracle StorageTek's SSI client software. This software will not be started by HPSS and should be running when HPSS is started. It is recommended that the SSI be started by the platform's initialization scripts every time the platform is booted.

The SSI requires that the system environment variables CSI_HOSTNAME and ACSAPI_PACKET_VERSION be correctly set. Note that due to limitations in the Oracle StorageTek Client System Component Developer's Toolkit, if the SSI is not running when the HPSS STK PVR is started, or if the SSI crashes while the HPSS STK PVR is running, the HPSS STK PVR will lock up and will have to be manually terminated by issuing `"kill -9 <pid>"`.

3.7.6.2. AML PVR

The AML PVR is supported by special bid only.

The AML PVR can manage ADIC AML robots that use Distributed AML Server (DAS) software. The DAS AML Client Interface (ACI) operates synchronously; that is, once a request is made to the AML, the request process does not regain control until the operation has completed or terminated. Therefore, the AML PVR must create a process for each service request sent to the DAS (such as mount, dismount, or eject a tape).

HPSS is designed to work with ADIC Distributed Automated Media Library Server (DAS) software version 1.3 and the ABBA Management Unit (AMU) version 2.4.0. DAS is the ADIC software which consists of the Automated Media Library (AML) Client Interface (ACI) and the DAS server

components. The AMU is the host computer software by which the ADIC Storage System manages the archive database, which is based on a DB2 compatible database for an OS/2 system.

The AMU must run on an OS/2 PC host computer connected to the AML robot while the HPSS AML PVR can run on any RS/6000 workstation that has a TCP/IP connection to the OS/2 host computer. The workstation running the HPSS AML PVR must also contain the DAS/ACI software that is called by the HPSS AML PVR.

Refer to the *ADIC DAS Installation and Administration Guide* and *Reference Guide AMU* for additional information.

3.7.6.3. Operator PVR

The Operator PVR displays mount requests for manually mounted drives. The mount requests are displayed on the appropriate SSM screen.

All of the drives in a single Operator PVR must be of the same type. Multiple operator PVRs can be configured without any additional considerations.

3.7.6.4. SCSI PVR

The SCSI PVR communicates with tape libraries and robots through a generic SCSI interface. The interface uses the SCSI-3 command set.

3.7.7. Mover

The Mover configuration is largely dictated by the hardware configuration of the HPSS system. Each Mover can handle both disk and tape devices and must run on the node to which the storage devices are attached. The Mover is also capable of supporting multiple data transfer mechanisms (such as TCP/IP and shared memory) for sending data to or receiving data from HPSS clients.

3.7.7.1. Tape devices

All tape devices that will be used to read and write HPSS user data must be set to handle variable block sizes to allow for the ANSI standard 80-byte volume label and file section headers. This section describes the procedure for setting this option on each supported operating system.

HPSS supports tape devices on Linux with the use of the native SCSI tape device driver (**st**). To enable the loading of the Linux native tape device, uncomment the following lines in the `.config` file and follow the procedure for rebuilding your Linux kernel.

```
CONFIG_SCSI=y
CONFIG_CHR_DEV_ST=y
```

In Linux, tape device files are dynamically mapped to SCSI IDs/LUNs on your SCSI bus. The mapping allocates devices consecutively for each LUN of each device on each SCSI bus found at the time of the SCSI scan, beginning at the lower LUNs/IDs/buses. The tape device file will be in this format: `/dev/st[0-31]`. This will be the device name to use when configuring the HPSS device.

3.7.7.2. Disk devices

All locally attached magnetic disk devices (for example, SCSI or SSA) should be configured using the pathname of the raw device (that is, character special file).

For Linux systems, this may involve special consideration.

HPSS supports disk devices on Linux with the use of the native SCSI disk device driver (**sd**) and the raw device driver (**raw**).

The Linux SCSI Disk Driver presents disk devices to the user as device files with the following naming convention: `/dev/sd[a-h][0-8]`. The first variable is a letter denoting the physical drive, and the second is a number denoting the partition on that physical drive. Occasionally, the partition number will be left off when the device corresponds to the whole drive. Drives can be partitioned using the Linux **fdisk** utility.

The Linux raw device driver is used to bind a Linux raw character device to a block device. Any block device may be used.

See the Linux manual page for more information on the SCSI Disk Driver, the Raw Device Driver, and the **fdisk** utility.

To enable the loading of the Linux native SCSI disk device, uncomment the following lines in the configuration file and follow the procedure for rebuilding your Linux kernel.

```
CONFIG_SCSI=y
CONFIG_BLK_DEV_SD=y
```

Also, depending on the type of SCSI host bus adapter (HBA) that will be used, you will need to enable one or more of the lower level SCSI drivers. For example, if you are using one of the Adaptec HBAs with a 7000 series chipset, uncomment the following lines in the `.config` file and follow the procedure for rebuilding your Linux kernel.

```
CONFIG_SCSI_AIC7XXX=y
CONFIG_AIC7XXX_CMDS_PER_DEVICE=253
CONFIG_AIC7XXX_RESET_DELAY_MS=15000
```

3.7.7.3. Performance

The configuration of the Movers and attached devices can have a large impact on the performance of HPSS because of constraints imposed by a number of factors; for example, device channel bandwidth, network bandwidth, and processor power.

A number of conditions can influence the number of Movers configured and the specific configuration of those Movers:

- Each Mover process is built to handle a specific device interface, for example, IBM SCSI-attached 3590/3590H/3580 drives. If multiple types of devices are to be supported, multiple Movers must be configured.
- Each Mover currently limits the number of concurrently outstanding connections. If a large number of concurrent requests are anticipated on the drives planned for a single Mover, the device workload should be split across multiple Movers. This is primarily an issue for Movers that will support disk devices.
- The planned device allocation should be examined to verify that the device allocated to a single node will not overload that node's resource to the point that the full transfer rates of the device cannot be achieved (based on the anticipated storage system usage). To offload a single node, some number of the devices and their corresponding Mover can be allocated to other nodes.

- In general, the connectivity between the nodes on which the Movers will run and the nodes on which the clients will run should have an impact on the planned Mover configuration. For TCP/IP data transfers, the only functional requirement is that routes exist between the clients and Movers; however, the existing routes and network types will be important to the performance of client I/O operations.
- Mover to Mover data transfers, performed for migration, staging, and repack operations, also impact the Mover configuration. For devices that support storage classes involved in migration or staging, the Movers controlling those devices should be configured so that there is an efficient data path among them. If Movers involved in a data transfer are configured on the same node, the transfer will occur via a shared memory segment.

3.7.8. Logging service

Logging Services are included in HPSS servers and tools.

HPSS servers and tools send log messages based on their log policies to syslog on the same node as the process. Based on the policy, messages may also be forwarded to the SSM for display in the Alarm and Events list.

3.7.9. Startup Daemon

The Startup Daemon is responsible for starting, monitoring, and stopping the HPSS servers. The daemon responds only to requests from the SSM System Manager. It shares responsibility with each HPSS server for ensuring that only one copy of the server runs at a given time. It helps the SSM determine whether servers are still running, and it allows the SSM to send signals to servers. Normally, the SSM stops servers by communicating directly with them but, in special cases, the SSM can instruct the Startup Daemon to send a **SIGKILL** signal to cause the server to shut down immediately.

If a server is configured to be restarted automatically, the Startup Daemon will restart the server when it terminates abnormally. The daemon can be configured to restart the server without limit, or up to a fixed number of restarts, or not at all.

Choose a descriptive name for the daemon that includes the name of the computer where the daemon will be running. For example, if the daemon will be running on a computer named tardis, use the descriptive name "Startup Daemon (**tardis**)".

The Startup Daemon is started by running the `/opt/hpss/bin/rc.hpss` script. It ignores most signals and may only be killed using the **kill -9 <pid>** command. The Startup Daemon must be run under the *root* account so that it has sufficient privileges to start the HPSS servers.

The Startup Daemon runs on every HPSS server node. However, it does not run on remote Mover nodes.

3.7.10. Storage System Management

SSM has three components:

- A System Manager (**hpss_ssmsm**), which communicates with all other HPSS components requiring monitoring or control.

- The GUI (**hpssgui**), which provides the HPSS administrator or operator the ability to configure or monitor the HPSS system through a set of windows.
- A Command Line Interface (**hpssadm**), which provides the HPSS administrator or operator the ability to configure or monitor a subset of the HPSS system through a set of interactive or batch commands.

There can be only one SSM System Manager configured for an HPSS installation. The System Manager (SM) is able to handle multiple SSM GUI or Command Line clients (on different hosts or on the same host).

Starting up the SSM GUI (**hpssgui**) directly from the HPSS server node where the SSM System Manager is running and displaying the SSM window on the user's desktop is discouraged. This is due to known Java/X performance problems. Instead, it is recommended to install the Java and HPSS GUI client software on the user's desktop and execute it there. See the *SSM desktop client packaging* section of the *HPSS Management Guide* for more information.

There are no performance problems associated with running the SSM Command Line Interface (**hpssadm**) directly on the server UNIX platform, and this is the recommended configuration.

Both the SSM GUI client, **hpssgui**, and the SSM Command Line client, **hpssadm**, may be executed on any platform that complies with *the section called "SSM client requirements"*. Before starting the SM, a review of SM key environment variable settings would be wise. Following is a table of key SM environment variables along with the default value and meaning. Depending on the size (number of servers and number of SSM clients) and activity of the HPSS system, these values may need to be overridden in `env.conf`.

Table 3.8. Key SM environment variables

Variable	Default value	Functionality
HPSS_SM_SRV_ CONNECT_FAIL_COUNT	3	Connection fail count. The number of connection failures to a server before the maximum connection interval takes effect.
HPSS_SM_SRV_ CONNECT_INTERVAL_MIN	20	Interval between attempting server connections when the connection fail count has not yet been reached (in seconds).
HPSS_SM_SRV_ CONNECT_INTERVAL_MAX	60	Maximum connection interval. The interval between server connections when the connection fail count has been reached without a successful connection (in seconds).
HPSS_SM_SRV_ MONITOR_THREADS	5	The number of threads created to monitor server connections.
HPSS_SM_SRV_ QUEUE_SIZE	5	Request queue size used by the System Manager server interface. A default of five slots in the server interface request queue to be used when the server interface thread pool is completely full. The queue is used to hold RPC requests from servers until a thread is available to process the request.

Variable	Default value	Functionality
		Note that if the request queue has any entries in it, it means that all the threads in the server thread pool are busy and the SM response will be degraded. If this happens, then it would be good to increase the number of threads available to the server interface using the HPSS_SM_SRV_TPOOL_SIZE variable. Increasing the size of the queue will not help with performance.
HPSS_SM_SRV_ TPOOL_SIZE	100	Thread pool size used by the System Manager server interface. If the thread pool is exhausted, then server RPC requests will be queued in the server RPC request queue to wait for a thread to become available. When the thread pool is exhausted, SM performance may be degraded. Increase this value if that is the case. Typically, one thread per HPSS server should be adequate, but a few extra wouldn't hurt.
HPSS_SM_SRV_ MAX_CONNECTIONS	50	Number of HPSS server connections to maintain at once. If this number of connections is exceeded, then old connections will be closed to maintain this number of connections.

The SM attempts to throttle the connection attempts to other servers. It will attempt to reconnect to each server every HPSS_SM_SRV_CONNECT_INTERVAL_MIN seconds until the number of failures for that server has reached HPSS_SM_SRV_CONNECT_FAIL_COUNT. After the failure count has been reached the SM will only try to reconnect to the server every HPSS_SM_SRV_CONNECT_INTERVAL_MAX seconds until a successful connection is made at which time the connection interval for the server will be set back to HPSS_SM_SRV_CONNECT_INTERVAL_MIN.

3.8. Storage subsystem considerations

Storage subsystems are provided in HPSS for the purpose of increasing the scalability of the system, particularly with respect to the Core Servers. An HPSS system consists of one or more subsystems, and each subsystem contains its own Core Server. If multiple Core Servers are desired, this is accomplished by configuring multiple subsystems.

Each subsystem uses a separate DB2 subsystem database. Adding a subsystem to an HPSS system means adding an additional database that must be maintained and backed up. All subsystems share the config database.

3.9. Storage policy considerations

This section describes the various policies that control the operation of the HPSS system.

3.9.1. Migration policy

The migration policy provides the capability for HPSS to migrate (copy) data from one level in a hierarchy to one or more lower levels. The migration policy defines the amount of data and the conditions under which it is migrated; however, the number of copies and the location of those copies is determined by the storage hierarchy definition. The site administrator will need to monitor the usage of the storage classes being migrated and adjust both the migration and purge policies to obtain the desired results.

3.9.1.1. Migration policy for disk

Disk migration in HPSS copies files from a disk storage class to one or more lower levels in the storage hierarchy. Removing or purging of the files from the disk storage class is controlled by the purge policy. The migration and purge policies work in conjunction to maintain sufficient storage space in the disk storage class.

When data is copied from the disk, the copied data will be marked purgeable but will not be deleted. Data is deleted by running purge on the storage class. If duplicate copies are created, the copied data is not marked purgeable until all copies have been successfully created. The migration policy and purge policy associated with a disk storage class must be set up to provide sufficient free space to deal with the demand for storage. This involves setting the parameters in the migration policy to migrate a sufficient number of files and setting the purge policy to reclaim enough of this disk space to provide the free space desired for users of the disk storage class.

Disk migration is controlled by several parameters. By default, these parameters are the same across all subsystems. However, subsystem-specific policies may be created which override all of these values. For a list of these parameters, refer to the *Disk Migration Policy configuration* section in the *HPSS Management Guide*.

3.9.1.2. Migration policy for tape

The tape file migration feature allows a Class of Service to be configured so that files written to a tape storage class will be copied to another tape storage class and retained in both. This provides a backup copy of the files on tape.

The migration policy parameters which apply to tape file migration are detailed in the *Tape Migration Policy configuration* section in the *HPSS Management Guide*.

3.9.2. Purge policy

The purge policy allows the MPS to remove the bitfiles from disk after the bitfiles have been migrated to a lower level of storage in the hierarchy. A purge policy should not be defined for a tape storage class or a disk storage class which does not support migration. Sites may or may not wish to define a purge policy for all disk storage classes that support migration. Purging from tapes is controlled by the "Migrate Files and Purge" flag of the tape migration policy; there is no separate purge policy for tape storage classes.

The specification of the purge policy in the storage class configuration enables the MPS to do the disk purging according to the purge policy for that particular storage class. Purge is run for a storage class

on a demand basis. The MPS maintains current information on total space and free space in a storage class by periodically extracting this information from the HPSS Core Server. Based upon parameters in the purge policy, a purge run will be started when appropriate. The administrator can also force the start of a purge run via SSM.

The disk purge is controlled by several parameters:

- The **Do not purge files accessed within <nnn> minutes** parameter determines the minimum amount of time a site wants to keep a file on disk. Files that have been accessed within this time interval are not candidates for purge.
- The **Start purge when space used reaches <nnn> percent** parameter allows the amount of free space that is maintained in a disk storage class to be controlled. A purge run will be started for this storage class when the total space used in this class exceeds this value.
- The **Stop purge when space used falls to <nnn> percent** parameter allows the amount of free space that is maintained in a disk storage class to be controlled. The purge run will attempt to create this amount of free space. Once this target is reached, the purge run will end.
- The **Purge Locks expire after <nnn> minutes** parameter allows the length of time a file can be "purge locked" before it will appear on the MPS report to be controlled. The "purge lock" is used to prevent a file from being purged from the highest level of a hierarchy. Purge locks only apply to a hierarchy containing a disk on the highest level. HPSS will not automatically unlock locked files after they expire. HPSS reports the fact that they have expired in the MPS report.
- The **Purge by** list box allows sites to choose the criteria used in selecting files for purge. By default, files are selected for purge based on their migration time. Alternately, the selection of files for purging may be based on the time the file was created or the time the file was last accessed. Files may be purged in an unpredictable order if this parameter is changed while there are existing purge records already in metadata until those existing files are processed.

Purging is also controlled by the presence of super purge locks. These should be rare. The super purge lock prevents any level of the file from being purged except manually by force. It is set temporarily by the **recover** utility and released when the **recover** utility determines that a safe copy of the file still exists. It is set internally by the core server whenever a manual force purge is performed on a file, which should not be done except with guidance from HPSS support. Super purge locks do not expire. They may be removed manually by the **scrub purgeunlock** command with the assistance of HPSS support. The MPS and the **plu** utility have not yet been updated to report super purge locks.

Administrators should perform performance tuning to determine the parameter settings that will fit the needs of their site. If a site has a large amount of disk file write activity, the administrator may want to have more free space and more frequent purge runs. However, if a site has a large amount of file read activity, the administrator may want to have smaller disk free space and less frequent purge runs, and allow files to stay on disk for a longer time.

3.9.3. Accounting policy and validation

The purpose of the accounting policy is to describe how a site will charge for storage, and, in addition, to describe the level of user authorization (validation) to be performed when maintaining accounting information. The policy is required for HPSS to run, even if the site doesn’t charge for HPSS usage. The accounting information is extremely useful for understanding usage patterns and planning site

layout and configuration. Even if sites do not charge for HPSS usage, the accounting information should be examined regularly. This information can assist sites in adjusting their system configuration to better handle their workload.

A site must decide which style of accounting to use before creating any HPSS files or directories. There are two styles of accounting: UNIX-style accounting and Site-style accounting. In addition, a site may decide to customize their style of accounting by writing an accounting site policy module for the Gatekeeper.

If a site chooses Site-style accounting and account validation is turned off, LDAP must be used as the authorization mechanism. The hpssGECOS field which is maintained for each user in LDAP contains the account index which allows Site-style accounting to be used. However, if account validation is turned on, then the account index comes from the account validation metadata (through the Gatekeeper).

If UNIX authorization is used and account validation is turned off, UNIX-style accounting must be used because there is no hpssGECOS field. The basic limitation is that if the account index is needed out of the hpssGECOS field, it does not exist in UNIX. It only exists in LDAP.

The metadata for each file and directory in an HPSS system contains an Account field, which determines how the storage will be charged. Each user has at least one default account index, which is put into the Account field of all new files and directories.

When using UNIX-style accounting, the account index is the user's UID. When the user's UID is combined with the user's realm ID, a unique account is created.

When using Site-style accounting, each user may have more than one account index and may switch among them at runtime.

Each site must decide whether they wish to validate accounts. However, when using UNIX-style accounting no authorization checking need be done since the account is always the user's UID.

If account validation is enabled, additional authorization checks are performed when the following events occur: when files and directories are created, when their ownership is changed, when their account index is changed, or when a user attempts to use an account index other than their default. If the authorization check fails, the operation fails with a permission error.

Using account validation is highly recommended for sites that will be accessing remote HPSS systems. The use of account validation will help keep account indexes consistent. If remote sites are not being accessed, account validation is still recommended as a mechanism to keep consistent accounting information.

For Site-style accounting, an account validation metadata file must be created, populated, and maintained with valid user account indexes. See the Account Validation Editor (**hpss_avaledit**) manual page for details on the use of the Account Validation Editor.

If the **Require Default Account** field is enabled when using Site-style accounting and account validation, users are required to have valid default account indexes before performing almost any client API action. If the **Require Default Account** field is disabled (which is the default behavior) users will only be required to have a valid account set when performing an operation which requires an account to be validated such as a create, an account change operation, or an ownership change operation.

When using Site-style accounting with account validation, if the **Account Inheritance** field is enabled, newly created files and directories will automatically inherit their account index from their parent directory. The account indexes can then be changed explicitly by users. This is useful when individual users have not had default accounts set up for them or if entire directory trees need to be charged to the same account. When **Account Inheritance** is disabled (which is the default) newly created files and directories will obtain their account from the user's current session account, which is initially set to the user's default account index. This default account index may be changed by the user during the session.

A site may decide to customize the way they do accounting. In most cases, these sites should enable account validation with Site-style accounting and then implement their own site policy module which will be linked with the Gatekeeper. See *Section 3.7.3, "Gatekeeper"* as well as the appropriate sections of the *HPSS Programmer's Reference* for more information.

By default account validation is disabled (bypassed). If it is disabled, the style of accounting is determined by looking up each user's hpssGECOS account information in the authorization registry. The following instructions describe how to set up users in this case.

If users have their default account index encoded in a string of the form **AA=<default-acct-idx>** in the principal's LDAP hpssGECOS attribute, then Site-style accounting will be used; otherwise, UNIX-style accounting will be used.

To keep the accounting information consistent, it is important to set up all users in the HPSS Authorization services with the same style of accounting (that is, they should all have the **AA=** string in their hpssGECOS attribute or none should have this string). The **hpss_ldap_admin** tool can be used to set attributes for a user including the hpssGECOS field. For more information, see the **hpss_ldap_admin** man page.

See the *Accounting* section of the *HPSS Management Guide* for more information.

3.9.4. Security policy

HPSS server authentication and authorization make extensive use of UNIX or Kerberos authentication and either UNIX or LDAP authorization mechanisms. Each HPSS server has configuration information that determines the type and level of services available to that server. HPSS software uses these services to determine the caller's identity and credentials. Server security configuration is discussed in more detail in the *Server configuration* section of the *HPSS Management Guide*.

Once the identity and credential information of a client has been obtained, HPSS servers enforce access to their interfaces based on permissions granted by an access control list stored in the DB2 table AUTHZACL.

HPSS client interface authentication and authorization security features for end users depend on the interface and are discussed in the following subsections.

3.9.4.1. Client API

The Client API interface uses either UNIX username/password or Kerberos authentication and either UNIX or LDAP authorization features. Applications that make direct Client API calls must have valid credentials prior to making those calls. Kerberos credentials can be obtained either at the command line level via the **kinit** mechanism or within the application via the **sec_login_set_context** interface.

UNIX credentials are determined by the HPSS rpc library based on the UNIX User ID and Group ID of the application process.

3.9.4.2. FTP/PFTP

By default, FTP and Parallel FTP (PFTP) interfaces use either a username/password mechanism or Kerberos credentials to authenticate. Either UNIX or LDAP is used to authorize end-users. The end user identity credentials are obtained from the principal and account records in the appropriate security registry.

3.9.4.3. Name space

Enforcement of access to HPSS name space objects is the responsibility of the Core Server. A user's access rights to a specific name space object are determined from the information contained in the object's ACL and the user's credentials.

3.9.4.4. Security audit

HPSS provides the ability to record information about authentication, file creation, deletion, access, and authorization events. The security audit policy in each HPSS server determines what audit records a server will generate. In general, all servers can create authentication events, but only the Core Server will generate file events. The security audit records are sent to HPSS logging services and recorded as security-type log messages.

3.9.5. Logging policy

The logging policy provides the capability to control which message types are written to the HPSS log files. In addition, the logging policy is used to control whether alarm and event messages are sent to the Storage System Manager to be displayed. Logging policy may be set on a per-server basis, or a default policy may be used for servers that don't have their own logging policy. Refer to the *Creating a log policy* section of the *HPSS Management Guide* for a description of the supported message types.

If a logging policy is not explicitly defined for a server, the default log policy will be applied. The default log policy is selected from the *Global Configuration* window. If no default log policy entry has been defined, only Alarm and Event messages will be logged. All Alarm and Event messages generated by the server will also be sent to the Storage System Manager, if enabled.

The administrator might consider changing a server's logging policy under one of the following circumstances:

- A particular server is generating excessive messages. Under this circumstance, the administrator could use the logging policy to limit the message types being logged or sent to the Storage System Manager. This will improve performance and potentially eliminate clutter from the HPSS *Alarms and Events* window. Message types to disable first would be Trace messages followed by Request messages.
- One or more servers are experiencing problems which require additional information to troubleshoot. If Alarm, Debug, or Request message types were previously disabled, enabling these message types will provide additional information to help diagnose the problem. HPSS support personnel might also request that Trace messages be enabled for logging.

3.9.6. Location policy

In past versions of HPSS, the location policy was used to provide the ability to control how often Location Servers in an HPSS installation contacted other servers. The location policy was used to determine how often remote Location Servers were contacted to exchange server location information.

This location policy information is still read by the Location Server, but since the 6.2 version of HPSS it has no practical value. It will probably be removed in future versions of HPSS.

3.9.7. Gatekeeping

The Gatekeeping Service provides a mechanism for HPSS to communicate information through a well-defined interface to an installation specific customized software policy module. The policy module is placed in a shared library, `/usr/local/lib64/libgksite.[a|so]` or `/opt/hpss/lib/libgksite.[a|so]`, which is linked into the Gatekeeper. A module in `/usr/local/lib64` will take precedence over one placed in `/opt/hpss/lib` and can be useful for maintaining a site library across HPSS releases and installations. The default policy module does no gatekeeping. If gatekeeping services are desired in an HPSS installation, this default policy module must be replaced with one that implements the desired policy.

The site implemented policy module determines which types of requests will be monitored (authorized caller, create, open, and stage). Upon initialization, each Core Server looks for a Gatekeeper configured in its storage subsystem. If one is found, the Core Server asks the Gatekeeper for its monitor types by calling the **gk_GetMonitorTypes** API which calls the site implemented **gk_site_GetMonitorTypes** function which determines which types of requests to monitor. This query by the Core Server occurs each time the Core Server connects to the Gatekeeper, which occurs whenever the Core Server or Gatekeeper is restarted. Therefore, if a site wants to change the types of requests to be monitored, the Core Server and Gatekeeper must be restarted.

For each type of request being monitored, the Core Server calls the appropriate Gatekeeping Service API (**gk_Create**, **gk_Open**, **gk_Stage**) passing along information pertaining to the request. This information includes:

Table 3.9. Gatekeeping call parameters

Name	Description	create	open	stage
AuthorizedCaller	Whether or not the request is from an authorized caller. These requests cannot be delayed or denied by the site policy.	Y	Y	Y
BitFileID	The unique identifier for the file.	N/A	Y	Y
ClientConnectId	The end client's connection UUID.	Y	Y	Y
RealmId	The HPSS realm identifier for the user.	Y	Y	Y

Name	Description	create	open	stage
GroupId	The user's group identifier.	Y	Y	Y
HostAddr	Socket information for originating host.	Y	Y	Y
OpenInfo	Open file status flag (Oflag).	N/A	Y	N/A
StageInfo	Information specific to stage (flags, length, offset, and storage level).	N/A	N/A	Y
UserId	The user's identifier.	Y	Y	Y

Each Gatekeeping Service API will then call the appropriate Site Interface passing along the information pertaining to the request. If the request had AuthorizedCaller set to `TRUE`, then the Site "Stat" Interface will be called (**gk_site_CreateStats**, **gk_site_OpenStats**, **gk_site_StageStats**) and the Site Interface will not be permitted to return any errors on these requests. Otherwise, if AuthorizedCaller is set to `FALSE`, then the normal Site Interface will be called (**gk_site_Create**, **gk_site_Open**, **gk_site_Stage**) and the Site Interface will be allowed to return no error or return an error to either retry the request later or deny the request. When the request is being completed or aborted, the appropriate Site Interface will be called (**gk_site_Close**, **gk_site_CreateComplete**, **gk_site_StageComplete**). Examples of when a request gets aborted are when the Core Server goes **DOWN** or when the user application is aborted.

NOTES:

1. All open requests to the Core Server will call the Gatekeeping Service open API (**gk_Open**). This includes opens that end up invoking a stage.

2. Any stage call that is invoked on behalf of open will *not* call the Gatekeeping Service stage API (**gk_Stage**). For example, the ftp **site stage <filename>** command will use the Gatekeeping Service open API, **gk_Open**, rather than the Gatekeeping Service stage API, **gk_Stage**.

3. Direct calls to stage (**hpss_Stage**, **hpss_StageCallBack**) will call the Gatekeeping Service stage API (**gk_Stage**).

4. If the site is monitoring authorized caller requests, then the site policy interface won't be allowed to deny or delay these requests; however, it will still be allowed to monitor these requests. For example, if a site is monitoring authorized caller and open requests, then the site **gk_site_Open** interface will be called for open requests from users and the **gk_site_OpenStats** interface will be called for open requests due an authorized caller request (for example, migration by the MPS). The site policy can *not* return an error for the open due to migration; however, it can keep track of the count of opens by authorized callers to possibly be used in determining policy for open requests by regular users. Authorized caller requests are determined by the Core Server and are requests for special services for MPS. These services rely on timely responses, thus gatekeeping is not allowed to deny or delay these special types of requests.

5. The Client API uses environment variables HPSS_API_TOTAL_DELAY to place a maximum limit on the number of seconds a call will delay because of HPSS_ERETRY status codes returned from the Gatekeeper. See the *Client API configuration* section of the *HPSS Management Guide* for more information.

Refer to *HPSS Programmer's Reference* for further specifications and guidelines on implementing the Site Interfaces.

3.10. Storage characteristics considerations

This section defines key concepts of HPSS storage and the impact the concepts have on HPSS configuration and operation. These concepts, in addition to the policies described above, have a significant impact on the usability of HPSS.

Before an HPSS system can be used, the administrator must create a description of how the system is to be viewed by the HPSS software. This process consists of learning as much about the intended and desired usage of the system as possible from the HPSS users and then using this information to determine HPSS hardware requirements and the configuration of the hardware to provide the desired performance. The process of organizing the available hardware into a desired configuration results in the creation of a number of HPSS metadata objects. The primary objects created are classes of service, storage hierarchies, and storage classes.

A storage class is used by HPSS to define the basic characteristics of storage media. These characteristics include the media type (the make and model), the media block size (the length of each basic block of data on the media), the transfer rate, and the size of media volumes. These are the physical characteristics of the media. Individual media volumes described in a storage class are called Physical Volumes (PVs) in HPSS.

Storage classes also define the way in which Physical Volumes are grouped to form Virtual Volumes (VVs). Each VV contains one or more PVs. The VV characteristics described by a storage class include the VV Block Size and VV data stripe width. RAIT tape storage classes also define the parity stripe width, the Minimum Write Parity, and the Read Verification flag. If PVs are grouped one at a time, so that their data stripe width is one, they are still defined as VVs.

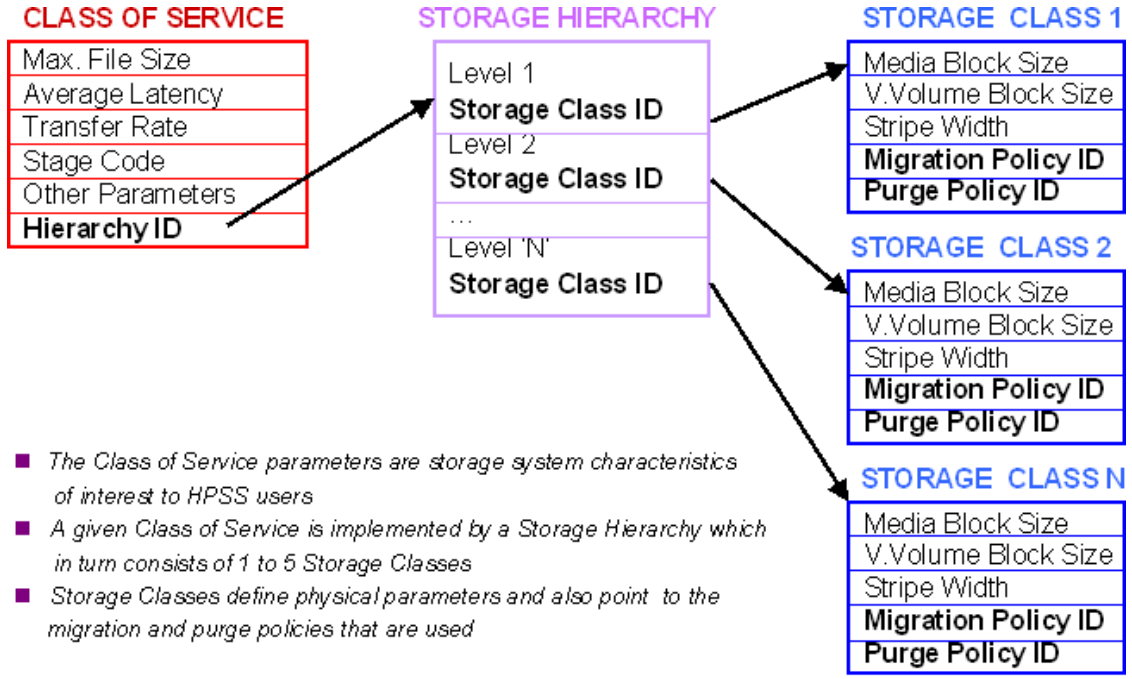
A number of additional parameters are defined in storage classes. These include migration and purge policies, minimum and maximum storage segment sizes, and warning thresholds.

An HPSS storage hierarchy consists of multiple levels of storage where each level is described by a storage class. Files are moved up and down the storage hierarchy via stage and migrate operations, based upon storage policy, usage patterns, storage availability, and user requests. If a file is recorded at multiple levels in the hierarchy, the more recent data will be found at the higher level (lowest level number) in the hierarchy.

Class of Service (COS) is an abstraction of storage system characteristics that allows HPSS users to select a particular type of service based on performance, space, and functionality requirements. Each COS describes a desired service in terms of characteristics such as minimum and maximum file size, transfer rate, access frequency, latency, and valid read or write operations. A file resides in a particular COS which is selected when the file is created. Underlying a COS is a storage hierarchy that describes how data for files in that class are to be stored in the HPSS system. A COS can be associated with a fileset such that all files created in the fileset will use the same COS.

The relationship between storage class, storage hierarchy, and COS is shown in *Figure 3.6, "Relationship of Class of Service, storage hierarchy, and storage class"*.

Figure 3.6. Relationship of Class of Service, storage hierarchy, and storage class



3.10.1. Storage class

Each virtual volume and its associated physical volumes belong to some storage class in HPSS. The SSM provides the capability to define storage classes and to add and delete virtual volumes to and from the defined storage classes. A storage class is identified by a storage class ID and its associated attributes. For detailed descriptions of each attribute associated with a storage class, see the *Configured Storage Classes window* section of the *HPSS Management Guide*.

The sections that follow give guidelines and explanations for creating and managing storage classes.

3.10.1.1. Media block size selection

Guideline: Select a block size that is smaller than or equal to the maximum physical block size that a device driver can handle.

Explanation: For example, see *Section 3.10.1.12, "Some recommended parameter values for supported storage media"* for recommended values for tape media supported by HPSS.

3.10.1.2. Virtual volume block size selection (disk)

Guideline: The virtual volume block size must be a multiple of the underlying media block size.

Explanation: This is needed for the correct operation of striped I/O. It may be necessary to experiment with combinations of disk and tape VV block sizes to find combinations that provide maximum transfer performance.

3.10.1.3. Virtual volume block size selection (tape)

Guideline 1: The VV block size must be a multiple of the media block size.

Explanation: This is needed for the correct operation of striped I/O.

Guideline 2: Pick an I/O transfer buffer size such that the size of the buffer being used to write this storage class is an integer multiple of the VV block size.

Explanation: Assume files are being written via standard FTP directly into a tape storage class. Also assume FTP is set up to use a 4 MB buffer size to write the data. This means that writes are done to the tape with a single 4 MB chunk being written on each write operation. If the tape virtual volume block size is not picked as indicated by the guideline, two undesirable things will happen. A short block will be written on tape for each one of these writes, which will waste data storage space, and the Core Server will create a separate storage segment for the data associated with each write, which wastes metadata space. Performing these extra steps will degrade transfer performance. See also *Section 3.10.1.12, "Some recommended parameter values for supported storage media"* for further information about selecting block sizes.

Guideline 3: Disk and tape VV block sizes should be equal if possible.

Explanation: The system is designed to maximize the throughput of data when it is migrated from disk to tape or tape to disk. For best results, the sizes of the VV blocks on disk and tape in a migration path should be the same. If they are different, the data will still be migrated, but the Movers will be forced to reorganize the data into different size VV blocks which can significantly impact performance.

3.10.1.4. Stripe width selection

Stripe width determines how many physical volumes will be accessed in parallel when doing read/ writes to a storage class.

Guideline 1: For tapes, the stripe width should be less than half the number of available tape drives. In the case of RAIT tapes, the total stripe width (data plus parity) should be less than half of the number of available tape drives.

Explanation: There must be enough tape drives to support the total stripe width. The **repack** and **recover** utility programs copy data from one tape VV to another, so the number of available tape drives of the appropriate type must be at least twice the total tape stripe width for these programs to function. Migration of files between tape storage classes in a hierarchy, that are of the same media type, requires at least twice as many available tape drives as the total stripe width of the storage class.

Guideline 2: Select a stripe width that results in data transmission rates to and from the drives matching or exceeding the network transmission rate.

Explanation: Configuring stripe widths that result in transmission rates that exceed the network transmission rate will waste device resources, since more hardware and memory (for Mover data buffers) will be allocated to the transfer, without achieving any performance improvement over a smaller stripe width. Also, if a large number of concurrent transfers are expected, it may be better, from an overall system throughput point of view, to use stripe widths that provide something less than the throughput supported by the network. The aggregate throughput of multiple concurrent requests will saturate the network. Overall throughput will be improved by consuming fewer device and memory resources. Note that when considering RAIT tapes, only the data stripe width should be taken into consideration. Parity information is generated or consumed by the RAIT Engines and doesn't flow across the network with the data.

Guideline 3: For smaller files, use a small tape stripe width or a stripe width of "1".

Explanation: If writing directly to tape, rather than via a disk cache, writing a file will result in the mounting and positioning of all of the tapes before data transmission can begin. This latency will be driven by how many mounts can be done in parallel, plus the mount time for each physical volume. If the file being transmitted is small, all of this latency could cause performance to be worse than if no striping were used at all.

As an example of how to determine data stripe width based on file size and drive performance, imagine a tape drive that can transmit data at about 10 MB/second and it takes about 20 seconds on average to mount and position a tape. For a one-wide stripe, the time to transmit a file would be:

(<File Size in MB>/10) + 20

Now consider a 2-wide stripe for this storage class which has only one robot. Also, assume that this robot has no capability to do parallel mounts. In this case, the transmission time would be:

(<File Size in MB>/20) + (2 × 20)

The calculation indicates that the single stripe would generally perform better for files that are less than 400 MB in size.

Writing small files to RAIT tapes involves different considerations. The smallest RAIT virtual volumes are 2+1 - two data tapes and one parity tape. When writing these volumes, all three must be mounted, positioned, and written to, regardless of the size of the file. While this configuration may take more time to write than a 1-wide tape VV, the data is more secure. The failure of a single tape can be recovered by repacking the VV, while the failure of a single 1-wide data tape will take all of the files on the tape with it.

Recording small files twice, once on each of two levels of a Class Of Service, provides a backup copy of any given file if the primary copy fails. Secondary copies can be reconstructed from the primary copy as well. But this recording strategy consumes twice as much tape as one copy of the files occupies. The tape media utilization rate is 50%.

Recording the same files to a 2+1 RAIT storage class records the files one time with the same level of backup (one tape can be lost without data loss), but the tape utilization rate is 66%.

Guideline 4: Migration can use larger stripe widths.

Explanation: The tape virtual volume is usually mounted and positioned only once when migrating from disk to tape. In this case, larger stripe widths can perform much better than smaller stripe widths.

Guideline 5: The number of drives available for media in this storage class should be a multiple of the total stripe width.

Explanation: Unless the drives are shared across storage classes (which is usually the case), if the number of drives available is not a multiple of the total stripe width then less-than-optimal use of the drives is likely.

3.10.1.5. Blocks between tape marks selection (tape only)

The number of tape physical blocks written between tape marks can be controlled. Tape marks are generated for two reasons: (1) to force tape controller buffers to flush so that the Mover can better

determine what was actually written to tape, and (2) to quicken positioning for partial file accesses. In general, larger values for Blocks Between Tape Marks are favored as modern tape drives rely on data streaming to maximize performance. Aggregation of small files into relatively large streams of data on tape is an effective way to reduce the number of tape marks and increase migration rates. For recommended values for various media types, see *Section 3.10.1.12, "Some recommended parameter values for supported storage media"*.

3.10.1.6. Minimum storage segment size selection (disk only)

The Core Server maps disk files onto a series of disk storage segments. The size of the storage segments is controlled by parameters from both the storage class configuration and the Class of Service configuration. It is determined by the **Min Storage Segment Size**, the **Max Storage Segment Size**, and the **Average Number of Segments** parameters from the storage class configuration and by the **Allocation Method** and **Truncate Final Segment** parameters from the Class of Service configuration. See the *Class of Service Configuration window* section of the *HPSS Management Guide* for a description of how these parameters work together to determine the segment size. The smallest amount of disk storage that can be allocated to a file is determined by the **Min Storage Segment Size** parameter. This parameter should be chosen with disk space utilization in mind. For example, if writing a 4 KB file into a storage class where the storage segment size is 1024 KB, then 1020 KB of the space will be wasted. At the other extreme, each file can use at most 10,000 disk storage segments, so it isn't possible to write a terabyte file to a disk storage class with a maximum storage segment size below 128 MB. Under the Classic Style **Allocation Method**, when file size information is available, the Core Server will attempt to choose an optimal storage segment size between **Min Storage Segment Size** and **Max Storage Segment Size** with the goal of creating an **Average Number of Segments** for the bitfile.

Guideline 1: Care should be taken when selecting the minimum storage segment size. If data will be migrated from disks in the storage class to a tape storage class, the value of the Minimum Storage Segment Size parameter should meet one of the following conditions. These rules help prevent the creation of excessive numbers of tape storage segments when files are migrated from disk to tape.

- If the storage segment size is larger than the tape stripe length, it should be an integer multiple of the tape stripe length. The storage segment size may be equal to the tape stripe length.

- If the storage segment size is smaller than the tape stripe length, the tape stripe length should be an integer multiple of the storage segment size, and, it should be not more than 32 times the storage segment size.

Guideline 2: When a large range of file sizes are to be stored on disk, define multiple disk storage classes with appropriate storage segment sizes for the sizes of the files that are expected to be stored in each storage class or consider using the Variable Length Allocation Method.

Explanation: The Class of Service (COS) mechanism can be used to place files in the appropriate place. Note that although the Core Server provides the ability to use COS selection, current HPSS interfaces only take advantage of this in two cases. First, the **pput** command in PFTP automatically takes advantage of this by selecting a COS based on the size of the file. If the FTP implementation on the client side supports the **alloc** command, a COS can also be selected based on file size. Files can also be directed to a particular COS with FTP and PFTP commands by using the **site setcos** command to select a COS before the files are stored. When setting up Classes of Service for disk hierarchies, take into account both the **Minimum Storage Segment Size** parameter and the **Maximum Storage**

Segment Size parameter in determining what range of file sizes a particular COS will be configured for.

3.10.1.7. Maximum storage segment size selection

Maximum storage segment size selection (disk only)

This parameter, along with **Min Storage Segment Size** and **Average Number of Storage Segments** parameters from the storage class configuration and the **Allocation Method** and **Truncate Final Segment** parameters from the Class of Service configuration, is used by the Core Server to optimally choose a storage segment size for bitfiles on disk. The largest storage segment size that can be selected for a file in a storage class is limited by this parameter.

Guideline: In order to avoid creating excessive fragmentation of the space on disks in this storage class, it is recommended that this be set no higher than 5% of the size of the smallest disk Virtual Volume allocated in this storage class.

Maximum storage segment size selection (tape only)

The **Max Storage Segment Size** field in the storage class definition allows a site to limit the maximum amount of data written into a single tape storage segment. This can be useful when storing very large files that might span multiple tapes or fill an entire tape. By limiting the storage segment size, a site can take steps to guarantee that repack will be able to copy the storage segment to another tape during repack operations.

Repack requires that an existing tape storage segment be copied to a single target tape. A value of zero in this field indicates that tape storage segments are not limited in size. A nonzero value for this field must be between 10% and 50% of the VVSize defined for the storage class.

3.10.1.8. Maximum VVs to write (tape only)

This parameter restricts the number of tape VVs, per storage class, that can be concurrently written by the Core Server. It is meant to be used in "direct to tape" systems in which user files are written directly to tapes. Its purpose is to limit the number of tapes that can be simultaneously written so that a sudden increase in the number of clients writing tapes doesn’t cause the system to attempt to mount and write a large number of tapes. The number of tape drives used to write files in the storage class will be limited to approximately the value of this field times the total stripe width defined for the storage class. Note that this field affects only tape write operations. Read operations are not limited by this parameter.

3.10.1.9. Average number of storage segments (disk only)

Under the Classic Style **Allocation Method**, this parameter, along with **Min Storage Segment Size** and **Max Storage Segment Size**, is used by the Core Server to optimally choose a storage segment size for bitfiles on disk. The Core Server attempts to choose a storage segment size between **Min Storage Segment Size** and **Max Storage Segment Size** that would result in creating the number of segments indicated by this field.

Guideline: For best results, it is recommended that small values (less than "10") be used. This results in minimizing metadata and optimizing migration performance. The default of "4" will be appropriate in most situations.

3.10.1.10. PV estimated size and PV size selection

Guideline: For tape, select a value that represents how much space can be expected to be written to a physical volume in this storage class with hardware data compression factored in.

Explanation: The Core Server uses this value as a guide in selecting tapes for writing, but regards it as an estimate only. Regardless of its value, the tape will be filled before another tape is chosen for writing.

Rule 1: For disk, the **PV Size** value must be the exact number of bytes available to be written on the PV. This value must be a multiple of the media block size and the VV block size. It may be necessary to round the actual size of the volume down to one of these multiples. The SSM will enforce these rules when the window fields are filled in.

Rule 2: For disk, the **PV Size** value must be less than or equal to the **Capacity** value described in the *Configure a new device and drive* section of the *HPSS Management Guide*.

3.10.1.11. Optimum access size selection

Guideline: Generally, a good value for **Optimum Access Size** is the stripe length, which is the virtual volume block size times the data stripe width.

Explanation: This field is advisory in nature in the current HPSS release. In the future, it may be used to determine buffer sizes. Generally, a good value for this field is the stripe length; however, in certain cases, it may be better to use a buffer that is an integer multiple of the stripe length. The simplest thing at the present time is to set this field to the stripe length. It can be changed in the future without complication.

3.10.1.12. Some recommended parameter values for supported storage media

Table 3.10, "Suggested block sizes for disk" and *Table 3.11, "Suggested block sizes for tape"* contain suggested values for storage resource attributes based on the media type. The given values are not the *only* acceptable values, but represent reasonable settings for the various media types. See other subsections in *Section 3.10, "Storage characteristics considerations"* for more information about setting the storage characteristics.

Disk media parameters

The following table contains attributes settings for the supported disk storage media types.

Table 3.10. Suggested block sizes for disk

Disk type	Media block size	Minimum access size	Minimum virtual volume block size
SCSI attached	4 KB	0	1 MB
SAS	4 KB	0	1 MB
SSD	4 KB	0	1 MB

Disk type	Media block size	Minimum access size	Minimum virtual volume block size
Fibre Channel attached	4 KB	0	1 MB

In the above table:

- "Media block size" is the block size to use in the storage class definition. For disk, this value should also be used when configuring the Mover devices that correspond to this media type. Note that this value will not limit the amount of data that can be read from or written to a disk in one operation; it is used primarily to perform block boundary checking to ensure that all device input/output requests are block aligned. This value should correspond to the physical block size of the disk device.
- "Minimum access size" is the size of the smallest access request that should regularly be satisfied by the media type. The performance of smaller accesses will be seriously degraded. A value of zero indicates that the media is suitable for supporting all data accesses.
- "Minimum virtual volume block size" is the smallest block size that should be configured for virtual volumes of the media type. Smaller values will cause increased overhead when transferring to or from volumes configured with stripe widths greater than one. Virtual volume block size has little or no effect on virtual volumes whose stripe width is one.

Note: *When SCSI, SAS, SSD, or Fibre Channel attached disks are combined to form striped virtual volumes, the minimum access size should become, at a minimum, the stripe width of the virtual volume multiplied by the virtual volume block size. If not, data access will only use a subset of the striped disks and therefore not take full advantage of the performance potential of the virtual volume.*

Tape media parameters

The following table contains attributes settings for the supported tape storage media types.

Table 3.11. Suggested block sizes for tape

Tape type	Media block size	Blocks between tape marks	Estimated physical volume size
IBM 3580 (LTO)	256 KB	2700	100 GB
IBM 3580 (LTO Gen 2)	256 KB	6300	200 GB
IBM 3580 (LTO Gen 3)	256 KB	14400	400 GB
IBM 3580 (LTO Gen 4)	256 KB	21600	800 GB
IBM 3580 (LTO Gen 5)	256 KB	25200	1.5 TB
IBM 3580 (LTO Gen 6)	512 KB	18000	2.5 TB

Tape type	Media block size	Blocks between tape marks	Estimated physical volume size
IBM 3580 (LTO Gen 7)	512 KB	27000	6.0 TB
IBM 3580 (LTO Gen 7 M8)	512 KB	27000	9.0 TB
IBM 3580 (LTO Gen 8)	512 KB	27000	12.0 TB
IBM 3580 (LTO Gen 9)	512 KB	36000	18.0 TB
IBM 3590	256 KB	1620	10, 20 GB
IBM 3590E	256 KB	2520	20, 40 GB
IBM 3590H	256 KB	2520	60, 120 GB
IBM 3592 J1A	256 KB	7200	60 GB (JJ/JR), 300 GB (JA/JW)
IBM 3592 E05 (TS1120)	256 KB	1800	100 GB (JJ/JR), 500 GB (JA/JW), 700 GB (JB/JX)
IBM 3592 E06 (TS1130)	256 KB	28800	128 GB (JJ/JR), 640 GB (JA/JW), 1000 GB (JB/JX)
IBM 3592 E07/EH7 (TS1140)	256 KB	43200	500 GB (JK), 1.6 TB (JB/JX), 4.0 TB (JC/JY)
IBM 3592 E08/EH8 (TS1150)	256 KB	54000	900 GB (JK), 7.0 TB (JC/JY)
IBM 3592 E08/EH8 (TS1150)	256 KB	64800	2.0 TB (JL), 10.0 TB (JD/JZ)
IBM 3592 55E/F/G (TS1155)	256 KB	64800	3.0 TB (JL), 15.0 TB (JD/JZ)
IBM 3592 60E/F/G (TS1160)	256 KB	72000	5.0 TB (JM), 20.0 TB (JE/JV)
IBM 3592 70F/S (TS1170)	256 KB	72000	50.0 TB (JF)

Tape type	Media block size	Blocks between tape marks	Estimated physical volume size
Sony GY-8240	256 KB	4320	60, 200 GB
Sony SAIT-1	256 KB	5400	500 GB
StorageTek 9840A	256 KB	1800	20 GB
StorageTek 9840B	256 KB	3600	20 GB
StorageTek 9840C	256 KB	5400	40 GB
StorageTek 9840D	256 KB	5400	75 GB
StorageTek 9940A	256 KB	1800	60 GB
StorageTek 9940B	256 KB	5400	200 GB
StorageTek T10000A	256 KB	21600	500 GB
StorageTek T10000B	256 KB	21600	1000 GB
StorageTek T10000C	256 KB	43200	5000 GB
StorageTek T10000D	256 KB	45720	8000 GB

In the above table:

- "Media block size" is the block size to use in the storage class definition. This is the size of the data blocks written to tape. Note that for tape devices, the Mover configuration does not contain the media block size. This value may have a significant impact on data transfer performance, as for most tape devices each input/output request must be for the media block size. If a large block size is used for relatively small write requests, space may be wasted on the tape.
- "Blocks between tape marks" is the number of media blocks to be written between tape marks. A relatively small value has the benefit of shorter positioning times to the middle of files. Small values have the penalties of poorer media utilization and lower performance when writing tapes. Since files are usually read in their entirety, and modern tape controllers employ sophisticated positioning logic and are designed to stream data to tape, larger values of the **Blocks Between Tape Mark** parameter are recommended. The values in the table above are guidelines that should provide good general performance. It is possible that better performance might be achieved via experimentation with this setting in your environment.
- "Estimated physical volume size" is the estimated size of the physical volumes to be set in the storage class definition. These values are based on the expected media to be used with the specified type. In some cases, different length tape media may be used, which may have an effect on the estimated size for a given physical volume (for example, regular or extended length 3480/3490 format cartridges). Note that the values listed do not take into account any data compression that may be performed by the tape drive. Also, note that this value is for informational purposes only and does not affect the amount of user data written to a tape volume by the Core Server. The server fills each tape Virtual Volume such that the amount of data written to the tape varies with the compressibility of the data.

3.10.2. Storage hierarchy

Each HPSS file is stored in a storage hierarchy consisting of an ordered list of storage classes. A storage hierarchy can have up to five levels starting with level 0. The highest level (first level

written to) is always level 0 and the lowest is level 4. Storage classes are expected to be arranged in a hierarchy in order of descending performance. For example, a level 0 storage class could be a fast disk while a level 4 storage class could be a slower, large-capacity tape system. The SSM provides a means to define storage hierarchies. A storage hierarchy is identified by a storage hierarchy ID and its associated attributes. For detailed descriptions of each attribute associated with a storage hierarchy, see the *Storage Hierarchy Configuration window* section of the *HPSS Management Guide*. The following is a list of rules and guidelines for creating and managing storage hierarchies.

Rule 1: All writes initiated by clients are directed to the highest level (level 0) in the hierarchy.

Rule 2: Parts or all of a file may appear at multiple levels in a storage hierarchy. If data for a file does appear at multiple levels of the hierarchy, the data at the higher level is always the more recent data.

Rule 3: Migration of data does not skip levels in the hierarchy, except in the special case of creating duplicate copies when doing disk migration.

Rule 4: The client stage command can only stage data to the top level (level 0) in the hierarchy.

Rule 5: A given storage class can only occur once in the same hierarchy.

3.10.3. Class of Service

Each HPSS file belongs to a single Class of Service (COS) which is selected when the file is created. It is selected via Class of Service hints information passed to the Core Server when the bitfile is created. If using the Client API, the application program has full access to this hints information. FTP users can use the **quote** command to set the COS. A **pput** request in PFTP automatically selects a COS based on file size unless the user explicitly selects the COS.

The SSM provides a means to define Classes of Service. A COS is identified by a COS ID and its associated attributes. For detailed descriptions of each attribute associated with a Class of Service, see the *Classes of Service* section of the *HPSS Management Guide*.

The Force Selection flag can be set in the COS definition to prevent automatic selection. If this flag is set, the COS can only be selected by asking for the COS by ID or Name.

The sections that follow give guidelines and explanations for creating and managing classes of service.

3.10.3.1. Selecting minimum file size

Guideline: This field is used to indicate the smallest file that should be stored in the COS.

Explanation: This limit is not enforced and is advisory in nature. The minimum file size can be used as a criteria for selecting a COS via the COS hints mechanism. Currently, PFTP and FTP clients that support the **alloc** command will use the size hints when creating a new file. The SSM will enforce the requirement that the **Minimum File Size** is less than the **Maximum File Size**.

3.10.3.2. Selecting maximum file size

Guideline: This field can be used to indicate the largest file that may be stored in the COS.

Explanation: If the **Enforce Max File Size** option is selected, an attempt to perform an operation on a file that would cause this value to be exceeded will be rejected. The underlying storage hierarchy should be set up so that the defined storage classes support files of this size in a reasonable fashion. For details, see *Section 3.10.1, "Storage class"* and *Section 3.10.2, "Storage hierarchy"* on storage classes and storage hierarchies. This field can be used via the COS Hints mechanism to affect COS selection. PFTP and FTP clients that support the **alloc** command will use the size hints when creating a new file.

3.10.3.3. Selecting stage code

This field determines whether a file is to be staged to the highest level in the storage hierarchy when the file is opened by one of the Client API function calls to the Core Server. This field can be used via the COS Hints mechanism to affect COS selection. Stage behavior is also affected by the value of the Auto Stage Retry flag.

Guideline 1: Select the **No Stage** option if staging of files is not desired.

Explanation: Data read from the file may come from lower levels in the storage hierarchy if the data does not exist at the top level. This option is normally selected if the top level in the hierarchy is not disk or if the users of files stored in this COS wish to control staging directly via user stage requests.

Guideline 2: Select the **Stage on Open** option to stage the entire file to the top level of the hierarchy synchronously. The Client API open operation will block while the file is being staged.

Explanation: This option is commonly selected when the top level of the hierarchy is disk and the files in this Class of Service are small to moderate in size. Use this option if you want to be guaranteed that the file is completely and successfully staged before it is read. If the stage operation fails, the open will return with an error.

Guideline 3: Select the **Stage on Open Async** option if you wish to stage the entire file to the top level in the hierarchy and do not want the Client API open to block.

Explanation: When this option is selected, the file is staged in sections and the read and write operations that access this file are blocked only until the portion of the file they reference has been completely staged. Normally, this option is selected when the top level of the hierarchy is disk and the files in this COS are fairly large in size. This option is honored only when the top level in the hierarchy is disk. If the top level is tape and this option is specified, the **No Stage** option will be used instead.

Guideline 4: Select the **Stage on Open Background** option if you want the stage operation to be queued in the Core Server and processed by a background Core Server thread.

Explanation: The Client API open request will return with success if the file is already staged. Otherwise, a stage request is placed in a queue and will be processed by the Core Server in the background. A busy error is returned to the caller. This option allows a large number of stages (up to 2000) to be queued in the Core Server and processed as resources permit. The other stage options will result with a busy error if Core Server threads are not immediately available to process the request.

Guideline 5: Select the **Auto Stage Retry flag** if you want to configure a storage class so that stage failures of the primary copy can be automatically retried from a valid second copy. The associated hierarchy must first be configured for multiple copies.

Explanation: When a stage from the primary copy fails and a secondary copy of the file is available, HPSS will usually reissue the stage operation from the secondary copy. This is typically done when a tape, holding the primary copy, becomes damaged and cannot be read. A warning that the stage operation has used the secondary copy will appear in the SSM *Alarms and Events* window.

3.10.3.4. Selecting optimum access size

This field is only advisory in nature; however, for later releases it may be used by interfaces to select good buffer sizes dynamically.

Guideline 1: Generally, if the file is being staged on open, **Optimum Access Size** should be set to the same value that **Optimum Access Size** is set to in the storage class at the top of the hierarchy.

Guideline 2: If data is not being staged to the top level before it is read (either automatically or by user command), select a value that is an integer multiple of the largest **Optimum Access Size** field found among the storage classes that make up this hierarchy.

3.10.3.5. Selecting average latency

This field can be used via the COS Hints mechanism to affect COS selection.

Guideline 1: This field should generally be set to the value of the **Average Latency** field in the storage class at the top level of the hierarchy when the **Stage on Open** option is in effect. If files are usually accessed only once after they are staged, the average latency should be set to the latency of the level they are staged from.

Guideline 2: If it is expected that most of the requests for files in this COS are read requests, then it may be best to set the value of this field equal to the **Average Latency** field in the Storage Class in the hierarchy where most of the data accesses come from.

Guideline 3: If files are written into the COS much more frequently than read, use the **Average Latency** field from the Storage Class at the top level in the hierarchy.

3.10.3.6. Selecting transfer rate

This field can be used via the COS Hints mechanism to affect COS selection.

Guideline 1: This field should generally be set to the value of the **Transfer Rate** field in the storage class that is at the top level in the hierarchy. This should always be the case if the data is being staged on open.

Guideline 2: If a large percentage of the reads are being done from a lower level in the hierarchy, consider setting the transfer rate based on the **Transfer Rate** associated with the storage class at this lower level.

3.10.3.7. StripeLength and StripeWidth hints

These fields can be used via the COS Hints mechanism to affect COS selection.

Guideline: StripeLength and StripeWidth hints are available in the hints mechanism. When specified in the hints, StripeLength and StripeWidth from the storage class at the top level of each hierarchy are used in the COS selection algorithm.

3.10.4. File families

A file family is an attribute of an HPSS file that is used to group a set of files on tape virtual volumes. When a file is migrated from disk to tape, it is migrated to a tape virtual volume assigned to the family associated with the file. If no tape is associated with a family, the file is migrated to the next available tape not associated with a family, then that tape is assigned to the family. The family affiliation is preserved when tapes are repacked. Each file in HPSS is assigned a family designation. The default family is family zero, which is interpreted by HPSS as no family affiliation. Configuring file families is an HPSS administrator function.

HPSS places no restriction on the values assigned to the File Family IDs, other than the special meaning of family zero. A name must be assigned to each file family.

To associate a file with a file family, the HPSS administrator could associate a file family with a fileset. This will cause all files created in the fileset to be assigned to a particular file family. Alternately, the HPSS administrator could allow the user to set the file family attribute of each file that they want assigned to a file family. This will need to be done just after creation, but before data transfer. See the user's guide for your file transfer applications for further information on the commands (for example, FTP supports **getfam** and **setfam** to get and set the file family for a file).

Defining multiple file families may have an impact on system migration performance. MPS may have to mount significantly more tapes to complete a migration from a disk storage class if the files are spread across a large number of file families, compared to the number of mounts that would be required if all the files were in the same family.

3.11. HPSS performance considerations

This section provides some additional hints for enhancing HPSS performance.

3.11.1. DB2

Because all HPSS operations involve DB2, it is important to optimize DB2 performance. There are a number of configuration settings that can greatly affect DB2 performance. Some of these are discussed in *Section 5.10, "Tune DB2"*. For a detailed discussion on tuning DB2, refer to the *Database Fundamentals >> Performance tuning* section under the DB2 V10.5 InfoCenter at : http://publib.boulder.ibm.com/infocenter/db2luw/v10r5/index.jsp.

The following is a list of areas to consider when optimizing DB2 performance for HPSS:

- Database usage
  - Average number of users and applications connected to DB2
  - Maximum users and applications connected to DB2
  - Nature of usage: read or update
- Database logging
  - Hardware or software mirroring

  - Disk speed and reliability: select the fastest, most reliable disk
  - To help achieve best availability and logging performance, separate the transaction logs and DB2 data, or tablespaces, onto separate spindles and onto separate LUNse
- Database recovery
  - Enabling dropped table recovery will decrease database performance
- Amount of available memory and size of buffer pools
  - Buffer pools should not be larger than the tables using them (that is, not over-allocating memory)
  - Increase buffer pool size (that is, increase memory available for metadata)
- Type of node
  - Amount of cache, memory, CPU available
- Scale Db2 transactions across multiple database partitions and distribute the partitions across multiple computers to run as many transactions in parallel as possible.
- Storage media

3.11.2. Bypassing potential bottlenecks

HPSS performance is influenced by many factors, such as device and network speeds, configuration and power of HPSS server nodes, DB2 configuration, storage resource configuration, and client data access behavior.

HPSS provides mechanisms to bypass potential data transfer performance bottlenecks, provided that the system configuration provides the additional resources necessary. For example, if the performance of a single disk device is the limiting factor in a transfer between HPSS and a client application, the disks can be reconfigured in a striped storage class to allow parallel transfers at higher transfer rates. If after forming the stripe group, the I/O or processor bandwidth of a single node becomes the limiting factor, the devices can be distributed among a number of nodes, alleviating the limitations of a single node.

If the client node or single network between the client and HPSS becomes the limiting factor, HPSS supports transferring data to or from multiple client nodes in parallel, potentially using multiple physical networks, to bypass those potential bottlenecks.

During system planning, consideration should be given to the number and data rates of the devices, node I/O bandwidth, network bandwidth, and client node bandwidth to attempt to determine a configuration that will maximize HPSS performance given an anticipated client workload.

3.11.3. Configuration

The configuration of the HPSS storage resources (see *Section 3.10, "Storage characteristics considerations"*) is also an important factor in overall HPSS performance, as well as how well the configuration of those resources matches the client data access patterns.

For example, if a site provides access to standard FTP clients and allows those clients to write data directly to tape, the buffer size used by the FTP server and the virtual volume block size defined for the storage class being written to will have a significant impact. If the buffer size used by the FTP server is not a multiple of the virtual volume block size, each buffer written will result in a distinct storage segment on the tape. This will cause additional metadata to be stored in the system and extra synchronization processing of the tape. However, if the buffer size is a multiple of the virtual volume block size, each write will continue to append to the same storage segment as the previous write. This will continue until the final write for the file, which will usually end the segment, thus reducing the amount of metadata generated and media processing.

3.11.4. FTP/PFTP

Data transfers performed using the standard FTP interface are primarily affected by the buffer size used by the FTP daemon. The buffer size can be configured as described in the *FTP/PFTP daemon configuration* section of the *HPSS Management Guide*. It should be a multiple of the storage segment size, if possible. Otherwise, it should be at least a multiple of the virtual volume block size. If the buffer size is too small, the FTP daemon will need to issue a large number of individual read or write requests; however, if the buffer size is too large, the FTP daemon will require a large amount of memory, which may cause additional paging activity on the system.

The size of the FTP daemon buffer is extremely important if the FTP clients write files directly to a tape storage class, as described in *Section 3.11.3, "Configuration"*.

Parallel FTP (PFTP) uses TCP/IP to move data.

Note that the PFTP data transfer commands (such as **pput** and **pget**) are not influenced by the FTP daemon buffer size because the data flows directly between the client and Movers.

Note that PFTP clients that use the standard FTP data transfer commands (such as **put** and **get**) have the same performance considerations as standard FTP clients.

Parallel transfers move the data between the Mover and the end-client processes bypassing the HPSS FTPD. Users should be educated to use the parallel functions rather than the non-parallel functions.

ASCII transfers are not supported by the parallel functions and the non-parallel functions will need to be specified for ASCII transfers. ASCII transfers are *not* typically required, but the end-customer should familiarize themselves with the particulars.

Parallel transfers should be optimized so that the Class of Service (COS), media stripe widths, network stripe widths, and Parallel Block Sizes are consistent with each other. For example, using a network stripe width of "4" with a media width of "2" may result in poorer performance than if both specifications are equal. Specifying a network stripe width of "4" where there is only one network interface may not provide any improvement over a lower network stripe width ("2") if the bandwidth of the network is (over-)filled by a 4-way stripe.

Non-parallel transfers occur via a "stage and forward" approach (device <==> Mover <==> HPSS FTP daemon <==> FTP client.) It is recommended that the "Non-Parallel Hostname" option be specified in the `HPSS.conf` file if the FTP daemon system has multiple interfaces. The hostname should refer to the highest speed interface available for transmission of data between the FTP daemon and HPSS Movers.

Where reasonable, the standard FTP ports should be modified to something other than 20/21 on the system acting as the HPSS FTP daemon. The HPSS FTP daemon should be set to use the 20/21 ports by default. This reduces the problem of requiring the end-customer to know which port to use for transfers to HPSS. In conjunction with this, it is highly recommended that the {ftpbanner} file be used with an appropriate message to provide information to the end-customer that they are accessing HPSS as opposed to a standard system.

3.11.5. Client API

The Client API provides the capability to perform data transfer of any size (the size being parameters supplied by the client to the read and write interfaces). The size of the data transfers can have a significant impact on the performance of HPSS. In general, larger transfers will generate less overhead than a series of smaller transfers for the same total amount of data.

The size of the transfers is extremely important because the clients may write files directly to a tape storage class, as described in *Section 3.11.3, "Configuration"*.

3.11.6. Core Server

Minor performance degradations may be seen when the Core Server is processing path names with a large number of components, servicing requests for objects which have a large number of Object ACL entries, and servicing create requests for objects which have Initial Container/Initial Object ACL entries.

3.11.7. Location Server

The location policy defined for a site generally determines how the Location Server will perform and how it will impact the rest of the HPSS system. View the help for the fields on this screen to determine if the values need to be changed. The default policy values are adequate for the majority of sites. Usually, the only time the policy values need to be altered is when there is an unusual HPSS setup.

The Location Server itself will give a warning when a problem is occurring by posting alarms to SSM. Obtain the information for the Location Server alarms listed in the *HPSS Error Manual*. To get a better view of an alarm in its context, view the Location Server's statistics screen.

If the Location Server consistently reports a heavy load condition, increase the number of request threads and recycle the Location Server. Remember to increase the number of threads on the Location Server's basic server configuration screen as well. Note that a heavy load on the Location Server should be a very rare event.

3.11.8. Logging

Excessive logging by the HPSS servers can degrade the overall performance of HPSS. If this is the case, it may be desirable to limit the message types that are being logged by particular servers. There are two ways to do this.

The most effective way of managing excessive logging also provides the least fine-grained control. The server Logging Policy can be updated to control which message types are logged. A default Log

Policy may be specified to define which messages are logged. Typically, Trace, Accounting, and Request messages are not logged. Other message types can also be disabled. Once the Logging Policy is updated for one or more HPSS servers, it will begin being used the next time those servers scan to update their logging policies (usually within 30 seconds).

Another way to manage excessive logging, which provides more control, but less overall performance benefit, is to use syslog filtering. This can be done using normal syslog filtering mechanisms to prevent messages from being logged. However, HPSS still forwards the messages to syslog.

3.11.9. Cross-realm trust

Cross-realm trust should be established with the minimal reasonable set of cooperating partners. Excessive numbers of cross-realm connections may diminish security and cause performance problems due to Wide Area Network (WAN) delays. The communication paths between cooperating realms should be reliable.

3.11.10. Gatekeeping

Sites may choose to implement site policy in the Gatekeeper for load balancing create, open, and stage requests. The site policy could limit the maximum number of non-authorized caller requests allowed at once by either delaying or denying particular requests. To delay the request, the site policy may return a special retry status along with the number of seconds to wait before the Client API retries the request. Delaying requests should limit the number of create, open, and stage requests performed at a particular point in time, thus decreasing the load on the system. However, care must be taken to figure out the best retry wait scheme to meet the requirements for each site and to configure the correct number of Gatekeepers if the load on one Gatekeeper is heavy. (Note: The maximum number of Gatekeepers per storage subsystem is one.) Also, sites need to write their Site Interfaces optimally to return in a timely manner.

Two special error status codes (**HPSS_ETHRESHOLD_DENY** and **HPSS_EUSER_DENY**) may be used to refine how a site may deny a create, open, or stage request. If the Core Server receives either of these errors, then it will return this error directly to the Client API rather than performing a retry. Errors other than these two or the special **HPSS_ERETRY** status will be retried several times by the Core Server. See either volume of the *HPSS Programmer's Reference* for more information.

Create, open, and stage requests from authorized callers (MPS) can *not* be delayed or denied due to timing sensitivity of the special requests these servers make to the Core Server. For example, migration of a file by MPS is an authorized caller open request. The site policy could keep track of authorized caller requests to further limit non-authorized caller requests.

If a Gatekeeper is being used for Gatekeeping Services, then the Core Server for each storage subsystem configured to use a particular Gatekeeper will return errors for the create, open, and stage requests being monitored by that Gatekeeper when that Gatekeeper is down. For example, if storage subsystem #2 is configured to use Gatekeeper #2, and Gatekeeper #2 is monitoring open requests and is DOWN, then each open by the Core Server in storage subsystem #2 will eventually fail after retrying several times.

3.11.11. HPSSFS-FUSE interface

Refer to the *HPSSFS-FUSE Administrator's Guide*, bundled with the HPSSFS-FUSE RPM.

3.12. HPSS metadata backup considerations

This section contains guidelines for proper maintenance of the HPSS metadata stored in DB2.

The policies described should be fully understood and implemented to protect the HPSS metadata. Failure to follow these policies can lead to unrecoverable data loss.

The remainder of this section is a set of rules associated with backing up HPSS metadata. Though automated archive software like Tivoli Storage Manager is used to backup and protect DB2 data, it is important that each site review this list of rules and check to ensure that their site’s backup is consistent with these policies.

When deciding on the size and number of disks for metadata, keep in mind the following:

1. At a minimum, for subsystem-specific metadata, the disks hosting the DB2 instance installation path (that is, the home directory of the instance owner) and the databases' tablespaces should be separate. This is not critical for configuration metadata since it changes infrequently.

2. Ideally, several physical disks should be available for DB2 tablespaces (for example, so that name space data can be on one and bitfile data on another).

3. If the backup software employed involves some sort of disk staging area, this should be on a separate disk, and a large amount of disk should be allocated for this purpose.

For reliability and availability reasons, the disk hosting the instance installation data should be mirrored. The disks hosting the tablespaces should also be mirrored if possible. For performance and reliability reasons, mirroring should be done using separate physical devices.

3.13. HPSS security considerations

The security requirements between sites differ widely. The HPSS System Administrators must be aware of the sites security requirements and should be aware of the security configuration required in HPSS. Sites should contact HPSS support if they have questions regarding HPSS security. For more information on security, see the *Security and system access* chapter of the *HPSS Management Guide*.

Chapter 4. System preparation

This section covers the steps that must be taken to appropriately prepare your system for installation and configuration of HPSS and its infrastructure.

- *Section 4.1, “General setup”*
- *Section 4.2, “Set up file systems”*
- *Section 4.3, “Set up tape libraries”*
- *Section 4.4, “Verify tape drives”*
- *Section 4.5, “Set up disk drives”*
- *Section 4.6, “Set up network parameters”*

  To understand and optimize the operation of HPSS, some baseline measurement, evaluation, and tuning of the underlying network and IO subsystems is necessary. Appropriate tuning of these components is necessary for HPSS to perform as expected. Any one component that is not performing at its optimal rate will have an adverse effect on the performance of the entire system. The steps and tools listed in this chapter will help a site make the best use of their available resources. The measurements taken should be saved for future reference. If performance problems occur later on, these values can be compared with new measurements to determine if the performance problems are related to changes in the subsystems. Though disk and tape configurations rarely change without an administrator’s knowledge, networks can "mysteriously" degrade for a variety of reasons. This is especially true for client access to the HPSS system.

4.1. General setup

- Download copies of the *HPSS Installation Guide* and *HPSS Management Guide* for the appropriate version of HPSS. It will probably be useful to print copies of these documents and keep them handy while installing HPSS.
- Install, configure, and verify the correct prerequisite operating system version on all HPSS, Kerberos client and/or server, and DB2 nodes.
- Check the format of `/etc/hosts`. If `/etc/hosts` is used to augment DNS, fully qualified hostnames must be listed first. For example:

  ```
  123.45.6.789 host1.domain.gov host1
  ```

- Verify the current operating system level:

  ```
  % uname -a
  ```

- Create appropriate HPSS UNIX user accounts. The "hpss" user and group ID should be created at this time.

- Install the prerequisite software for Kerberos, C compiler, Java, Perl, SSH, and any other specific software that will be used by HPSS. Verify the correct patch levels are, or will be, installed. Refer to *Section 3.3, "Prerequisite software considerations"* for additional information.

- Configure the Perl prerequisite software on HPSS nodes.

- Configure the SSH prerequisite software on the core HPSS server node (at a minimum) and configure SSH to accept connections from IBM Houston. Include the Houston subnet IP addresses 192.94.47 and 12.39.169 in the local firewall routing rules, as necessary.

- Contact your HPSS Support representative to get a copy of the HPSS Test Plan (HTP) and install it on each node. Run and become familiar with the **lsnode** tool, which will be helpful in other steps. Additional support tools are provided on the HPSS Administrator Wiki (https://hpss-collaboration.clearlake.ibm.com/adminwiki/doku.php?id=start) under support → support_tools.

  To run **lsnode** and save the output to `/var/hpss/stats/lsnode.out`:

  ```
  % cd /<HTP_install_location>/config
  % ./lsnode.ksh > /var/hpss/stats/lsnode.out
  ```

- Obtain your HPSS Realm ID from HPSS support. Make sure that the Realm ID does not exceed 32 bits. This information will be needed when **mkhpss** is used to configure HPSS. For an existing installation, this is the ID which was previously referred to as the DCE Cell ID.

4.2. Set up file systems

The following sections describe how to set up the various file systems used by HPSS and DB2.

4.2.1. DB2 file system

Each database in DB2 has its own log. We recommend that these logs be placed on a file system reserved exclusively for this purpose. This should be a fast, but more importantly, stable disk, preferably a RAID device. For example, the following file systems might be created for a configuration database and subsystem database:

```
/db2_log/cfg
/db2_log/subsys1
```

The NEWLOGPATH database configuration variable can be used to direct the database logs to these file systems.

Additionally, the following file systems are required for storing the mirrored copy of the DB2 logs:

```
/db2_logmirror/cfg
/db2_logmirror/subsys1
```

Additionally, the following file systems are required for storing the archived copy of the DB2 logs:

```
/db2_logarchive1/cfg
/db2_logarchive1/subsys1
/db2_logarchive2/cfg
```

```
/db2_logarchive2/subsys1
```

The DB2 log mirroring is configured using the MIRRORLOGPATH variable.

The home directory of the DB2 instance owner should be created as a separate file system on each HPSS node that runs a database instance. By default, the DB2 instance owner is "hpssdb" and its home directory is `/db2data/db2_hpssdb`.

4.2.2. HPSS file system

Configure `/var/hpss` as a separate file system on each HPSS server node. This file system will store HPSS configuration files, log files, MPS reports, and other HPSS-related files. It is recommended that this file system be at least 1 GB in size.

Configure `/var/hpss/adm/core` as a separate file system on each HPSS server node. If an HPSS process fails and creates a core file, it will be placed in this location. It is recommended that this file system be configured with at least 2 GB of disk space on server nodes and 1 GB on Mover nodes.

Configure `/db2data/db2_hpssdb` as a separate file system on the HPSS core server node. This file system stores the HPSS DB2 instance configuration information as well as the *CFG* database tables. This file system should be created with at least 1 GB of disk storage and, like many of the other file systems related to HPSS, it should be monitored for fullness periodically by the administrator.

Configure `/opt/hpss` as a separate file system on each of the HPSS server and Mover nodes. This directory contains the HPSS executables, documentation, libraries, and, for those sites with source code, the development environment to build the HPSS system. For most sites, this file system can be configured with 1 GB of space. For those sites with source code, at least 2 GB are required.

4.3. Set up tape libraries

4.3.1. Oracle StorageTek

For an Oracle StorageTek tape library:

- If using an Oracle StorageTek tape library, configure the ACSLS (server) and SSI (client) software properly and verify that it is working correctly.

  To test the ability to mount and dismount a tape in an Oracle StorageTek library, use the **stk_ctl** utility.

  To mount a tape:

  ```
  % stk_ctl mount <driveSpec> <tapeLabel>
  ```

  where **driveSpec** is four integers separated by commas (no spaces), identifying the ACS, LSM, panel, and drive (for example, "0,0,1,2").

  To dismount a tape:

  ```
  % stk_ctl dismount <driveSpec>
  ```

To query a drive:

```
% stk_ctl query <driveSpec>
```

Refer to the *STK PVR-specific configuration window* section in the *HPSS Management Guide* for more information.

4.3.2. AML

The AML PVR is supported by special bid only.

For AML tape libraries:

- If using an AML PVR, configure the Insert/Eject ports using the configuration files `/var/hpss/etc/AML_EjectPort.conf` and `/var/hpss/etc/AML_InsertPort.conf`.

Refer to the *AML PVR-specific configuration window* section of the *HPSS Management Guide* for more information.

4.3.3. SCSI

For a SCSI-connected tape library, configure the SCSI Medium Changer (SMC) library device on the node that will run the HPSS SCSI PVR, and verify that it is operational.

To test the ability to mount and dismount a tape in a SCSI-connected library, use the **umccp** utility.

To start **umccp**:

```
%umccp <devname>
```

where **devname** is the SMC device for that library. The device names for SMCs can be determined using the **device_scan** utility. **Umccp** will inventory the library before prompting you for a command.

To mount a tape:

```
umccp:  mount <cartridge> <drive address>
```

where cartridge is a 6-character cartridge label and drive address is one of the addresses given by the **umccp drives** command.

To dismount a tape:

```
umccp:  dismount <drive address> [destination address]
```

If both parameters are used, the cartridge will be dismounted to the destination address. Otherwise, the first free address will be used.

To query a drive:

```
umccp:  query drive
```

This will query all drives.

Note that **umccp** does not modify its internal inventory as you perform operations. Use the **umccp inventory** command to update the inventory state.

Refer to the *SCSI PVR-specific configuration window* section in the *HPSS Management Guide* for more information.

4.4. Verify tape drives

Verify that the correct number and type of tape devices are available on each tape Mover node.

Repeat the above steps for each tape drive.

4.4.1. Linux

On each tape Mover node, verify that each tape drive has the variable-length block size option enabled.

To determine if the variable block size option is enabled, the following should complete successfully:

```
% dd if=/dev/null of=/dev/rmt/tps2d6nr bs=80 count=1
% dd if=/dev/null of=/dev/rmt/tps2d6nr bs=1024 count=1
```

If the variable-length block size option is not enabled, consult your driver documentation for procedures to enable it.

On each tape Mover node, verify that the raw read and write I/O performance of all HPSS tape drives are at the expected levels. Create one or more tables documenting the results. The HPSS Test Plan (HTP) provides a baseline test for generating tape drive performance outputs.

To conduct the read and write tests on **st1**, mount a scratch tape on **st1** and issue the following commands.

The contents of this tape will be overwritten by iocheck, so be sure to mount the correct tape cartridge.

To measure uncompressed write performance (but note that specifying nst will cause the tape to not rewind):

```
% iocheck -w -t 20 -b 1mb /dev/nst1
```

To measure the maximum-compressed write performance on st1 (and then rewind the tape):

```
% iocheck -w -t 20 -f 0 -b 1mb /dev/st1
```

To measure read performance on drive st1 using the previously-written uncompressed and compressed files:

```
% iocheck -r -t 20 -b 1mb /dev/nst1
```

```
% iocheck -r -t 20 -b 1mb /dev/nst1
```

To empty the tape:

```
% mt -f /dev/st1 rewind
% mt -f /dev/st1 weof 2
```

4.5. Set up disk drives

4.5.1. Linux

For Linux platforms, specific commands and syntax are not listed in this document. Perform the following steps using the appropriate operating system commands:

- Verify that the correct number and type of disk devices are available on each DB2 and disk Mover node.
- Create all necessary raw disk volumes to be used by the HPSS disk Movers. If a file system interface is being used, then create the sparse files to be used as devices.
- On each disk Mover node, measure the raw read and write I/O performance of all HPSS disks and verify that they are at expected levels. Create one or more tables documenting the results. The output of these tests should be stored in `/var/hpss/stats` for later analysis.

4.6. Set up network parameters

- Install and configure all network interfaces and corresponding network connections.

  Refer to IBM's internal network technologies home page for resources on configuring and tuning networks and TCP/IP.

  The network interfaces section of the **lsnode** report from each node shows the network interfaces that are configured.

  To determine how many network interfaces are available (Linux):

  ```
  % netstat -i
  ```

  For interface information, use the Name column from **netstat -i** in the **ifconfig** commands. Note the IP address follows the inet phrase in the output from the **ifconfig** command.

  To test whether an IP address is reachable (nonzero exit status indicates the **ping** was not successful):

  ```
  % ping -c 1 <ipAddress>
  ```

- Isolate Kerberos communication to the designated control networks on all HPSS and DB2 nodes in order to separate the HPSS control and data paths.
- Place all HPSS, DB2, and Kerberos server node IP addresses in a local host table (`/etc/hosts`).

For Linux, changes should be made to `/etc/nsswitch.conf`:

```
hosts: nis dns files
```

- For each AIX Ethernet network interface, verify that the **en0** and **et0** interfaces are not both configured at the same time (we recommend only using **en0** unless the other nodes in the network are all using the **802.3 et*** interface). Configure the local name service with the unique hostname for each network interface on all nodes and verify that each hostname is resolvable from other nodes.

- Verify that network TCP throughput has been optimized and the performance of each network is at expected levels in both directions (especially check HPSS data networks between Movers and between Mover and client nodes). Using **iperf** or another network tool, measure the performance of all the networks that will be used in communications between Kerberos, HPSS, DB2, and client nodes. If multiple paths between nodes are to be used, then all of them need to be measured as well. The transfer rates for networks differ greatly depending upon the individual technology and network settings used. It is important to gather performance data using a variety of settings to determine the optimal combinations. The primary values that govern performance include send/receive buffers, size of reads and writes, and **rfc1323** value for high performance networks (such as Gigabit Ethernet). Create a table showing these values.

Consult the network tool documentation for measuring performance. The HPSS Test Plan (HTP) has a test which can be used for generating network performance results using **iperf**.

Each node's send and receive performance should be measured for each interface planned for use. Multiple block sizes should be tried to find the optimal settings for your network.

You are looking for the best values possible for each network connection. These values will be used by HPSS to optimize its data transfers. This example is, by no means, a complete picture of what controls network performance. In fact, it is assumed that you have already optimized the networks. The reason for gathering these values is to optimize HPSS performance on an already tuned network, not to fix underlying network problems.

HPSS makes extensive use of a system's networking capabilities. Therefore, the setting of the tunable networking parameters for the systems on which the various HPSS servers and clients will run can have a significant impact on overall system performance.

Some options that typically impact performance within an HPSS system environment are:

Table 4.1. Network options

Network option	Description
thewall	Controls the maximum amount of system memory that can be used by the system networking code. A value that is too low can cause networking requests to be delayed or denied. The recommendation from AIX development is to set this value to at least two times the maximum number of concurrent connections times the size of the socket send/

Network option	Description
	receive buffers. The default setting for AIX 4.3.2 and later is the smaller of (1) half the size of physical memory or (2) 1 GB.
sb_max	Controls the maximum size allowed for send and receive buffers for a socket.
udp_recvspace	Controls the default size of the receive buffer for UPD/IP sockets. A value that is too small can cause server RPC sockets to be overrun.
tcp_recvspace, tcp_sendspace	Controls the default size for the receive and send buffers for TCP/IP sockets. Internally, HPSS servers and clients attempt to set these buffers sizes explicitly, but other utilities may not.
rfc1323	Controls whether large TCP window sizes are used. Usually set to ON for higher throughput networks (such as 10 Gb Ethernet) and set to OFF for lower throughput networks (such as 10 Mb Ethernet).

It is recommended that the available combination of options be tested as part of the initial HPSS system testing. In particular, poor network performance has been experienced where options on one system do not match options on other remote systems.

There are also attributes that are specific to the individual network interface that may affect network performance. It is recommended that the available interface-specific documentation be referenced for more detailed information.

The anticipated load should also be taken into account when determining the appropriate network option settings. Options that provide optimal performance for one or a small number of transfers may not be the best settings for the final multi-user workload.

4.6.1. `HPSS.conf` configuration file

The `HPSS.conf` configuration file contains tuning options to be used by HPSS clients and servers.

The `HPSS.conf` file may also contain options used by non-HPSS applications. Application developers are asked to observe the "Reserved for Future Use" components specified in Appendix D.

The `HPSS.conf` configuration file is located in the directory named by either:

- The **HPSS_CFG_FILE_PATH** environment variable,
- the directory `/usr/local/etc`, or
- the directory `/var/hpss/etc` (preferred),
- in that order. If the file is not present or no matching entry is found, the Parallel FTP Client, Client API, and Mover will use system defaults.

See Appendix D: *Appendix D, `HPSS.conf` configuration file* for more details.

4.7. Port mapping and firewall considerations

If your Core Server machines are behind a firewall and you wish to access your HPSS system from an application that uses the HPSS client library from the other side of the firewall, you will have to configure both HPSS and the firewall to allow for this. The HPSS client library will connect to the core servers, gatekeepers, and location servers configured into your system. In order for this to work, you will have to configure HPSS so that these servers listen for connections on a (small) range of ports. You will also have to configure the firewall to allow incoming connections to this range of ports and to port 111 (the RPC portmapper).

By default, when an HPSS server starts, it will select a random ephemeral port number on which to listen for connections. It registers this port number with the portmapper daemon running on that host. When a client application wishes to communicate with the server, it will contact the portmapper to find out which port the server is listening on. It will then connect to the server on that port. The Core Servers, Gatekeepers, and Location Servers can be configured to select a listening port from a small range specified by an HPSS environment variable. The client will still have to contact the portmapper to determine which of these ports the server is listening on, but since they are listening on a predetermined fixed range of ports, the firewall can be configured to allow connection through to these ports.

The port range is specified using the HPSS_LCG_SERVER_RPC_PORT_RANGE environment variable specified in the `/var/hpss/etc/env.conf` file on your Core Server machines with the following format:

```
HPSS_LCG_SERVER_RPC_PORT_RANGE=LOW-HIGH
```

where LOW and HIGH are integers that specify the port range. LOW and HIGH are included in the range. For example, if you specify:

```
HPSS_LCG_SERVER_RPC_PORT_RANGE=29913-29915
```

then any Core Server, Location Server, or Gatekeeper that runs on that machine will always use one of the ports: 29913, 29914, or 29915. If no ports in the range are available at the time the server starts it will abort with an error message logged to the SSM *Alarms and Events* window. Therefore, it is important to configure a wide enough range so that each of these servers can be allocated a port. For example, if you have two Core Server machines, A and B with the following configuration:

Machine A:

- Core Server 1
- Location Server
- Gatekeeper 1

Machine B:

- Core Server 2

Gatekeeper 2

You will have to configure Machine A with a range of at least 3 ports and Machine B with at least 2 ports. Note that the range of ports for the two machines can overlap (or be the same), since it is the combination of the machine interface name and the port number that must be unique.

It is sometimes convenient to configure the port range to larger than the minimum number. When you stop a server, or it exits, the machine operating system will keep that port in a TIME-WAIT state for several minutes before allowing reuse of the port. Configuring an extra one or two ports in the range will allow you to stop and restart a server without having to wait for the port to time out.

4.8. Semaphore values

Adjust the semaphore values (do this for the Core Server and Movers). The example that follows is for a system with 64 GB of memory:

1. Determine the amount of system memory.

```
# grep "MemTotal" /proc/meminfo
MemTotal:      67108864 kB
```

Memory in bytes = 67,108,864 × 1,024 = 68,719,476,736
Memory in GB = 67,108,864 / 1,024 / 1,024 = 64

2. Calculate the following variables which will be used to set the semaphore settings in `/etc/sysctl.conf`. + In RHEL 7.8 a 32768 limit was added to the SEMMNI and MSGMNI values. If the calculated value is above the new limit then 32768 will need to be used.

Table 4.2. Kernel parameter expressions

Memory_in_Bytes	MemTotal × 1024
Memory_in_GB	MemTotal / 1024 / 1024
shmmni	256 × Memory_in_GB
msgmni	1024 × Memory_in_GB
shmmax	Memory_in_Bytes
shmall	2 × Memory_in_Bytes / 4096
sem	4096 2048000 32 <256 × Memory_in_GB>
msgmax	65536
msgmnb	65536

3. Open `/etc/sysctl.conf` in an editor and add a section to the file to include the entries listed below. If the system variable does not appear in the file, then add it.

Example:

```
# ----------------------------------
# Semaphore/memory values for HPSS/DB2

kernel.shmmni = 16384
```

```
kernel.msgmni = 65536

# Controls the maximum shared segment size, in bytes
kernel.shmmax = 68719476736

# Controls the maximum number of shared memory segments, in pages
kernel.shmall = 33554432

# Semaphore setting
kernel.sem = 4096 2048000 32 12032

# Controls the maximum size of a message, in bytes
kernel.msgmnb = 65536

# Controls the default maximum size of a message queue
kernel.msgmax = 65536

# ----------------------------------
```

4. Set randomize_va_space in `/etc/sysctl.conf`. If this setting does not appear, then add it.

```
kernel.randomize_va_space = 0
```

Here is the OSB (Operational Service Bulletin) describing why this setting is turned off: https://hpss-collaboration.clearlake.ibm.com/adminwiki/doku.php?id=support:osb:916_db2_aslr_issue_

5. Set the hung task timeout to 900 seconds when setting up a system which will be used as a tape Mover. This will prevent the system from getting hung task kernel messages during long-running tape operations.

```
kernel.hung_task_timeout_secs = 900
```

6. Commit the changes to `sysctl.conf` to the current environment.

```
# sysctl -p
```

4.9. Enable Core Dumps

While HPSS is generally very stable, at times a site may encounter a server crash. Since we can never know when a crash is likely to occur or whether it will be able to be triggered again, core dumps should always be configured on for HPSS Core, Mover, and Client systems so that we can capture information about a crash when it occurs.

You can check for the correct setting by reviewing the ulimit settings for root and other HPSS user profiles. It should be unlimited:

```
$ ulimit -c
unlimited
```

If it is not unlimited, review your operating system configuration materials on enabling core dumps. Generally, the steps involved require setting the core file ulimit. This should be done for root and all other HPSS user profiles:

```
ulimit -c unlimited
```

another common solution is to change the system-wide core file size soft limit in `/etc/security/limits.conf` like:

```
*               soft    core            unlimited
```

HPSS core files can be very large (gigabytes), based upon various configuration parameters such as the number of concurrent threads and configured I/O limits. We suggest setting the limit to unlimited to avoid a situation where we receive a truncated core file. Truncated core files are not useful in troubleshooting.

Installations should test that core dumps are occurring correctly by sending **kill -ABRT <hpss process id>** to an HPSS process such as `hpss_ls`. This will cause the server to die and attempt to drop a core dump in a subdirectory under `/var/hpss/adm/core`. Verify that a nonzero-sized core dump has appeared and restart the process that was killed.

Chapter 5. HPSS installation and infrastructure configuration

This chapter provides instructions and supporting information for installing the HPSS prerequisite software, extracting the HPSS and DB2 software from the HPSS distribution media, and performing the HPSS infrastructure configuration.

To install and set up an HPSS system, we recommend that the administrator be familiar with UNIX commands and configuration, be familiar with a UNIX text editor, and have some experience with shell scripts.

For information on upgrading from a previous version of HPSS, see the HPSS Conversion Guide.

The steps required to install and set up an HPSS system are listed below. Each step is discussed in more detail in the section referenced.

- *Section 5.1, "Prepare for installation"*
- *Section 5.2, "Install prerequisite software"*
- *Section 5.3, "Install HPSS with RPMs"*
- *Section 5.5, "Configure HPSS infrastructure"*
- *Section 5.6, "Prepare post-installation procedures"*
- *Section 5.7, "Locate HPSS documentation and set up manual pages"*
- *Section 5.8, "Define HPSS environment variables"*
- *Section 5.10, "Tune DB2"*

5.1. Prepare for installation

The following sections discuss the steps that need to be taken in order to prepare the system for HPSS installation.

5.1.1. Distribution media

Obtain the HPSS software from HPSS support.

5.1.2. Software installation packages

The HPSS software is provided in the following packages:

- HPSSSource-<*release*>.tar.Z - Contains HPSS source files.

- SSMHelp.<*release*>.tar - Contains the *HPSS Management Guide* in HTML format which is used by the SSM's Help menu to display window-specific help topics.
- hpss.<*release*>_doc.tar - Contains all HPSS documentation in PDF format.

5.1.3. Create owner account for HPSS files

The HPSS software must be installed by a **root** user. In addition, a UNIX User ID of *hpss* and Group ID of *hpss* is required for the HPSS installation process to assign the appropriate ownership for the HPSS files. If the *hpss* User ID does not exist, the installation process will create it with the **mkhpss** tool based on default UID and GID values defined by the `~hpss/include/hpss_env_defs.h` file. If alternate IDs are required, this include file should be updated prior to building, or rebuilding, **mkhpss**, or the *hpss* account/group should be created beforehand using standard administrative procedures. Also, note that HPSS will need a number of other user and group IDs for various servers and prerequisite components. These IDs are also predefined in the same include file and can be modified to support a site's particular account policy.

It is very important that the HPSS file permissions be set up properly. If they are not, HPSS may not work properly after it is configured. We recommend that the HPSS file ownerships and permissions set by the installation process be preserved. If they must be changed, care must be taken to ensure they are changed correctly. Refer to Section 5.6, "Prepare post-installation procedures" for more information on the HPSS file ownerships and permissions.

5.1.4. Installation target directory preparation

By default, the HPSS software is installed in the `/hpss_src` directory. Before installing the HPSS software, make sure that the installation target directory satisfies the following conditions:

- The `/hpss_src` directory is not being used.
- The disk, where the installation target directory resides, has enough space to hold all the HPSS packages to be installed on this node.

Do not use NFS-mounted directories for installing nor for allocating space for HPSS-related components. Installing on NFS is problematic and the errors can be difficult to diagnose.

5.2. Install prerequisite software

This section provides an overview of how to install the prerequisite software to prepare for the upcoming HPSS configuration. Verify that the correct software versions are obtained as described in the release notes for the version being installed at https://hpss-collaboration.clearlake.ibm.com/adminwiki/doku.php?id=docs:release_notes.

5.2.1. Install Java

Java is required to compile HPSS software. It is also required to run the SSM Command Line Interface (**hpssadm**). If the site needs to compile the HPSS code, the Java Software Development

Kit (SDK) must be installed. To run HPSS applications, either the Java SDK or the Java Runtime Environment (JRE) is required.

It is recommended that the Java SDK component be installed on the machines where HPSS code is to be compiled. Also, the JRE component should be installed on each SSM client machine. This is to enable the **hpssgui** application to run directly on the client machine to maximize **hpssgui** performance and to eliminate the X traffic on the HPSS server machine.

See *Section 3.3.1.5, "Java"* to obtain the download website. Follow the installation instructions provided on the website to install Java. Be sure that all Java's prerequisite requirements are met before installing the product.

5.2.2. Install Jansson

HPSS uses the Jansson library to handle JSON inputs and outputs. Jansson is included with RHEL distributions, but make sure that it matches or exceeds the level in the *Release Notes*.

5.2.3. Install TI-RPC

HPSS uses the transport-independent RPC runtime (TI-RPC). This is included with RHEL distributions. Ensure that the level installed matches or exceeds the level in the *Release Notes*.

5.2.4. Install Ncurses

The ncurses library is required to run the **mkhpss** tool. Ncurses is included with RHEL distributions. Follow the instructions provided with the installation files to install ncurses.

5.2.5. Install MIT Kerberos

The capability to use MIT Kerberos authentication is provided in HPSS; however, IBM Service Agreements for HPSS do not provide support for defects in MIT Kerberos. Kerberos maintenance and support must be site-provided.

For Linux, Kerberos is normally included in the operating system and does not need to be installed.

5.2.6. Install LDAP (if using LDAP authorization)

LDAP authorization is not supported by IBM Service Agreements. The following information is provided for sites planning to use LDAP authorization with HPSS as a site-supported feature. If UNIX authorization will be used, this product is not required. LDAP authorization is supported on all platforms.

The OpenLDAP client API is the only supported API in HPSS.

If planning to use the LDAP option, make sure you get the LDAP RPM set, or use the LDAP authentication provided by PAM via HPSS UNIX authentication. For OpenLDAP, set SASL_INSTALL, SSL_INSTALL, and LDAP_INSTALL_PATH, to the locations of their respective packages, and set LDAP_TYPE to OPENLDAP.

5.2.7. Install DB2 and set up permanent license

DB2 is provided as part of HPSS, though it is a separate installation. The DB2 installation image and related license file can be found on the DB2 Installation CD or image. It can also be obtained by contacting HPSS support. Untar the DB2 installation package (if necessary) onto a local file system.

Use the below commands to install DB2.

```
% su -
% cd <location of DB2 install files>
% cd server
Note: Make sure all of db2 prerequisites have been installed
% ./db2prerequisitecheck
% ./db2_install
```

To create a permanent DB2 license, issue the following commands:

```
# cd /opt/ibm/db2/V10.5/adm

# ./db2licm -a <path name to the DB2 generic license file>
```

Refer to the *DB2 Command Reference* document for more information on how to use the **db2licm** utility to manage the DB2 license. i

5.3. Install HPSS with RPMs

This section details the steps necessary to install HPSS using RPMs.

HPSS RPMs can be obtained either from the HPSS Admin Wiki, or from the HPSS Collaboration website for registered HPSS customer sites. RPMs are the preferred method of installing HPSS software.

1. https://hpss-collaboration.clearlake.ibm.com/adminwiki/doku.php?id=start

2. http://hpss-collaboration.org

Installation of HPSS with RPMs is simple. RPM installation is done with:

```
rpm -ivh <rpm-name>
```

There are eleven main RPM packages for HPSS:

Table 5.1. RPM packages

Package	Description
hpss-lib	This is a dependency for all the other RPMs except for the hpss-doc, hpss-src, and hpss-stk-src RPMs. It contains a prerequisite set of binaries required to run HPSS such as shared object files and a static library. Also, it contains source include files and make files for running tools such as hpss_db2_bindall.ksh
hpss-lib-devel	This contains include files required for compiling an HPSS API program from source.

Package	Description
hpss-core	This contains executable files and dependent source files required for setting up and running an HPSS Core server Machine. Installing it also sets up a link to the default IBM DB2 installation.
hpss-mvr	This contains binary executable files and source configuration files required for running an HPSS Mover.
hpss-clnt	This contains binary executable files for running client applications such as pftp.
hpss-clnt-devel	This contains source include files for developing HPSS client applications.
hpss-clnt-devel-mvrprot	This contains confidential header files for developing applications that use the HPSS Mover protocol. *This RPM is only available at special request.*
hpss-pvr	This contains executable files and dependant source files required for running a remote PVR. Installing it also sets up a link to the default IBM DB2 installation.
hpss-doc	This contains HPSS documentation files in the form of HTML and PDF files for the conversion guide, error manual, install guide, management guide, programmer's reference, and the user's guide. It also contains the files required to access help menus in the HPSS GUI.
hpss-src	This contains all the HPSS source files with the exception of files in test directories and IBM confidential STK files. *This RPM is only available at special request.*
hpss-stk-src	This contains the IBM confidential STK source files. *This RPM is only available at special request.*

For a typical installation, you must install hpss-lib, which is a dependency for most of the other RPMs. You would then install hpss-core if you needed core binaries, hpss-mvr for a Mover, and hpss-clnt for client functionality. After hpss-lib is installed, you can mix and match functionality between hpss-core, hpss-mvr, and hpss-clnt.

The hpss-pvr has some binaries which conflict with hpss-core, so these two packages should not be installed on the same machine. However, you can install the hpss-pvr, hpss-mvr, and hpss-clnt packages on the same machine.

The full name of an RPM also contains the HPSS version as well as the machine architecture and operating system version built for. The RPMs for AIX are built with AIX 6.1 and are designed to work with AIX 7 also. If installing on RHEL Linux, install the RPMs containing the appropriate operating system version ("el6" for RHEL6 and "el7" for RHEL7) and the appropriate machine architecture (ppc64 for PPC Linux, ppc64-le for PPC Little Endian Linux or x86_64 for x86_64 Linux).

5.4. Install HPSS

This section details the steps necessary to install HPSS on the root, Movers, and other client nodes.

5.4.1. On core

To install HPSS on a Core Server machine, the RPMs hpss-lib and hpss-core should be installed and a link set up for `/opt/hpss`, for example:

```
# rpm -iv hpss-lib-X.Y.Z.0-0.el6.x86_64.rpm hpss-core-X.Y.Z.0-0.el6.x86_64.rpm
Preparing packages for installation...
hpss-lib-X.Y.Z.0-0.el6
Files for package hpss-lib installed under
/hpss_src/hpss-X.Y.Z.0-0.el6
hpss-core-X.Y.Z.0-0.el6
ln -sf /opt/ibm/db2/default /hpss_src/hpss-X.Y.Z.0-0.el6/db2
Files for package hpss-core installed under
/hpss_src/hpss-X.Y.Z.0-0.el6

# unlink /opt/hpss
# ln -s /hpss_src/hpss-X.Y.Z.0-0.el6 /opt/hpss
```

Note that the install sets up a link file under the HPSS installation directory to point to the default DB2 link.

Note that if you will be installing HPSS on a different machine as a Mover, client, or remote PVR, you should create a tar file of the `/var/hpss/etc` directory, for example:

```
# tar -cvf /tmp/var_hpss_etc.tar /var/hpss/etc
```

After doing this you should create a `/var/hpss/etc` directory on the non-core machine and untar the above tar file to them.

5.4.2. On Mover

To install HPSS on a Mover machine, the RPMs hpss-lib and hpss-mvr should be installed and a link set up for `/opt/hpss`, for example:

```
# rpm -iv hpss-lib-X.Y.Z.0-0.el6.x86_64.rpm hpss-mvr-X.Y.Z.0-0.el6.x86_64.rpm
Preparing packages for installation...
hpss-lib-X.Y.Z.0-0.el6
Files for package hpss-lib installed under
/hpss_src/hpss-X.Y.Z.0-0.el6
hpss-mvr-X.Y.Z.0-0.el6
Files for package hpss-mvr installed under
/hpss_src/hpss-X.Y.Z.0-0.el6

# unlink /opt/hpss
# ln -s /hpss_src/hpss-X.Y.Z.0-0.el6 /opt/hpss
```

5.4.3. On client

To install HPSS on a client machine, the RPMs hpss-lib and hpss-clnt should be installed and a link set up for `/opt/hpss`, for example:

```
# rpm -iv hpss-lib-X.Y.Z.0-0.el6.x86_64.rpm hpss-clnt-X.Y.Z.0-0.el6.x86_64.rpm
Preparing packages for installation...
hpss-lib-X.Y.Z.0-0.el6
Files for package hpss-lib installed under
```

```
/hpss_src/hpss-X.Y.Z.0-0.el6
hpss-clnt-X.Y.Z.0-0.el6
Files for package hpss-clnt installed under
/hpss_src/hpss-X.Y.Z.0-0.el6

# unlink /opt/hpss
# ln -s /hpss_src/hpss-X.Y.Z.0-0.el6 /opt/hpss
```

5.4.4. On remote PVR

To install HPSS on a remote PVR machine, the RPMs hpss-lib and hpss-pvr should be installed and a link set up for `/opt/hpss`, for example:

```
# rpm -iv hpss-lib-X.Y.Z.0-0.el6.x86_64.rpm hpss-pvr-X.Y.Z.0-0.el6.x86_64.rpm
Preparing packages for installation...
hpss-lib-X.Y.Z.0-0.el6
Files for package hpss-lib installed under
/hpss_src/hpss-X.Y.Z.0-0.el6
hpss-pvr-X.Y.Z.0-0.el6
ln -sf /opt/ibm/db2/default /hpss_src/hpss-X.Y.Z.0-0.el6/db2
Files for package hpss-pvr installed under
/hpss_src/hpss-X.Y.Z.0-0.el6

# unlink /opt/hpss
# ln -s /hpss_src/hpss-X.Y.Z.0-0.el6 /opt/hpss
```

5.4.5. Generate and bind the DB2 helper program

When filesets are created or updated, it is sometimes necessary to make entries in both the global and the subsystem database. When updating both of these databases, it is very important that the update be performed atomically. So, to accomplish an atomic update, a DB2 helper program is created. This DB2 helper program is *run* by DB2 whenever it needs to perform an atomic update of the global and subsystem databases when creating or updating a fileset.

To generate this DB2 helper program and bind it to DB2 a script named **hpss_db2_bindall.ksh** is provided.

Whenever the base source tree is rebuilt, perform the following steps to generate and bind the DB2 helper program:

1. Log on as *hpss*.
2. Change directory to `$HPSS_ROOT/bin`.
3. Run the following command:

   ```
   % hpss_db2_bindall.ksh
   ```

5.4.6. Update default DB2 link

HPSS can support multiple versions of DB2; see release notes for specific versions.

HPSS uses a default link for the DB2 installation path. This link is `/opt/ibm/db2/default` and should point to the DB2 installation path. Another link, `$HPSS_ROOT/db2`, should point to `/opt/ibm/`

`db2/default` by default. The link `$HPSS_ROOT/db2` can be modified if multiple HPSS installations with different DB2 requirements must coexist on the same machine.

When HPSS is installed with RPMs, the `$HPSS_ROOT/db2` — typically `/opt/hpss/db2` — link is set to point to `/opt/ibm/db2/default`. However, the administrator must ensure that `/opt/ibm/db2/default` points to the correct DB2 installation. DB2 is typically installed in `/opt/ibm/db2/Vx.y`; for example, `/opt/ibm/db2/V10.5`.

When updating or changing the DB2 version used, the administrator should update `/opt/ibm/db2/default` to point to the correct DB2 installation. In the case of a system with multiple installations with different DB2 requirements, the `$BUILD_TOP_ROOT/db2` link for the installation, which will not use the default DB2 version, should be changed to point to the correct DB2 version.

5.5. Configure HPSS infrastructure

The information in this section will describe how to use the **mkhpss** utility to configure HPSS and DB2. The HPSS installation and infrastructure configuration must be performed on the following HPSS machines:

- Root subsystem
- Subsystems created via "Install Subsystem" tab in **mkhpss**

Although the installation and configuration steps to be performed are similar, each step will be processed differently depending on the machine type. Therefore, it is very important that the correct submenu (Root Subsystem, Secondary Subsystem) is selected to perform the configuration. The following sections describe the procedure to be performed on each machine.

5.5.1. Navigating and general mkhpss behavior

Prior to using **mkhpss**, a quick word on navigation and general behaviors.

There are four window panels in **mkhpss**. At the bottom left, the keyboard shortcut window displays the **mkhpss** keyboard shortcuts. This is a static window that cannot be selected. At the top left, the menu window shows the various configuration screens that can be selected. At the top right, the content window shows the currently selected menu selection. At the bottom right, the message window displays information about configuration actions and results.

A box in the bottom left displays the keyboard shortcuts. They are:

Tab	Cycles to the next field. The content window often has multiple fields, and the Tab key will cycle through each field in the window.
Shift+Tab	Same as Tab, but cycles in reverse.
F4	Exit. This exits **mkhpss**.
F6	Toggle Active Window. This will change the focus of the active window, allowing scrolling or selection in that window. The keyboard shortcut window is never active.

Users can also navigate between panels and fields using the mouse.

When the *Configure* button is pressed, **mkhpss** will run a series of configuration scripts, whose output will go in the message window. During this time, inputs to **mkhpss** will be ignored.

The **mkhpss** utility logs the messages window to `/tmp/mkhpss.log`. Each time scripts are run via *Configure*, a timestamp and duration in seconds will be added to the file. This file will be overwritten by subsequent invocations of **mkhpss**.

5.5.2. Configure HPSS - root subsystem machine

The root subsystem machine is the machine upon which the root Core Server executes. Perform the following steps on that (root subsystem) machine:

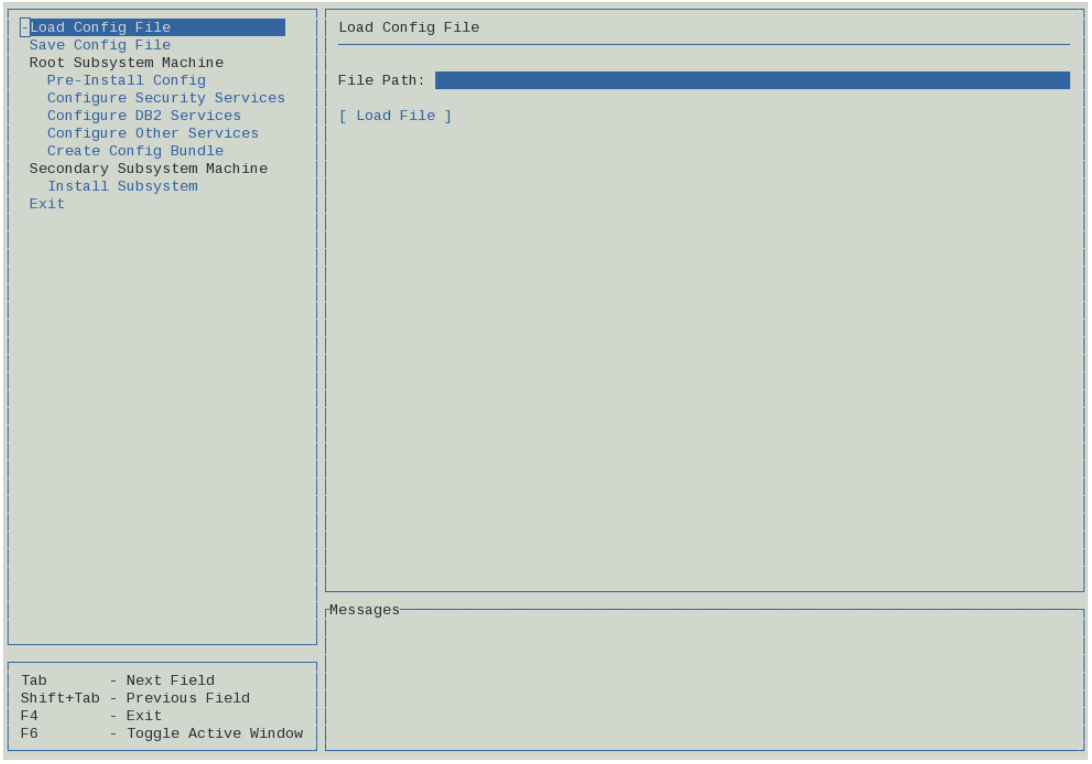
- *Section 5.5.2.1, "Pre-installation configuration"*
- *Section 5.5.2.2, "Configure HPSS security services"*
- *Section 5.5.2.3, "Configure DB2 services"*
- *Section 5.5.2.5, "Configure other services"*
- *Section 5.5.2.6, "Create configuration bundle"*

The following sections describe the procedures to perform these steps.

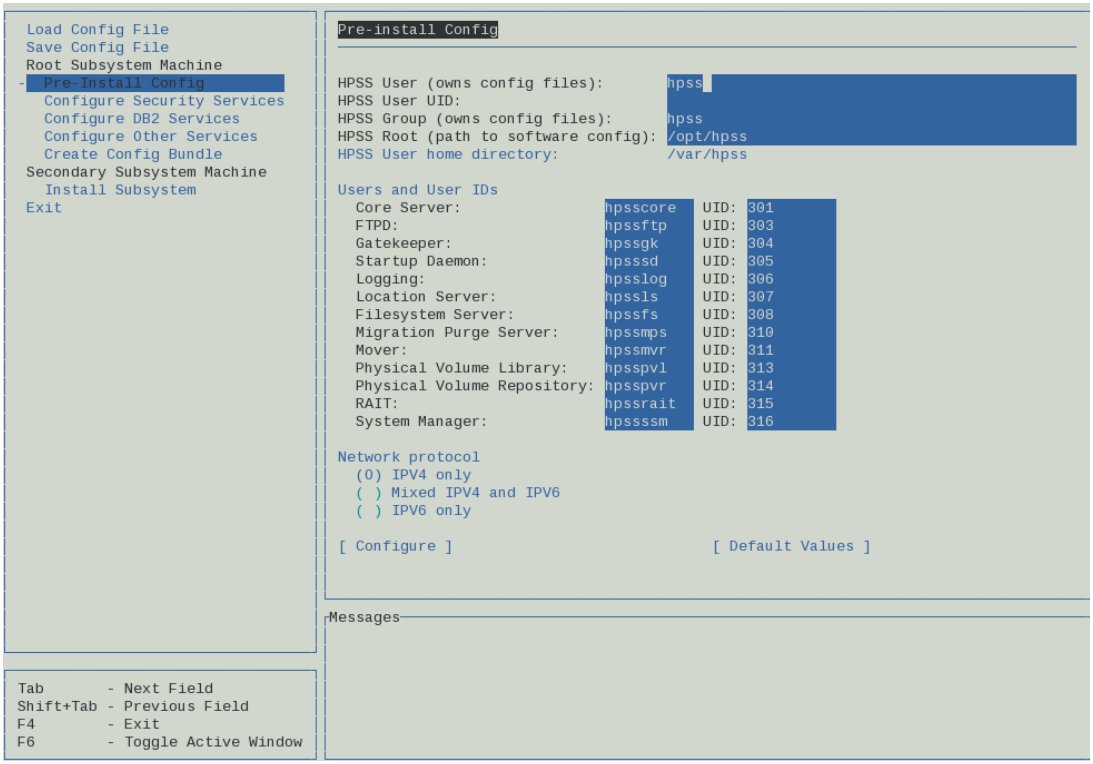
5.5.2.1. Pre-installation configuration

The **mkhpss** utility is used to set up HPSS user accounts, authentication, authorization, metadata resources, and initial system management configuration. It must be run prior to configuring HPSS software services and storage, but after other HPSS prereqs such as DB2 are installed.

1. The RPM install process will place **mkhpss** in the HPSS RPM `bin` directory, referred to as `$HPSS_ROOT/bin`. For example, this may be `/hpss_src/hpss-7.5.1.0-0.el7/`, in which case the **mkhpss** tool should be in `/hpss_src/hpss-7.5.1.0-0.el7/bin` directory. Invoke the **mkhpss** utility and you should see the following screen:



2. From the "Root Subsystem Machine" submenu, select the *Pre-Install Config* option in the content panel. **mkhpss** will display the following screen:



3. Verify that the default values are correct for the given installation and modify if necessary. Make sure that the server account UIDS in `/etc/passwd` or the local `/var/hpss/etc/passwd` (depending on authentication settings), either exist as the same UID or do not exist. Select the *Configure* button to perform the pre-installation setup. This will run a set of scripts to verify/create the *hpss* account and group, set up the `/var/hpss` directory with the required subdirectories and initialize the HPSS environment file, `env.conf` in `/var/hpss/etc`.

4. Check for any errors in the "Messages" window. When the execution is complete, the following line will display:

```
run command finished: mkhpss_run_preinstall
```

This will be followed by a timestamp and the duration of the command.

5.5.2.2. Configure HPSS security services

This section describes the authentication and authorization mechanisms which can be configured to enable HPSS to provide the desired security services. The following authentication and authorization combinations are supported:

Table 5.2. Supported authentication plus authorization methods

Authentication mechanism	Authorization mechanism
UNIX	UNIX
Kerberos	UNIX
Kerberos	LDAP

The following sections describe the procedure to configure the above security combinations using the **mkhpss** utility.

Configure UNIX authentication and UNIX authorization

From the "Root Subsystem Machine" submenu in the menu panel, select the "Configure Security Services" option. Perform the following steps using the content panel:

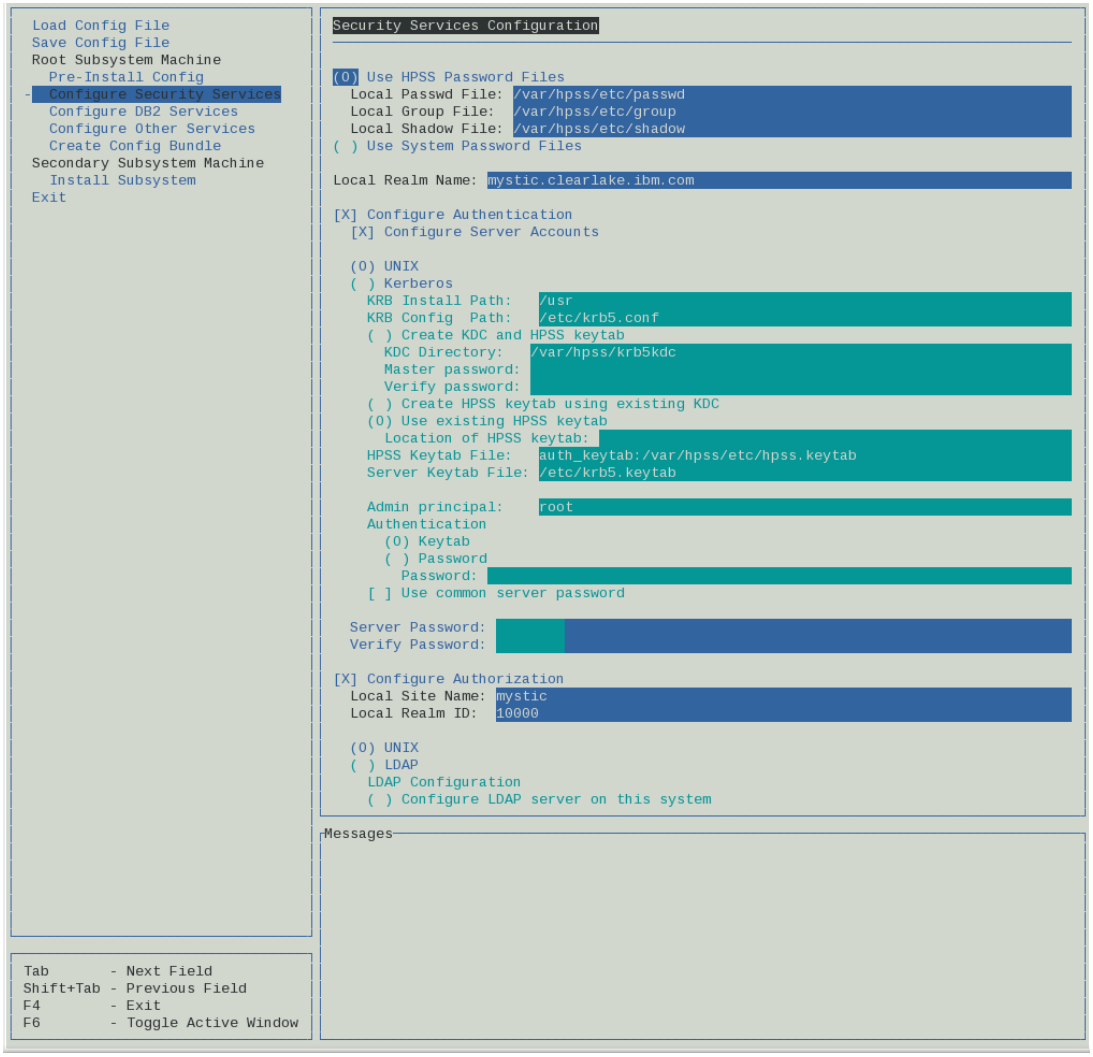
1. Select either local or system password files based on how the site wants accounts and passwords to be managed. By default, "Use HPSS Password Files" is selected with the HPSS configuration files set locally (`/var/hpss/etc/passwd`, `/var/hpss/etc/group`, and `/var/hpss/etc/shadow`) to administer the authentication and authorization services. As an option, "Use System Password Files" can be used instead (`/etc/passwd`, `/etc/group`, and `/etc/shadow`).

   Local Realm Name
   By convention, this value is usually set to the fully qualified domain name of the local host.

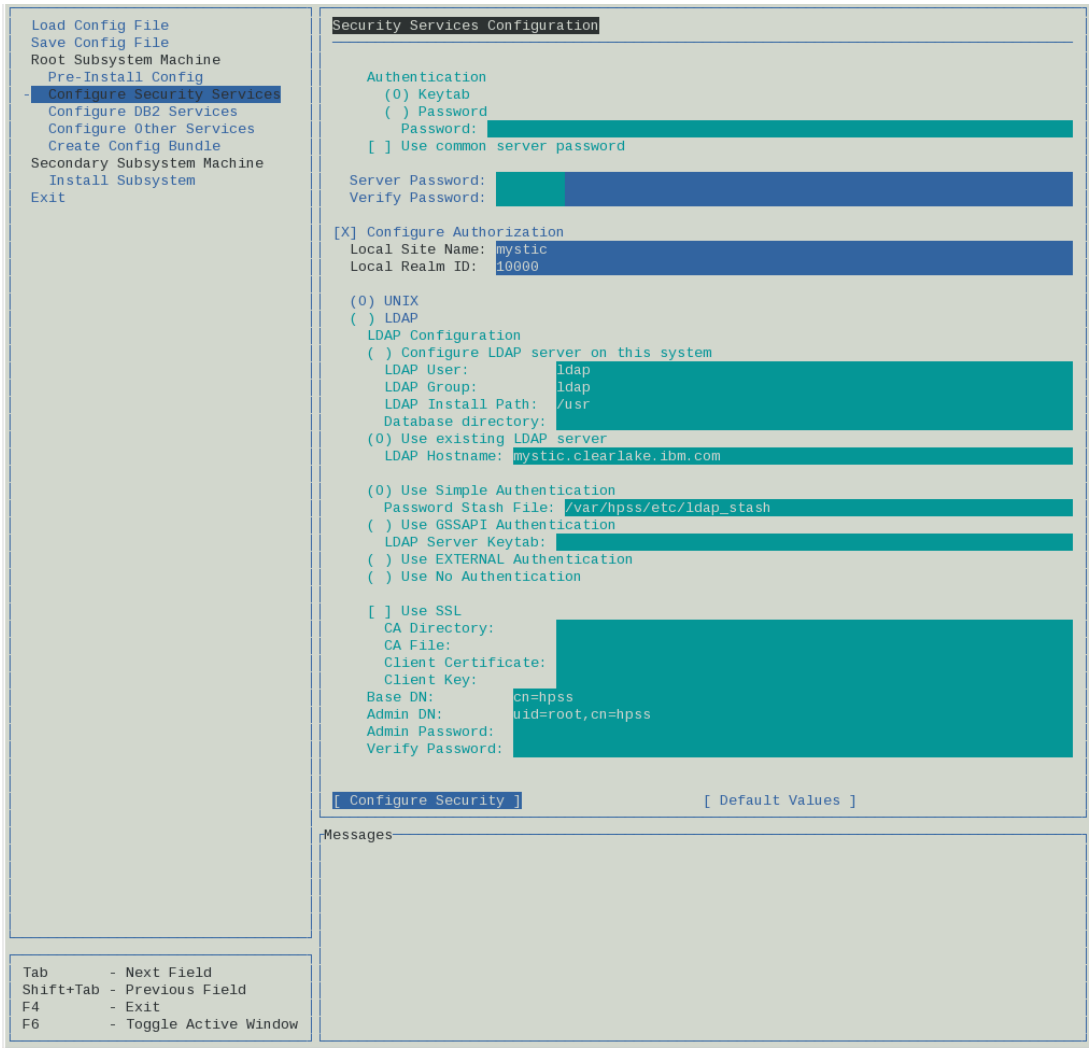
2. By default, the "Configure Authentication" check box is selected.

3. By default, the "Configure Server Accounts" and UNIX authentication check boxes are selected to create UNIX accounts for HPSS servers.

4. The fields to be set are as displayed on the screen shown:



5. Using the keyboard, scroll down through the content panel until the "Configure Authorization" fields are visible. It should look like the following:



6. By default, the "Configure Authorization" check box is selected.

7. Review and modify (if necessary) the active fields:

   Local Site Name
   The value is usually set to the fully qualified host name which can be determined using the **hostname** and **domainname** commands.

   Local Realm ID
   The field is set to a unique ID number for each site. Ask HPSS support for your site's value.

8. By default, "Authorization Service" is set to "UNIX".

9. Click on the "Configure Security" button at the bottom of the screen to perform the specified security configuration.

10. Check for any errors in the "Messages" window. When the execution is complete, the following line will display:

```
run command finished: mkhpss_run_core_sec_config
```

This will be followed by a timestamp and the duration of the command.

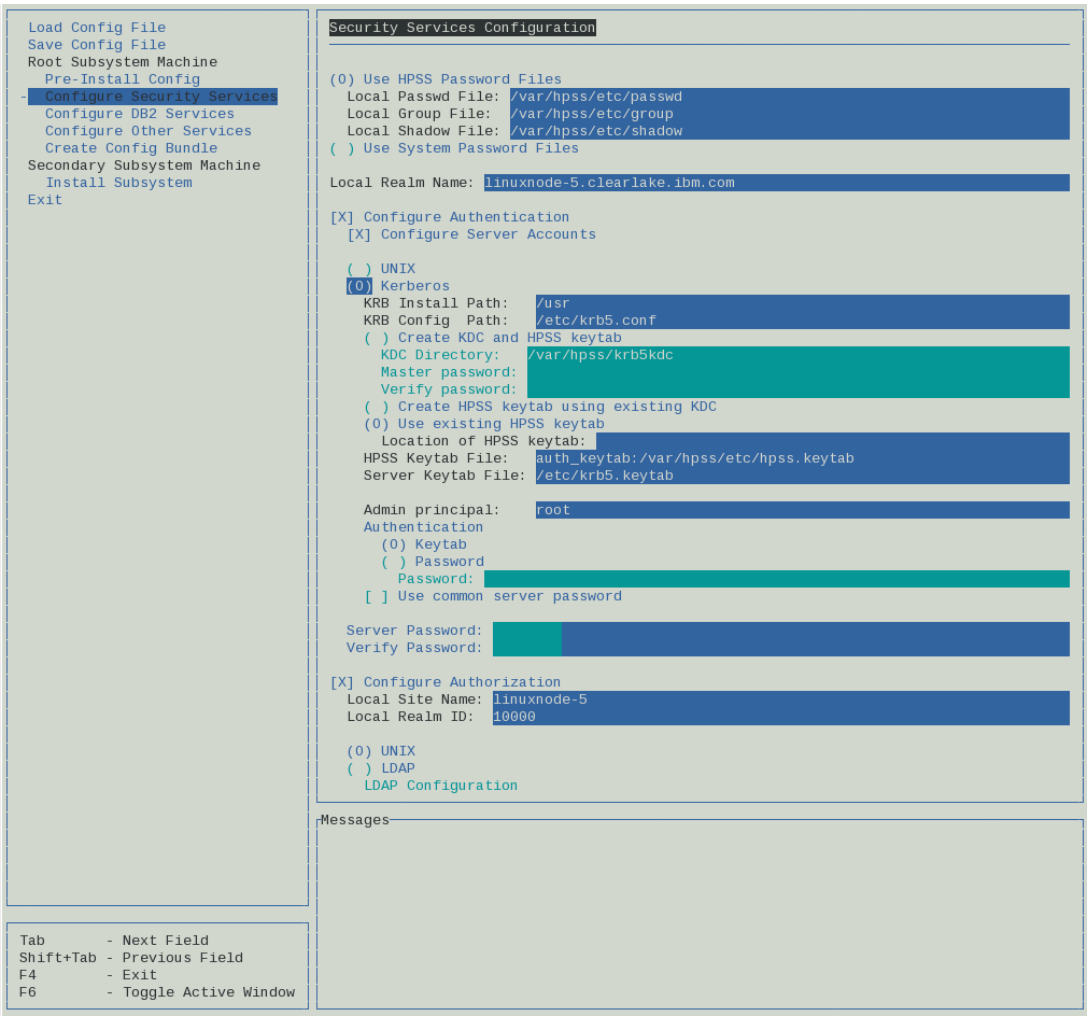
Configure Kerberos authentication and UNIX authorization

From the "Root Subsystem Machine" submenu in the menu panel, select the "Configure Security Services" option. Perform the following steps using . By default, the "Configure Authentication" check box is set.

1. Keep the "Configure Server Accounts" check box selected, but change the default setting from UNIX to Kerberos to create accounts for HPSS servers.

2. Select either local or system password files based on how the site wants accounts and passwords to be managed.

3. The fields to be configured are displayed on the screen shown:



4. Review and modify (if necessary) the following authentication fields:

Use HPSS Password Files

By default, selected with the HPSS configuration files set locally (`/var/hpss/etc/passwd`, `/var/hpss/etc/group`, and `/var/hpss/etc/shadow`) to administer the authentication and authorization services. As an option, "Use System Password Files" can be used instead (`/etc/passwd`, `/etc/group`, and `/etc/shadow`). Other HPSS utilities are available to administer

these HPSS configuration files. See the *Security Mechanisms* section of the *HPSS Management Guide* for more information.

Local Realm Name
By convention, this value is usually set to the Fully Qualified Domain Name of the local host. If you are not using **mkhpss** to create the Kerberos Key Distribution Center (*Create KDC and HPSS keytab* is not checked), then the Local Realm Name must be set to the name of the Kerberos Realm that will be used by HPSS. If *Create KDC and HPSS keytab* is checked, then the Kerberos authentication system will be created using the *Local Realm Name* as the Kerberos Realm Name.

5. By default, the "Configure Authentication" check box is set.

6. Keep the "Configure Server Accounts" check box selected, but change the default setting from UNIX to Kerberos to create accounts for HPSS servers.

Kerberos Install Path
The path to where Kerberos is installed. The default directory for Red Hat Linux is `/usr`.

Kerberos Config Path
The path to the Kerberos config file. The default path is set to `/etc/krb5.conf`.

Create KDC and HPSS keytab
If desired, **mkhpss** may be used to configure a Kerberos Key Distribution Center (KDC) on the Root Subsystem Machine. This includes creating a new Kerberos Principal database, establishing the Kerberos Realm and execution environment, as well as configuring HPSS startup and shutdown scripts to start and stop the Kerberos services. To create a KDC on the Root Subsystem Machine, select the "Create KDC and HPSS keytab" check box. If your site has an established Kerberos infrastructure which will be used for HPSS, or if you wish to create a KDC housed on a machine other than the Root Subsystem Machine, be sure that the "Create KDC and HPSS keytab" check box is not selected.

KDC Directory
The pathname of the KDC directory. This value is used when the "Create KDC and HPSS keytab" check box is checked. Default setting is `/var/hpss/krb5kdc`.

Master Password
The Kerberos administration password. This option is only available when *Create KDC and HPSS keytab* is checked.

Verify Password
Re-enter the Kerberos administration password. This option is only available when *Create KDC and HPSS keytab* is checked.

Be sure to remember this password to be able to administer the Kerberos environment later.

Create HPSS keytab using existing KDC
Creates an HPSS keytab using the existing KDC environment.

Use existing HPSS keytab

Use an existing HPSS keytab and an existing KDC environment. Enter the location of HPSS keytab.

HPSS Keytab File

Path to HPSS keytab file. Default setting is `auth_keytab:/var/hpss/etc/hpss.keytab`.

Server Keytab File

Path to server keytab file. Default is set to `/etc/krb5/keytab`.

Admin principal

The userid to administer the Kerberos environment.

Authentication

There are two supported options: Keytab File or Password. The Keytab File option allows HPSS servers or utilities to read a keytab file to authenticate. The Password option requires a password to be supplied each time an HPSS server or utility is invoked.

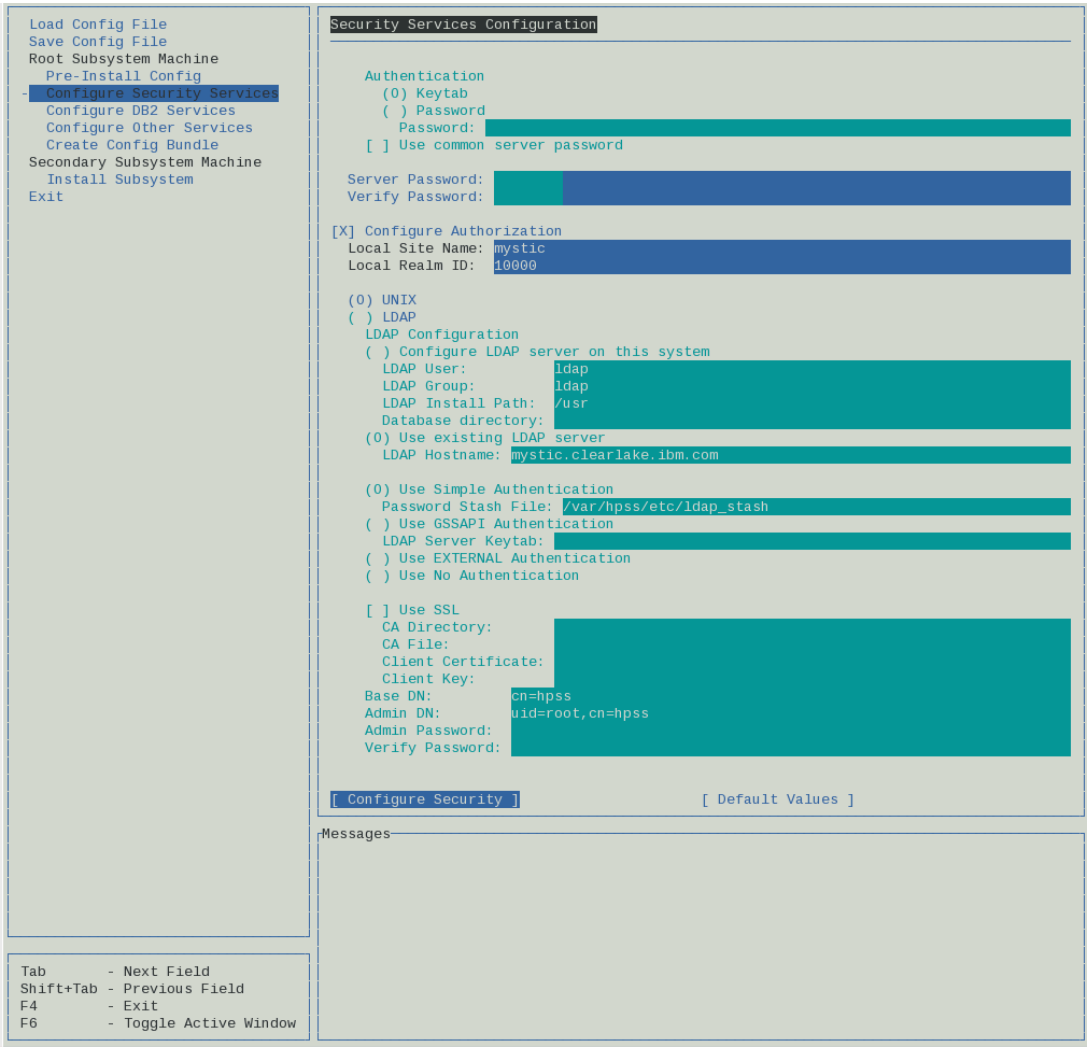
Keytab File

The pathname of the keytab file to be created if the Authentication Type is set to "Keytab File". This field is not enterable when the Authentication Type field is specified as Password.

Password

The password used to authenticate the caller when the HPSS server or utility is invoked. This field is not enterable when the Authentication Type field is specified as "Keytab File".

7. Using the keyboard, scroll down the content panel display until the "Authorization Service" information is seen. It should look like the following:



8. By default, the "Configure Authorization" check box is selected.

9. Review and modify (if necessary) the following authorization fields:

   Local Site Name
   The value is usually set to the fully qualified host name of the local host, which can be determined using the *hostname* and *domainname* commands.

   Local Realm ID
   The field is set to a unique ID number for each site. Ask HPSS support for your site's unique ID.

10. Keep the default setting for "Authorization Service" as "Unix".

11. Click on the "Configure Security Services" button at the bottom of the screen to perform the specified security configuration.

12. Check for any errors in the "Messages" window. When the execution is complete, the following line will display:

```
run command finished: mkhpss_run_core_sec_config
```

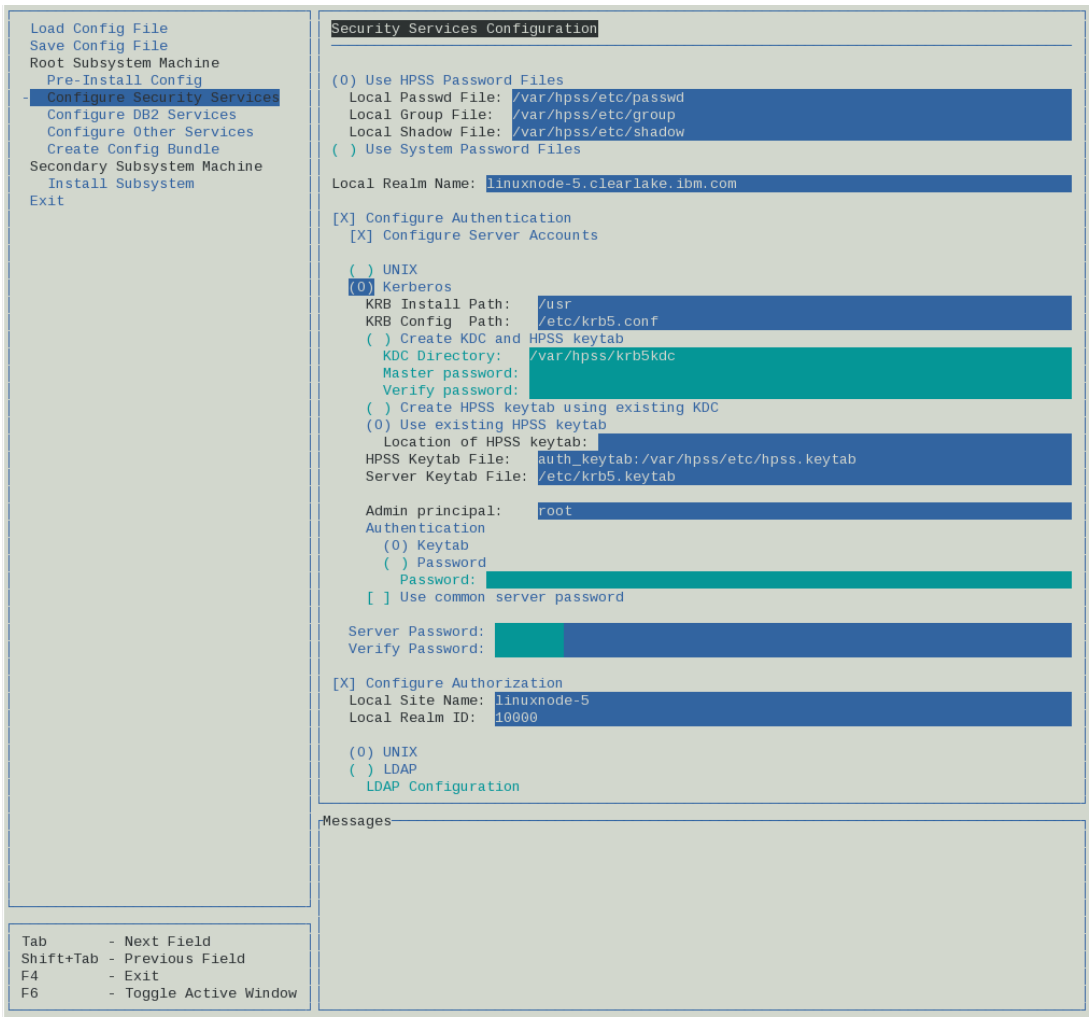
This will be followed by a timestamp and the duration of the command.

Configure Kerberos authentication and LDAP authorization

UNIX authentication is not supported when LDAP authorization is selected.

From the "Root Subsystem Machine" submenu in the menu panel, select the "Configure Security Services" icon. Perform the following steps using the content panel:

1. Select either local or system password files based on how the site wants accounts and passwords to be managed.
2. Select the "Configure Server Accounts" check box to create accounts for HPSS servers. Enter the password to use when creating the server accounts.
3. Select the "Configure Authentication" check box. Set the "Authentication Service" to "Kerberos".
4. The fields to be configured are as displayed on the screen shown:



5. Review and modify (if necessary) the following authentication fields:

Use HPSS Password Files

By default, the system's configuration files (`/etc/passwd`, `/etc/group`, and `/etc/shadow`) are used to administer the authentication and authorization services. As an option, HPSS

configuration files can be used instead. These files are created by **mkhpss** as part of this configuration step. Other HPSS utilities are available to administer these HPSS configuration files. See the *Security mechanisms* section of the *HPSS Management Guide* for more information. To use the HPSS configuration files, select the "Use HPSS Password Files" check box.

Local Realm Name

By convention, this value is usually set to the "Local Site Name" in upper-case letters. If you are not using **mkhpss** to create the Kerberos Key Distribution Center (*Create KDC and HPSS keytab* is not checked), then the Local Realm Name must be set to the name of the Kerberos Realm that will be used by HPSS. If *Create KDC and HPSS keytab* is checked, then the Kerberos authentication system will be created using the *Local Realm Name* as the Kerberos Realm Name.

Kerberos Install Path

The path to where Kerberos is installed. The default directory for RHEL is `/usr`.

Kerberos Config Path

The path to the Kerberos config file. Default path is `/etc/krb5.conf`.

Create KDC and HPSS keytab

If desired, **mkhpss** may be used to configure a Kerberos Key Distribution Center (KDC) on the Root Subsystem Machine. This includes creating a new Kerberos Principal database, establishing the Kerberos Realm and execution environment, as well as configuring HPSS startup and shutdown scripts to start and stop the Kerberos services. To create a KDC on the Root Subsystem Machine, select the *Create KDC and HPSS keytab* check box. If your site has an established Kerberos infrastructure which will be used for HPSS, or if you wish to create a KDC housed on a machine other than the Root Subsystem Machine, be sure that the *Create KDC and HPSS keytab* check box is not selected.

KDC Directory

The pathname of the KDC directory. This value is used when the "Create KDC and HPSS keytab" check box is checked. Default path is `/var/hpss/krb5kdc`.

Master Password

The Kerberos administration password. This option is only available when "Create KDC and HPSS keytab" is checked.

Be sure to remember this password to be able to administer the Kerberos environment later.

Verify Password

Re-enter the Kerberos administration password. This option is only available when *Create KDC and HPSS keytab* is checked.

Create HPSS keytab using existing KDC

Creates an HPSS keytab using the existing KDC environment.

Use existing HPSS keytab

Use an existing HPSS keytab and an existing KDC environment. Enter the location.

HPSS Keytab File
Path to HPSS keytab file. Default setting is `auth_keytab:/var/hpss/etc/hpss.keytab`.

Server Keytab File
Path to server keytab file. Default setting is `/etc/krb5/keytab`.

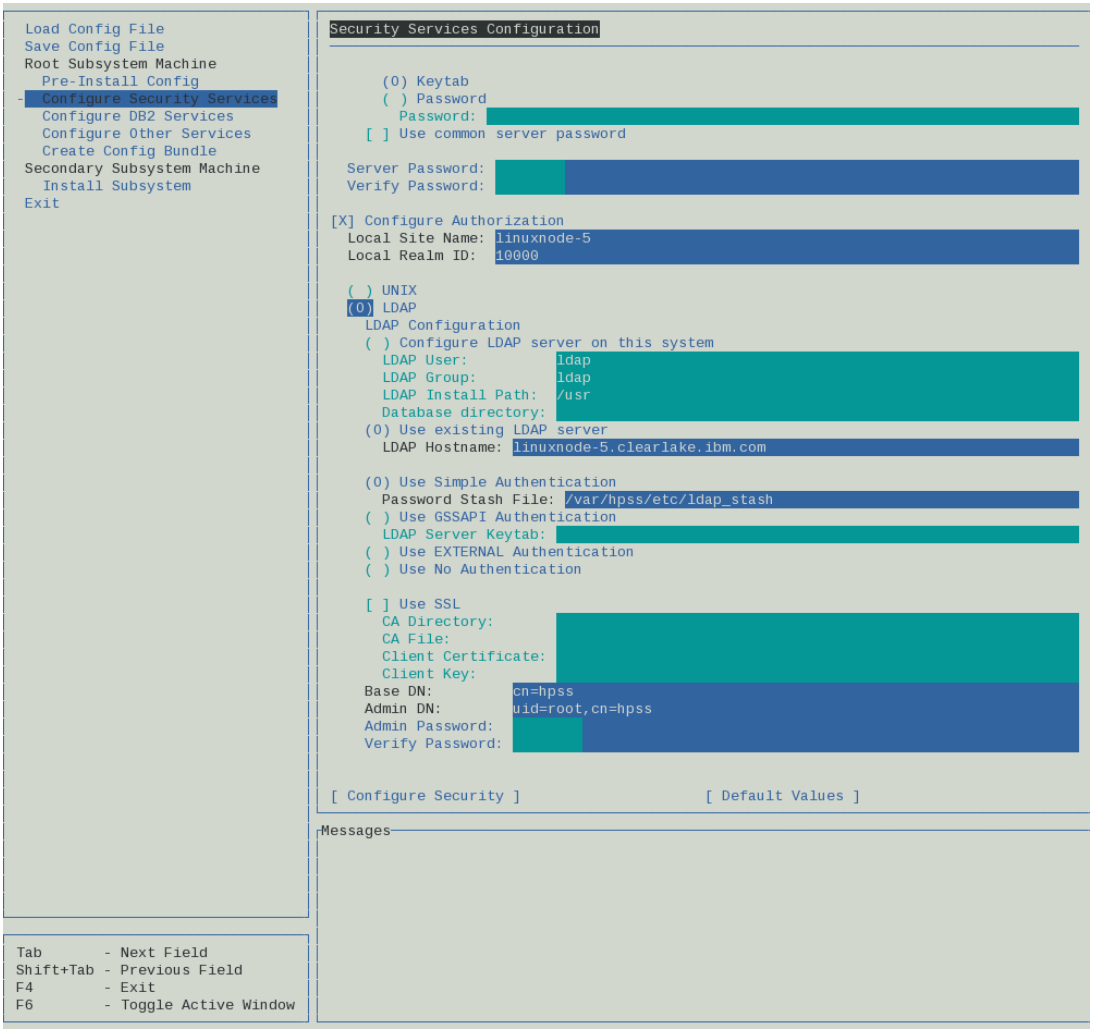
Admin principal
The userid to administer the Kerberos environment.

Authentication
There are two supported options: Keytab File or Password. The Keytab File option allows HPSS servers or utilities to read a keytab file to authenticate. The Password option requires a password to be supplied each time an HPSS server or utility is invoked.

Keytab File
The pathname of the keytab file to be created if the Authentication Type is set to "Keytab File". This field is not enterable when the Authentication Type field is specified as Password.

Password
The password used to authenticate the caller when the HPSS server or utility is invoked. This field is not enterable when the Authentication Type field is specified as "Keytab File".

6. Using the keyboard, scroll down the content panel display until the "Authorization Service" information is seen. It should look like the following:



7. By default, the "Configure Authorization" check box is selected.

8. Review and modify (if necessary) the following authorization fields:

   Local Site Name
   The value is usually set to the fully qualified host name of the local host, which can be determined using the **hostname** and **domainname** commands.

   Local Realm ID
   The field is set to a unique ID number for each site. Ask HPSS support for your site's unique ID.

9. Set the "Authorization Service" to "LDAP".

   Configure LDAP server in this system
   The flag is set to create an LDAP instance locally on this host machine. Tivoli Directory Server cannot run on the same host machine due to the incompatibility with DB2, and there is no support for setting up a database in OpenLDAP. Leave this box unchecked and set up the chosen database instance before configuring.

   LDAP User
   LDAP user name using the local UNIX UID that owns the LDAP server and database.

LDAP Group
LDAP Group name using the local UNIX GID that owns the LDAP server and database.

LDAP Install Path
The path is the installation prefix for OpenLDAP and its utilities. This is where it looks for ldapsearch, ldapadd, and other utilities.

Database directory
Location where the new database will be created.

Use existing LDAP server
Select this if you are using an existing database. This will deselect "Configure LDAP server on this system". The LDAP Hostname is the system that will be configured for all the servers to contact for the LDAP queries. Default will be the Fully Qualified Domain Name of the current host.

Authentication type (if one is used)
Simple, GSSAPI, EXTERNAL

Use Simple Authentication
Authentication via client DN and password.

Password Stash File
Path to the file where the username DN and password are stored for clients.

Use GSSAPI Authentication
Kerberos based and will need the LDAP Server Keytab for client use. Bind DN is derived from the authenticated user name. It takes the form "uid=principal@realm [mailto:principal@realm],cn=gssapi,cn=auth".

Use EXTERNAL Authentication
Typically through SASL, the bind DN is derived from the authentication that is performed externally. If connecting through a UNIX socket, it takes the form "gidNumber=<UNIX GID of connecting client>+uidNumber=<UID of client>,cn=peercred,cn=external,cn=auth". If connecting through SSL, the bind DN is the subject of the client's certificate if presented, or anonymous if no certificate is presented.

Use No Authentication
No authentication; client is anonymous. Anonymous access is not recommended.

Use SSL
Optionally, LDAP can be configured with SSL.

CA Directory
Path to certificate authority directory.

CA File
Certificate authority file.

Client Certificate
Path to client certificate file.

Client Key
Path to client key file.

Base DN (Distinguished Name)
Refers to the LDAP base distinguished name. Default value is cn=hpss.

Admin DN (Distinguished Name)
The administrator name that is allowed to add, update, and remove entries in LDAP. Default values are uid=root, cn=hpss.

Administrator Password
The password used by the administrator to manage entries in LDAP.

Verify Password
Repeat of the LDAP administrator password entered to verify it was entered correctly.

10. Click the "Configure Security" button at the bottom of the screen to perform the specified security configuration.

11. Check for any errors in the "Messages" window. When the execution is complete, the following line will display:

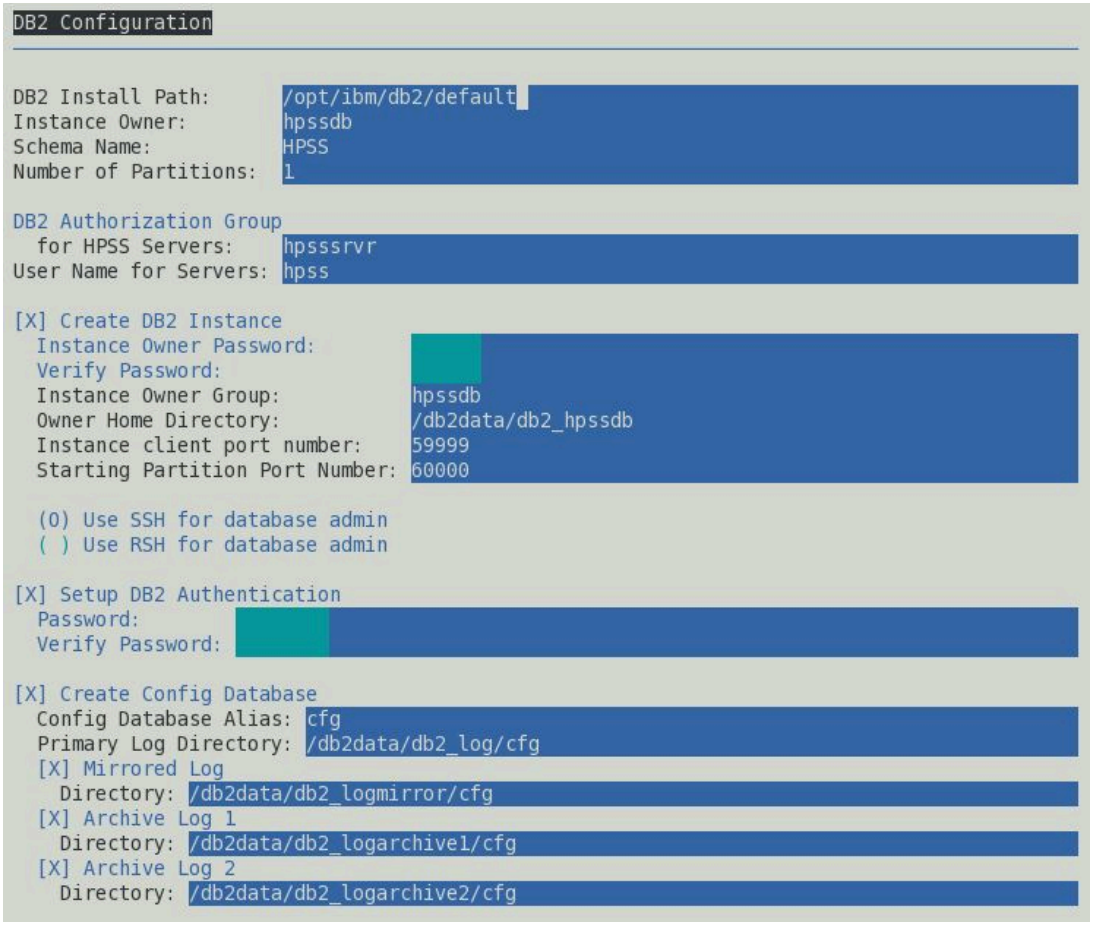
```
run command finished: mkhpss_run_core_sec_config
```

This will be followed by a timestamp and the duration of the command.

5.5.2.3. Configure DB2 services

To configure DB2 databases to manage HPSS metadata, perform the following steps:

1. From the "Root Subsystem Machine" submenu in the menu panel, select the "Configure DB2 Services" option. The following window will be shown:



2. Review and modify (if necessary) the following fields:

DB2 Install Path
The path where DB2 executables are installed. Default path is set to `/opt/hpss/db2`.

Instance Owner
The name of the DB2 instance owner. Normally set to *hpssdb*.

Schema Name
The name of the DB2 schema containing the HPSS metadata table. Normally set to *hpss*.

Number of Partitions
Determines the number of partitions the instance should have. The default setting is "1". Consult with the HPSS Systems Engineering and HPSS DB2 SME teams if your site is considered for number of partition greater than one. See HPSS Planning HPSS/DB2 File Systems chapter.

DB Authorization Group for HPSS Servers
The UNIX group ID that is used to allow the HPSS servers and utilities to access the HPSS DB2 instance. Use the default value of *hpsssrvr*.

User Name for Servers
The UNIX user name to be used for DB2 authentication. Use the default value of *hpss*.

Create DB2 Instance
By default, this check box is selected to create the DB2 instance.

Instance Owner Password and **Verify Password**
The UNIX password for the Instance Owner. Use the default prefilled hidden values.

Instance Owner Group
The UNIX group to which the Instance Owner belongs. Default value *hpssdb*.

Owner Home Directory
The directory where the HPSS DB2 instance configuration files are stored. Default directory is `/db2data/db2_hpssdb`.

Instance Client port number
The service port.

Starting Partition Port Number
The initial port number for partition communications. Additional partitions will use additional port numbers following this one. For example, if four partitions are to be configured, and the Starting Partition Port Number is 60000, then 60000-60003 will be used. The available port range should be checked to prevent potential collisions with other applications.

Use SSH or RSH admin
Select SSH or RSH for DB2 partition communications. RSH communications are unsecure, but can provide better performance. The selected mechanism should already be installed and running on the system prior to running configuration in this step. By default, SSH is selected.

Setup DB2 Authentication
By default, this check box is selected to set up DB2 authentication.

Password and **Verify Password**
The UNIX password used for DB2 authentication. Use the default prefilled hidden values.

Create Config Database
By default, this check box is selected to create the "cfg" database. If this is the initial instance of HPSS and the configuration *CFG* database has not been created, make sure the check box is selected.

If the CFG *database has already been created and you do not wish to destroy it, be sure to uncheck this box. Otherwise, it will be destroyed and re-created and any configuration information will be lost.*

Config Database Alias
The "real" database is configured as HCFG, while the database alias is the name used by HPSS servers and utilities to reference the database. Do not change the default value.

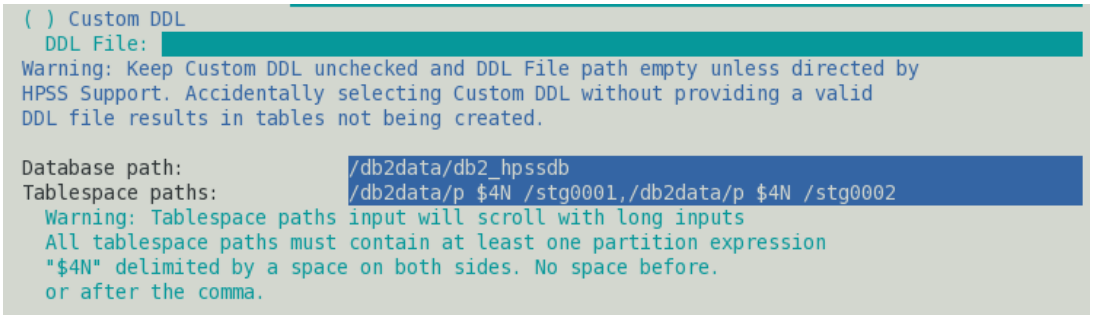
Primary Log Directory
The location of the cfg database log files. Default location is `/db2data/p0000/db2_log/cfg`. To change the location of the log files, enter the full pathname in the text box.

Mirrored Log
The location of the mirrored log files. It is highly recommended to configure a mirrored log directory. Used to protect information on two separate disks. Default log location is set to `/db2data/p0000/db2_logmirror/cfg`.

Archived Log 1
The location of the archived log files. It is highly recommended to configure an archived log directory. Used to archive files. Default location is set to `/db2data/p0000/db2_logarchive1/cfg`.

Archived Log 2
An alternate location of the archived log files. It is highly recommended to configure an archived log directory. Used to archive files. Default location is set to `/db2/data/p0000/db2_logarchive2/cfg`.

3. Continuation of DB2 Configuration fields are as displayed on the screen shown:



Custom DDL File
By default, *Custom DDL File* is unchecked with *DDL File* path left empty. This option should only be used under the guidance of HPSS support. This is a special option which can be used to customize the configuration of DB2 tablespaces, buffer pools, and tables. A valid DDL file must exist to utilize this option. IMPORTANT: *Keep default* Custom DDL File *unchecked and DDL File path empty unless directed by HPSS support. Accidentally selecting* Custom DDL File *without providing a valid DDL file results in tables not being created.*

Database path
Path to the database instance directory. Default path is `/db2data/db2_hpssdb`.

Tablespace paths
Path to database tablespace directories. Each tablespace path used must contain at least one partition expression "$4N" delimited by a space on both sides. No spaces before or after the comma if more than one path is used.
Eg /db2data/p $4N /stg0001,/db2data/p $4N /stg0002

All databases in HPSS are configured as automatic storage, whether or not the tablespaces associated with the database are automatic storage or not. Databases that are enabled for automatic storage have a set of one or more storage paths associated

with them. If using either automatic storage or DMS file containers, the tablespaces created to support HPSS will exist under these database paths.

Tablespace Paths use DB2 Partition Expressions [https://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.admin.partition.doc/doc/c0059241.html]. This should generally use the *$4N* format to spread partitions across available file system resources. *$4N* will put a directory, per partition, under the path specified. For example, */db2data/p $4N /stg0001* would create a directory per partition. If the *Number of Partitions* was set to "1" with one storage path file system, then DB2 would expect `/db2data/p0000/stg0001` to exist. If the *Number of Partitions* was set to "4" and the storage file system is 4, then the same expression will result with four directories that look like this: `/db2data/p0000/stg0001`, `/db2data/p0001/stg0002`, `/db2data/p0002/stg0003`, `/db2data/p0003/stg0004`.

Create Subsys Database

By default the check box to create "subsys" databases is selected.

If the SUBSYS *database has already been created and you do not wish to destroy it, be sure to uncheck this box. Otherwise, it will be destroyed and recreated and any existing data will be lost.*

Subsys Database Alias Base

The "real" database is configured as HSUBSYS1, while the database alias is the name used by HPSS servers and utilities to reference the database. The default value of *subsys* should normally be used. To support multiple subsystems, this is a *base* value. Each subsystem will have the subsystem number appended to the *base*; for example, if two subsystems were to be created, they would be subsys1 and subsys2. The *$S* symbol can be used to substitute the subsys name (such as subsys1) into a path expression.

Number of Subsystems
The number of subsys databases to configure. This should only be greater than one if the site plans to deploy multiple subsystems. For example, creating two subsystems would result in databases named something like subsys1 and subsys2.

Primary Log Directory
The location of the subsys database log files. To change the location of the log files, enter the full pathname in the text box. The directory `/db2data/p0000/db2_log/ $S` is the default setting where $S represents subsys1 based on "Number of Subsystems" equal to 1.

Mirrored Log
The location of the mirrored log files. It is highly recommended to configure a mirrored log directory. Used to protect information on two separate disks. The directory `/db2data/p0000/db2_logmirror/ $S` is the default setting.

Archived Log 1
The location of the first archived log files. It is highly recommended to configure an archived log directory. Used to archive files. The directory `/db2data/p0000/db2_logarchive1/ $S` is the default setting.

Archived Log 2
The location of the second archived log files. It is highly recommended to configure an archived log directory. The directory `/db2data/p0000/db2_logarchive2/ $S` is the default setting.

Extent Size
Used to configure the tablespace extent size.

Tablespace Type
There are three possible options in which to configure the DB2 tablespaces: Automatic Storage, Database Managed (DMS) file containers, and custom Database Definition Language (DDL) scripts.

Automatic Storage
Tablespaces will be created in the storage paths specified during database creation. DB2 manages the container and space allocation for the tablespaces as they are created and populated.

DMS file containers
This configuration works similarly to automatic storage, except that the administrator has the ability to extend the file containers manually. The tablespaces will be created in the storage paths specified during database creation.

Custom DDL File
By default, *Custom DDL File* is unchecked with the *DDL File* path left empty. This option should only be used under the guidance of HPSS support. This is a special option which can be used to customize the configuration of DB2 tablespaces, buffer pools, and tables. A valid DDL file must exist to utilize this option.

Keep default Custom DDL File *unchecked and* DDL File *path empty unless directed by HPSS support. Accidentally selecting* Custom DDL File *without providing a valid DDL file results in HPSS tables not being created.*

Database path
Path to the database instance directory.

Tablespace paths
Path to database tablespace directories.

All databases in HPSS are configured as automatic storage, whether or not the tablespaces associated with the database are automatic storage or not. Databases that are enabled for automatic storage have a set of one or more storage paths associated with them. If using either automatic storage or DMS file containers, the tablespaces created to support HPSS will exist under these database paths.

Tablespace Paths use DB2 Partition Expressions [https://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.admin.partition.doc/doc/c0059241.html]. This should generally use the *$4N* format to spread partitions across available file system resources. *$4N* will put a directory, per partition, under the path specified. For example, `/db2data/p $4N /stg0001` would create a directory per partition. If the *Number of Partitions* was set to "1", then DB2 would expect `/db2data/p0000/stg0001` to exist. If the *Number of Partitions* was set to "4", then the same expression will result with four directories that look like this: `/db2data/p0000/stg0001`, `/db2data/p0001/stg0002`, `/db2data/p0002/stg0003`, `/db2data/p0003/stg0004`.

If using multiple partitions in DB2, make sure to review rsh/ssh configurations that may restrict the connection of the DB2 Instance Owner. For example, if the system has an AllowedUsers section in the file `/etc/ssh/sshd_config`, the DB2 Instance Owner must be listed. When creating partitions directories, make sure it is owned by the DB2 Instance Owner and make sure nothing is inside of it that may interfere with DB2. The $S is used only by **mkhpss** and it means "substitute the subsystem name". It is the only allowable variable for the log file name and is only allowed for the subsystem log file name.

Check for any errors in the "Messages" window. When the execution is complete, the following line will display:

```
run command finished: mkhpss_run_core_db2_config
```

This will be followed by a timestamp and the duration of the command.

5.5.2.4. *Setting up off-node DB2*

Beginning with version 7.5.2, HPSS supports off-node DB2. This allows sites to scale the DB2 workload independently of the HPSS Core Server system. The **mkhpss** tool does not currently support setting up DB2 in this manner. This section details how the DB2 off-node configuration can be set up.

1. Install DB2 on *all* hosts that will run DB2 (call this the DB2 cluster).

   Note that database partition 0 should always be set up to run on the same host as the HPSS Core Server.

```
# As the ROOT user
> cd /path/to/DB2/FP8/universal
> ./db2_install
<snip>
************************************************************
Install into default directory (/opt/ibm/db2/<DB2 Version>) ? [yes/no]
yes


Specify one of the following keywords to install DB2 products.

  SERVER
  CONSV
  EXP
  CLIENT
  RTCL

Enter "help" to redisplay product names.

Enter "quit" to exit.

************************************************************
SERVER
************************************************************
Do you want to install the DB2 pureScale Feature? [yes/no]
no
<snip>
```

Where <DB2 Version> is in the following format: V<version>.<release>. For example:

- V10.5
- V11.1

2. Set up HPSS DB2 Symlink.

```
# As the ROOT user
> cd /opt/ibm/db2
> ln -s <DB2 Version> default
```

3. Install the necessary DB2 license files.

```
# For each license file, as the ROOT user
> /opt/ibm/db2/<DB2 Version>/adm/db2licm -a /path/to/<DB2 Version specific license fil
> /opt/ibm/db2/<DB2 Version>/adm/db2licm -l
```

4. Make sure that you have an NFS-mounted home directory for the UNIX user that will be your DB2 instance owner. Normally this is the hpssdb user with home directory `/db2data/db2_hpssdb`.

   This means that `/db2data/db2_hpssdb` must be shared across all the nodes that will be part of the DB2 cluster. Also, ensure that each machine in the cluster has root authority on the exported file system by using the "root" option. The rest of the instructions will assume that `/db2data/db2_hpssdb` is the instance owner's home directory and is mounted across all nodes in the DB2 cluster. You also need to ensure that all the relevant DB2 users db2fenc1/hpssdb have the same UID/GID across all the machines in the cluster as well.

5. Create the database instance.

```
# As the ROOT user
> cd /opt/ibm/db2/<DB2 Version>/instance
> ./db2icrt -u db2fenc1 hpssdb
```

6. Set up the db2nodes.cfg file.

   In the following example we have four DB2 hosts:

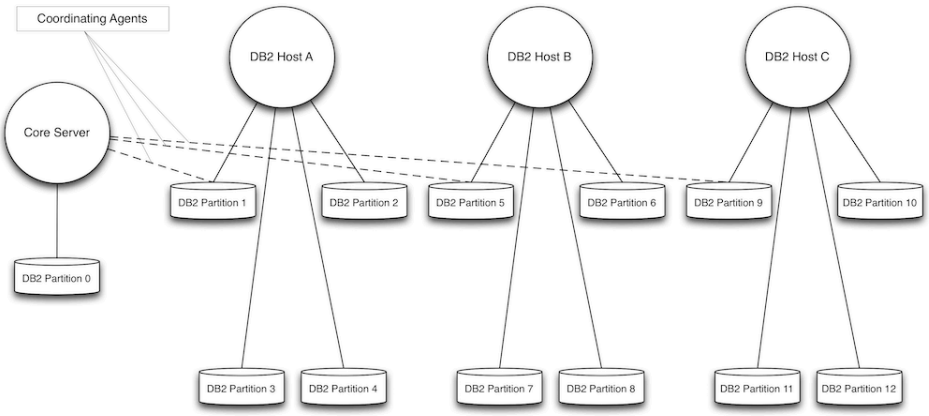
   hpsscore and *db2host[a-c]*

   (Note that the range string *db2host[a-c]* refers to three separate hosts *db2hosta*, *db2hostb*, and *db2hostc*. This document uses range strings extensively to express examples more compactly).

   hpsscore is where the HPSS Core Server will run and hosts DB2 partition 0.

   db2host[a-c] are the remote DB2 hosts each of which has four DB2 partitions for a total of thirteen database partitions.

Figure 5.1. DB2 Off-Node Example




```
> cat /db2data/db2_hpssdb/sqllib/db2nodes.cfg
0  hpsscore 0
1  db2hosta 0
2  db2hosta 1
3  db2hosta 2
4  db2hosta 3
5  db2hostb 0
6  db2hostb 1
7  db2hostb 2
8  db2hostb 3
9  db2hostc 0
10 db2hostc 1
11 db2hostc 2
12 db2hostc 3
```

7. Set up passwordless ssh between all hosts.

8. Ensure `/etc/services` has the same port numbers allocated to the database instance on each DB2 host. Add service name *db2c_hpssdb* with an assigned port to the `/etc/services` file on each

DB2 host as well. This will serve as the port DB2 will listen on for database connection requests. This service name is used to set the database manager SVCENAME parameter (see below….).

9. Create the database filesystems and directory structure.

   We recommend that the filesystems are formatted using the EXT4 filesystem.

   It is also desirable (but not required) to create the database filesystems on a solid-state disk. Ideally, at least the DB2 logs and mirror logs should reside on solid-state.

   Ensure that the instance owner can access all the paths in question.

   Continuing with our example, we set up the following directories on the specified hosts (note that we used paths stg[0001-0004] in the example, but any integer n could be chosen for the number of paths per partition):

```
hpsscore:
For DB2 data:
/db2data/p0000/dbpath/hcfg
/db2data/p0000/dbpath/hsubsys1
/db2data/p0000/stg[0001-0004]
/db2data/p0000/stg[0001-0004]
For DB2 primary logs:
/db2data/p0000/db2_log/hcfg
/db2data/p0000/db2_log/hsubsys1
For DB2 mirrorlogs:
/db2data/p0000/db2_logmirror/hcfg
/db2data/p0000/db2_logmirror/hsubsys1
For DB2 primary archive logs:
/db2data/p0000/db2_logarchive1
For DB2 secondary archive logs:
/db2data/p0000/db2_logarchive1
For DB2 database backup images:
/db2data/p0000/db2_backup1
/db2data/p0000/db2_backup2

db2hosta:
For DB2 data:
/db2data/p[0001-0004]/dbpath/hcfg
/db2data/p[0001-0004]/dbpath/hsubsys1
/db2data/p[0001-0004]/stg[0001-0004]

For DB2 primary logs:
/db2data/p[0001-0004]/db2_log/hcfg
/db2data/p[0001-0004]/db2_log/hsubsys1
For DB2 mirrorlogs:
/db2data/p[0001-0004]/db2_logmirror/hcfg
/db2data/p[0001-0004]/db2_logmirror/hsubsys1
For DB2 primary archive logs:
/db2data/p[0001-0004]/db2_logarchive1
For DB2 secondary archive logs:
/db2data/p[0001-0004]/db2_logarchive2
For DB2 database backup images:
/db2data/p[0001-0004]/db2_backup1
/db2data/p[0001-0004]/db2_backup2

db2hostb:
For DB2 data:
```

```
/db2data/p[0005-0008]/dbpath/hcfg
/db2data/p[0005-0008]/dbpath/hsubsys1
/db2data/p[0005-0008]/stg[0001-0004]
For DB2 primary logs:
/db2data/p[0005-0008]/db2_log/hcfg
/db2data/p[0005-0008]/db2_log/hsubsys1
For DB2 mirrorlogs:
/db2data/p[0005-0008]/db2_logmirror/hcfg
/db2data/p[0005-0008]/db2_logmirror/hsubsys1
For DB2 primary archive logs:
/db2data/p[0005-0008]/db2_logarchive1
For DB2 secondary archive logs:
/db2data/p[0005-0008]/db2_logarchive2
For DB2 database backup images:
/db2data/p[0005-0008]/db2_backup1
/db2data/p[0005-0008]/db2_backup2

db2hostc:
For DB2 data:
/db2data/p[0009-0012]/dbpath/hcfg
/db2data/p[0009-0012]/dbpath/hsubsys1
/db2data/p[0009-0012]/stg[0001-0004]
For DB2 primary logs:
/db2data/p[0009-0012]/db2_log/hcfg
/db2data/p[0009-0012]/db2_log/hsubsys1
For DB2 mirrorlogs:
/db2data/p[0009-0012]/db2_logmirror/hcfg
/db2data/p[0009-0012]/db2_logmirror/hsubsys1
For DB2 primary archive logs:
/db2data/p[0009-0012]/db2_logarchive1
For DB2 secondary archive logs:
/db2data/p[0009-0012]/db2_logarchive2
For DB2 database backup images:
/db2data/p[0009-0012]/db2_backup1
/db2data/p[0009-0012]/db2_backup2
```

10. Set the database manager parameters.

```
# As the database instance owner "hpssdb"
> db2 "update dbm cfg using AUTHENTICATION  SERVER_ENCRYPT"
> db2 "update dbm cfg using SVCENAME db2c_hpssdb"
> db2 "update dbm cfg using FEDERATED YES"
```

11. Set the DB2 registry variables.

```
# As the database instance owner "hpssdb"
> db2set DB2COMM=TCPIP
> db2set DB2RSHCMD=/usr/bin/ssh
```

12. Start the DB2 instance.

```
# As the database instance owner "hpssdb"
> db2start
```

13. Create the databases.

```
# As the database instance owner "hpssdb"

> db2 "create database hcfg automatic storage yes on \
      '/db2data/p \$4N /stg0001',                    \
```

```
      '/db2data/p \$4N /stg0002',                    \
      '/db2data/p \$4N /stg0003',                    \
      '/db2data/p \$4N /stg0004'                     \
      dbpath on '/db2data/p \$4N /dbpath/hcfg'       \
      pagesize 8192"
DB20000I  The CREATE DATABASE command completed successfully.

> db2 "create database hsubsys1 automatic storage yes on \
      '/db2data/p \$4N /stg0001',                        \
      '/db2data/p \$4N /stg0002',                        \
      '/db2data/p \$4N /stg0003',                        \
      '/db2data/p \$4N /stg0004'                         \
      dbpath on '/db2data/p \$4N /dbpath/hsubsys1'"
DB20000I  The CREATE DATABASE command completed successfully.
```

14. Make sure to set database parameters properly.

    Note the use of range strings below to express the parameters more compactly. They are not intended to be used literally. When executing the commands, make sure to run a command for each element in the range string.

```
# As the database instance owner "hpssdb"

db2 update db config for hcfg DBPARTITIONNUM [0-12] using NEWLOGPATH    /db2data/p[000
db2 update db config for hcfg DBPARTITIONNUM [0-12] using MIRRORLOGPATH /db2data/p[000
db2 update db config for hcfg DBPARTITIONNUM [0-12] using LOGARCHMETH1  DISK:/db2data/
db2 update db config for hcfg DBPARTITIONNUM [0-12] using LOGARCHMETH2  DISK:/db2data/

db2 update db config for hcfg using SELF_TUNING_MEM ON
db2 update db config for hcfg using LOGFILSIZ 25000
db2 update db config for hcfg using LOGPRIMARY 10
db2 update db config for hcfg using LOGSECOND -1
db2 update db config for hcfg using LOCKTIMEOUT 60
db2 update db config for hcfg using SOFTMAX 100
db2 update db config for hcfg using LOGBUFSZ 16384
db2 update db config for hcfg using DATABASE_MEMORY AUTOMATIC
db2 update db config for hcfg using MAXLOCKS AUTOMATIC
db2 update db config for hcfg using LOCKLIST AUTOMATIC
db2 update db config for hcfg using PCKCACHESZ AUTOMATIC
db2 update db config for hcfg using SORTHEAP AUTOMATIC
db2 update db config for hcfg using SHEAPTHRES_SHR AUTOMATIC

db2 update db config for hsubsys1 DBPARTITIONNUM 0      using SELF_TUNING_MEM OFF
db2 update db config for hsubsys1 DBPARTITIONNUM [1-12] using SELF_TUNING_MEM ON
db2 update db config for hsubsys1 DBPARTITIONNUM [0-12] using NEWLOGPATH    /db2data/p
db2 update db config for hsubsys1 DBPARTITIONNUM [0-12] using MIRRORLOGPATH /db2data/p
db2 update db config for hsubsys1 DBPARTITIONNUM [0-12] using LOGARCHMETH1  DISK:/db2d
db2 update db config for hsubsys1 DBPARTITIONNUM [0-12] using LOGARCHMETH2  DISK:/db2d

db2 update db config for hsubsys1 using LOGFILSIZ 25000
db2 update db config for hsubsys1 using LOGPRIMARY 10
db2 update db config for hsubsys1 using LOGSECOND 20
db2 update db config for hsubsys1 using LOCKTIMEOUT 60
db2 update db config for hsubsys1 using SOFTMAX 100
db2 update db config for hsubsys1 using LOGBUFSZ 16384
db2 update db config for hsubsys1 using DATABASE_MEMORY AUTOMATIC
db2 update db config for hsubsys1 using MAXLOCKS AUTOMATIC
db2 update db config for hsubsys1 using LOCKLIST AUTOMATIC
db2 update db config for hsubsys1 using PCKCACHESZ AUTOMATIC
```

```
db2 update db config for hsubsys1 using SORTHEAP AUTOMATIC
db2 update db config for hsubsys1 using SHEAPTHRES_SHR AUTOMATIC
```

15. Back up the databases. When database logging is altered from "circular" to "archival" (setting LOGARCHMETH1/LOGARCHMETH2), it is placed into "BACKUP PENDING" mode and must be backed up before activation is allowed.

```
# As the database instance owner "hpssdb"
> db2 "backup db hcfg on all dbpartitionnums to '/db2data/p \$4N /db2_backup1'"
> db2 "backup db hsubsys1 on all dbpartitionnums to '/db2data/p \$4N /db2_backup1'"
```

16. Activate the databases.

```
# As the database instance owner "hpssdb"
> db2 activate database hcfg
> db2 activate database hsubsys1
```

17. At this point the tablespaces, tables, and database permissions can all be set as before, when all database partitions were running on the same host as the Core Server.

Remote DB2 client access and fileset creation and deletion

This information is pertinent to sites that have chosen to deny remote client access to DB2. The method for configuring DB2 in such a manner is outside the scope of this document; refer to DB2 documentation and support for more information on such a configuration.

If, as part of configuring DB2 to deny remote access, a variable in the DB2 environment called DB2COMM has been unset, the creation and deletion of filesets will fail inside of DB2. You must have a variable named DB2_USE_LOCAL_RESYNC set to the value of *true* when starting DB2 in order for the aforementioned fileset operations to complete successfully:

In csh and tcsh:

```
setenv DB2_USE_LOCAL_RESYNC true
```

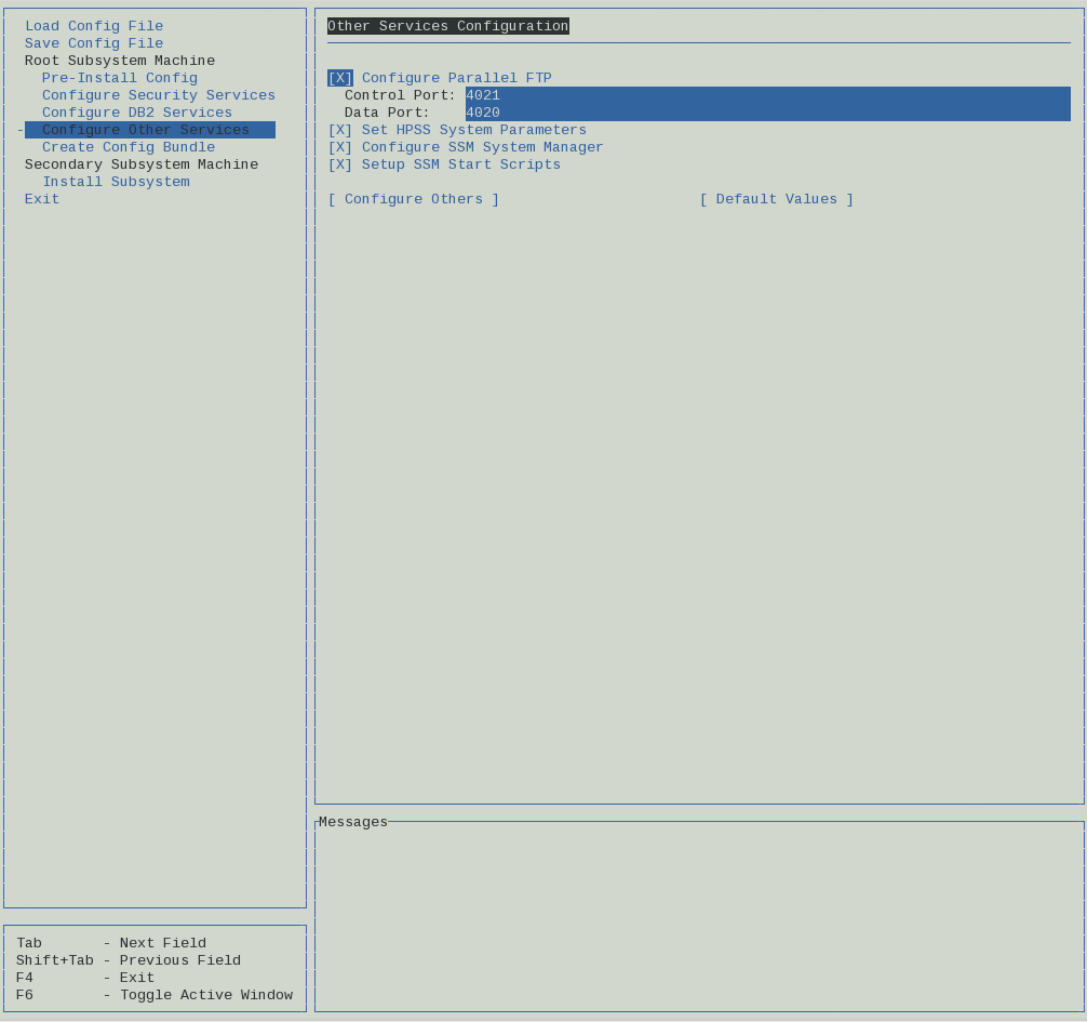
In sh and bash:

```
export DB2_USE_LOCAL_RESYNC=true
```

5.5.2.5. Configure other services

This menu configures various services such as Parallel FTP, HPSS System Parameters, SSM System Manager, and SSM Start Scripts. To configure other services, perform the following steps:

1. From the "Root Subsystem Machine" submenu in the menu panel, select the "Configure Other Services" option. The following window will be shown:



2. By default, all boxes are selected. If some items have already been configured, deselect the appropriate box to bypass the reconfiguration of that task. When satisfied with your selections, select the "Configure Others" button and verify the command completes. Depending upon the items selected, HPSS will 1) set up `/etc/services` and inetd to allow HPSS ftp/pftp daemon to be invoked on this system, 2) copy the SSM configuration template files to `/var/hpss/ssm`.

3. Check for any errors in the "Messages" window. When the execution is complete, the following line will display:

```
run command finished: mkhpss_run_core_misc_config
```

This will be followed by a timestamp and the duration of the command.

Failure to *configure other services* will result in SSM failing to start, unless it has already been configured.

1. After exiting **mkhpss**, verify permissions on the generated files. In particular, note the permissions on the keytab files. The `hpss.keytab` is used by HPSS servers to establish credentials. The `mm.keytab` is used by HPSS utility programs. The `kadm5.keytab` is used to establish credentials as the Kerberos admin. Be certain that the permissions on these files allow access only to the appropriate users.

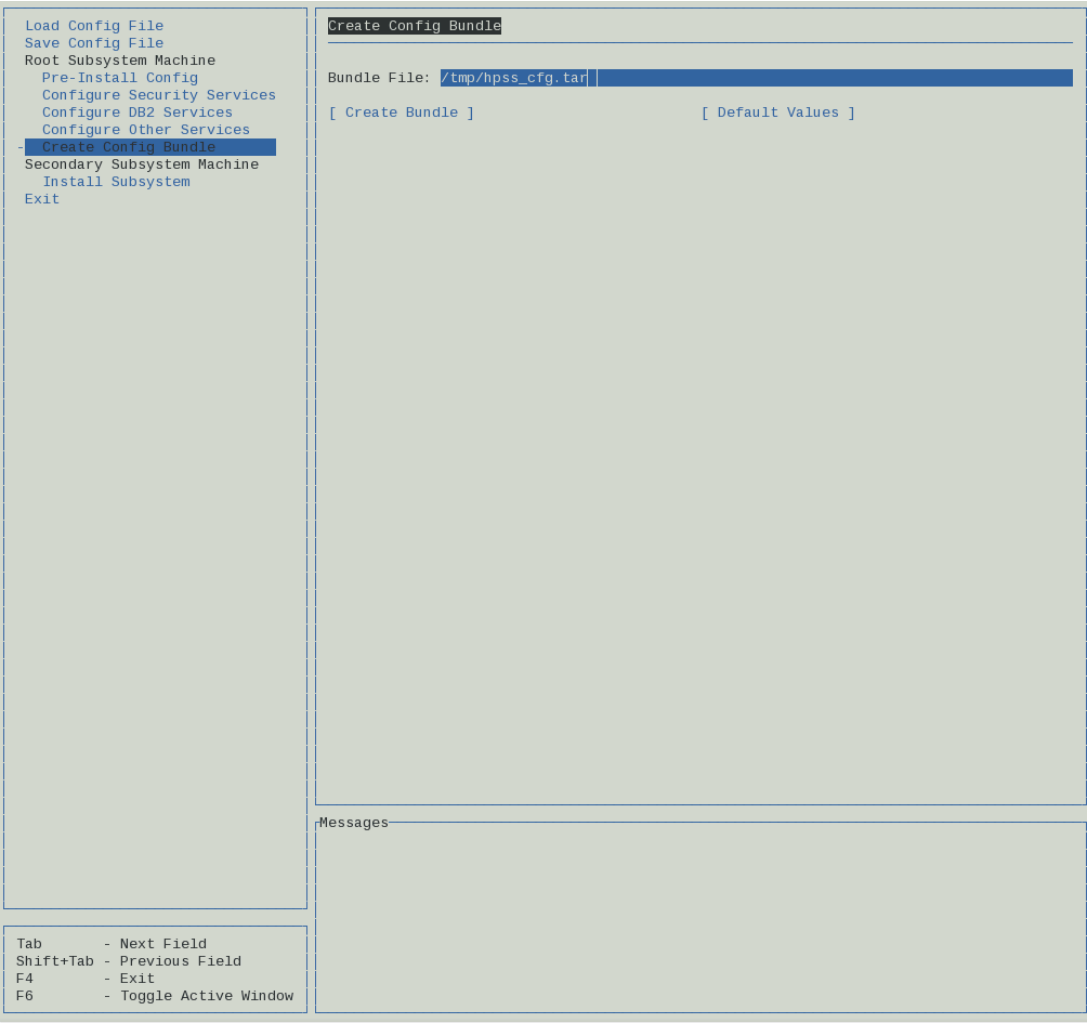
```
/var/hpss/etc/hpss.keytab
/var/hpss/etc/mm.keytab
/var/hpss/krb5kdc/kadm5.keytab
```

5.5.2.6. Create configuration bundle

To prepare the HPSS configuration information for distribution to other subsystem or Mover nodes, use the "Create Config Bundle" option on the **mkhpss** (may need to start **mkhpss** if exited previously). This option should be used *only* after the Location Server has been defined and configured by the administrator to include the EndPoint information in the `ep.conf` file. Otherwise, the configuration bundle will not contain the endpoint information and the file will need to be copied or transferred manually to the other nodes after the Location Server is configured.

To create the HPSS configuration bundle perform the following steps:

1. From the "Root Subsystem Machine" submenu in the menu panel, select the "Create Config Bundle" option. The following window will be shown:



2. Modify the "Bundle File" name, if desired.

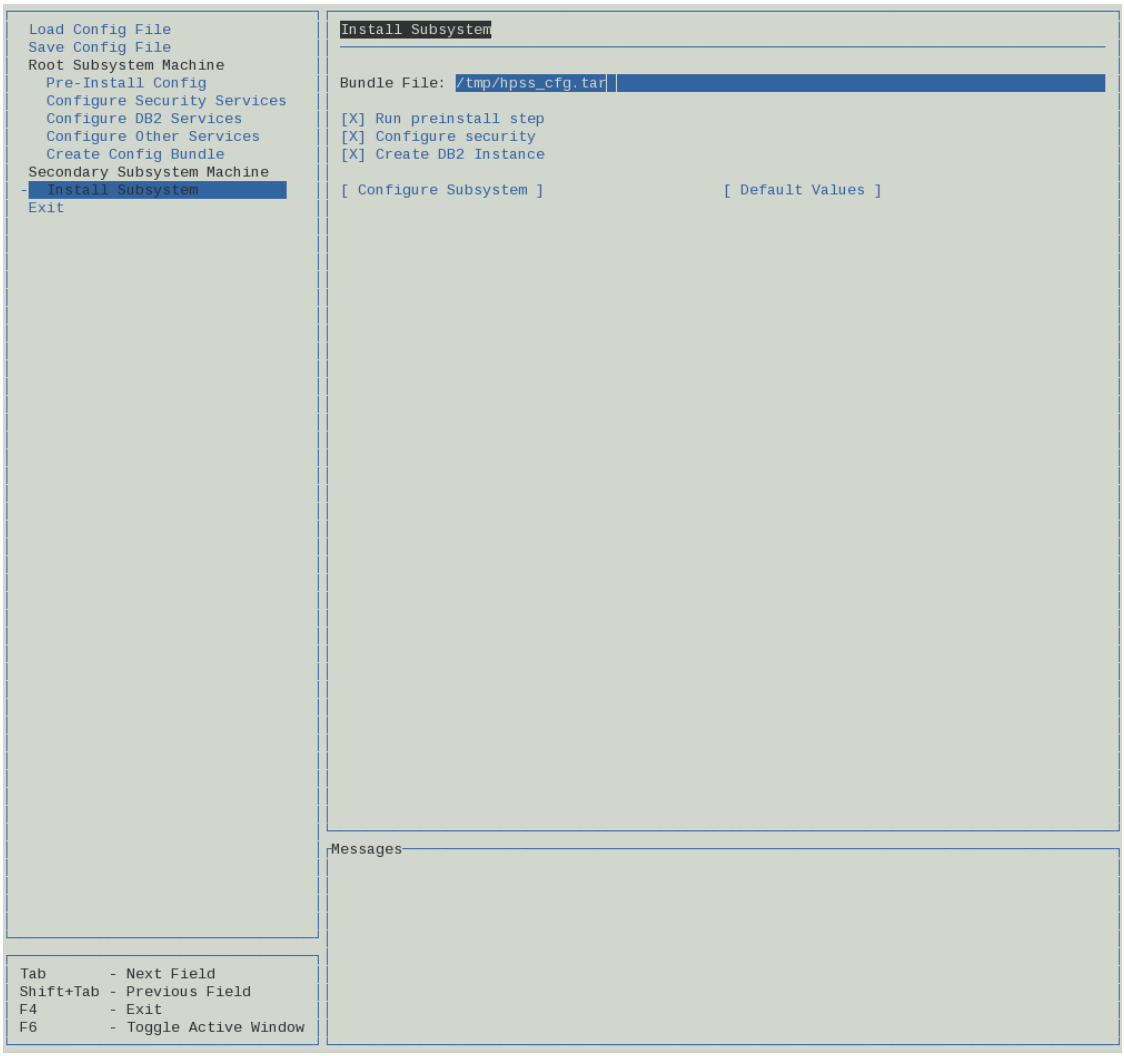
3. Check for any errors in the "Messages" window. When the execution is complete, the following line will display:

```
run command finished: mkhpss_run_core_create_bundle
```

This will be followed by a timestamp and the duration of the command.

5.5.3. Configure HPSS - secondary subsystem machine

A secondary subsystem machine is a node where a second Core Server is running. For the secondary subsystem machine, the following configuration steps must be performed. Installing the secondary subsystem requires the same procedure as installing the primary subsystem. So the user can just press on the Configure Subsystem button shown below:



5.5.4. Troubleshooting mkhpss

ERROR1: Before replacing an existing instance with a new one

1. Get rid of the previous User Home Directory (usually `/db2_hpssdb` or `/db2_data/db2_hpssdb`).
2. Delete the db2 instance owner from `/etc/passwd`. (You can check the DB2 instance owner by checking the name of the instance using db2ilist in `/opt/ibm/db2/V10.5`.)

3. Get rid of the existing process using **ps -ef** and grepping for the DB2 instance owner.

4. Get rid of any and all contents such as NODES, mount points, and log files in db2_log, db2_logmirror, db2_logarchives, and tablespaces.

5. Use `/opt/ibm/db2/V10.5/instance/db2idrop` to drop the previous instance.

ERROR2 : Updating an existing instance

1. Get rid of the database created in the database paths before running **mkhpss**.

2. There is no need to tamper with the logs. But if the old logs are needed make sure to specify new paths for logging, mirroring, and archiving.

ERROR3 : Services/port error

1. Clear the services that contain the DB2 Instance Owner in `/etc/services`.

ERROR4 : Enable Auto-Configuration Default

1. Get rid of the User Home Directory before running the DB2 portion of **mkhpss**.

2. Make sure **mkhpss** created the User Home Directory under the ownership of the DB2 Instance Owner.

5.6. Prepare post-installation procedures

After the HPSS software has been installed and its infrastructure has been configured, perform the following verifications:

1. Assuming default installation and configuration, verify the following directories have been created:

   `/opt/hpss/bin/`<HPSS binaries>

   `/opt/hpss/lib/`<HPSS libraries>

   `/opt/hpss/include/`<HPSS include and idl files>

   `/opt/hpss/msg/`<HPSS message catalog>

   `/opt/hpss/tools/`<HPSS tools>

   `/opt/hpss/man/`<HPSS man pages>

   `/opt/hpss/config/`<HPSS configuration scripts>

   `/opt/hpss/src/`<HPSS source files> *Included only if the hpss-src package is installed.*

   `/var/hpss/`<HPSS configuration files>

2. Verify that the HPSS file ownerships and file permissions are set as follows:

Executable files	`rwxr-xr-x bin bin`
Include files	`r--r--r-- bin bin` (write permission on generated files)
Library files	`r--r--r-- bin bin`
Source files	`r--r----- hpss hpss`
Make files	`rw-rw-r-- bin bin`
Configuration files	`rwxrwxr-x hpss hpss`

In particular, note that permissions on `/var/hpss/etc/mm.keytab` control the access to HPSS from many utility programs. Any user who can read `mm.keytab` will have permission to read and write directly into the HPSS database.

3. Verify that the DB2 permanent license has been set up by issuing the following commands:

```
% su -
% /opt/IBM/db2/V10.5/adm/db2licm -l
```

Refer to *Section 5.2.7, "Install DB2 and set up permanent license"* for more information on how to set up the DB2 license for an HPSS system.

5.7. Locate HPSS documentation and set up manual pages

This section describes the HPSS documentation provided to aid in the administration of the HPSS system as well as providing the basis for SSM help facility. It also describes the procedure to set up manual pages for HPSS utilities.

5.7.1. Documentation and SSM help package

The HPSS documentation is available via the HPSS website as a tar file and in PDF format. The HPSS documentation includes the following:

- *HPSS Installation Guide*
- *HPSS Management Guide*
- *HPSS Error Manual*
- *HPSS Programmer's Reference*
- *Programmer's Reference - I/O Supplement*
- *HPSS User's Guide*

The HTML version of the *HPSS Management Guide* also serves as the source for the SSM Help feature. These HTML files must be accessible from each host from which the SSM **hpssgui** program

is executed in order to be available in the SSM Help feature. The **hpssuser** program can be used to bundle the HTML files for delivery to the **hpssgui** host machine. The recommended installation location for the HTML files on each **hpssgui** host is `/hpss_src/hpss-<version>/ssmhelp` for AIX and Linux platforms and `c:\hpss\doc` for Windows platforms.

5.7.2. Manual pages setup

Perform one of the following steps to set up HPSS manual pages:

- Edit the `/etc/man_db.conf` file. Assuming that the HPSS tree is linked to `/opt/hpss`, add the following line in the section with other MANDATORY_MANPATH lines:

  ```
  MANDATORY_MANPATH /opt/hpss/man
  ```

- Edit a system file like `/etc/profile.d/hpss.sh` to ensure that the MANPATH environment variable includes the HPSS man pages directory. This can be done with the following:

  ```
  % su - root
  % vi /etc/profile.d/hpss.sh
  Add the following and save the file:
  if ! manpath | /bin/grep -q /opt/hpss/man; then
  export MANPATH="/opt/hpss/man:"
  fi
  ```

- Edit a system file like `/etc/profile.d/hpss.sh` to ensure that the PATH environment variable is set to include the `/opt/hpss/bin` directory. This can be done with the following:

  ```
  % su - root
  % vi /etc/profile.d/hpss.sh
  Add the following and save the file:
  if ! echo ${PATH} | /bin/grep -q /opt/hpss/bin; then
  PATH=/opt/hpss/bin:${PATH}
  fi
  ```

After one of the above is done, users who subsequently log in to the system should be able to view HPSS manual pages. The command **manpath** should show a directory containing `/opt/hpss/man`, and users should be able to view manual pages on HPSS commands or files. For example, to view information on **lshpss**, run the following:

```
% man lshpss
```

5.8. Define HPSS environment variables

While most, if not all HPSS environment variables can be used as defined by HPSS, they should be reviewed to ensure that they are set to reflect your environment. The HPSS environment variables and their default values are defined in `/opt/hpss/include/hpss_env_defs.h` file. See Appendix E: *Appendix E, hpss_env_defs.h* which lists the default values of all HPSS environment variables. These environment variables may be overridden in `/var/hpss/etc/env.conf` or in the system `/etc/environment` file.

The **/opt/hpss/config/hpss_set_env** utility can be used to insert environment variables into the `/var/hpss/etc/env.conf` file, list the values of the current HPSS environment variables, or list all HPSS default environment values:

Usage:

```
hpss_set_env [-all] [-def] [-set ENVNAME=NEWVALUE] |
                             ENVNAME [ENVNAME...]]
```

Where:

```
-all         Show current HPSS environment values
-def         Show all HPSS default environment values
-set         Set HPSS default environment values
ENVNAME      Display current value for ENVNAME
```

5.9. Set up a remote PVR

To set up a Remote PVR, the following steps must be completed:

1. **Initially configure the remote system**

   The remote system running the PVR should have the same authentication files available as the Core Server system. For example, if UNIX is being used with HPSS password files, the files in `/var/hpss/etc` should be copied over to the remote system. If UNIX password files are being used, care should be taken that the same passwords, users, and groups exist on the remote system.

   Further, the remote system should have all HPSS prerequisite software installed as described in the Release Notes.

2. **Install hpss-lib and hpss-pvr RPMs on the remote system**

   ```
   # rpm -ivh hpss-lib-X.Y.Z.0-0.el7.x86_64.rpm hpss-pvr-X.Y.Z.0-0.el7.x86_64.rpm
   ```

3. **Install and configure a Db2 client on the remote system**

   a. Create an "hpssdb" user

      Use the same hpssdb GID and UID from the core server.

      ```
      # mkdir /var/hpss
      # groupadd --gid <hpssdb gid> hpssdb
      # useradd --create-home --home <hpssdb homedir> --uid <hpssdb uid> \
        --gid <hpssdb gid> --shell /bin/bash hpssdb
      ```

   b. Create a Db2 instance

      ```
      # /opt/ibm/db2/vx.x/instance/db2icrt -a CLIENT -s client -u hpssdb hpssdb

      Where -a is authentication, -s is instance type, -u is for ID and
      instance name.
      ```

   c. Set DB2COMM

      Must be done as hpssdb.

      ```
      # su - hpssdb
      # db2set DB2COMM=tcpip
      ```

   d. Verify the local service in `/etc/services` for DB2 support

Copy the DB2 service entries from the Core Server `/etc/services`.

```
# Local services
db2c_hpssdb    59999/tcp
DB2_hpssdb     60000/tcp
DB2_hpssdb_1   60001/tcp
DB2_hpssdb_2   60002/tcp
DB2_hpssdb_END 60003/tcp
```

e. Catalog the tcpip node

```
# db2 catalog tcpip node CORSVR remote <core server hostname> server db2c_hpssdb
```

f. Catalog the database hcfg

```
db2 catalog db hcfg as cfg at node CORSVR
```

g. Verify that Db2 client can connect to the Db2 server on the HPSS core machine, list out the table for cfg/hcfg and verify the contents of a few known Db2 tables. The results should be identical to the HPSS Core Server.

```
  # db2 connect to cfg user hpssdb using hpssdb
  # db2 list tables for schema hpss
  # db2 "select * from hpss.COS"
```

4. **Configure logging**

   Configure logging using the `/opt/hpss/config/configure_logging` tool.

   ```
   /opt/hpss/config/configure_logging -s remote
   ```

5. **Configure security**

   Configure security using the `/opt/hpss/config/setup_pam.pm` tool. Alternatively, /etc/pam.d/hpss can be copied from the Core Server.

   ```
   /opt/hpss/config/setup_pam.pm
   [ setting up PAM config ]
   [ using local HPSS passwords ]
   ```

6. **Configure an HPSS Startup Daemon for the remote system**

   In SSM, configure a new HPSS Startup Daemon. The Execute Hostname field should be the remote system.

7. **Configure a SCSI PVR for the remote system**

   In SSM, configure a new SCSI PVR. The Execute Hostname field should be the remote system.

When you start HPSS, you will need to start the startup daemon by running the following command on the remote system:

```
/opt/hpss/bin/rc.hpss -d
```

You can then monitor and administer the remote SCSI PVR.

Once you have set up a single remote SCSI PVR, adding additional remote SCSI PVRs on the same remote system can be done via SSM with no additional setup required.

5.10. Tune DB2

Database tuning is an important aspect of maximizing HPSS performance, but it is a complex topic, requiring study and experimentation, to do well. **mkhpss** creates initial configuration parameters for HPSS DB2 instances that we find to be effective for most systems, but additional tuning to deal with the specifics of a given HPSS installation may improve database performance. Administrators wishing to learn more about DB2 tuning are referred to the *HPSS DB2 Tuning Guide*, available from HPSS support, the *DB2 Administration Guide: Performance*, available online from the IBM DB2 website, the IBM *"Database Fundamentals >> Performance tuning"* section under the DB2 V10.5 InfoCenter at http://publib.boulder.ibm.com/infocenter/db2luw/v10r5/index.jsp and the many after-market books available on this subject.

Be sure to contact HPSS support for advice before making changes to DB2 configuration parameters. See also the *Backup and recovery* chapter of the *HPSS Management Guide* for additional information.

5.11. Supporting both UNIX and Kerberos authentication for SSM

Once security services have been configured for your system (see Section *Section 5.5.2.2, "Configure HPSS security services"* for details), if both UNIX and Kerberos have been set up, it is possible to configure your system to support both UNIX and Kerberos authentication for the SSM. If you can log in to the system using the SSM GUI, you can use it to set this up, as described in the *Configuring the System Manager authentication for SSM clients* section of the *HPSS Management Guide*. If that is not an option because no available authentication mechanism is configured, you can use the procedure that follows to set up support for both authentication mechanisms. The combination of UNIX authentication with LDAP authorization is not supported at this time, so it only makes sense to do this if you are using UNIX authorization.

To set up support for both authentication mechanisms, change the following fields in DB2:

Table : Field	Old	New	Where
server : NUM_AUTH_MECHS	1	2	desc_name = 'SSM System Manager'
server : AUTHN_MECHS1_MECHANISM	0	2	
server : AUTHN_MECHS1_AUTH_TYPE_KEY	0	1	
serverinterfaces : AUTHN_MECH_SET_NUM_MECHS	1	2	server_id = (select server_id from server where desc_name = 'SSM System Manager') and descriptive_name = 'Administrative Client Interface'
serverinterfaces : AUTHN_MECH_SET_MECHS1	0	2	

This can be accomplished using the **db2** interactive utility. Here's a sample session showing the commands to use. You'll need to be logged in to UNIX as user *hpss* to have the needed permissions on the database.

```
$ db2
(c) Copyright IBM Corporation 1993,2002
Command Line Processor for DB2 SDK 8.2.5
...
For more detailed help, refer to the Online Reference Manual.

db2 => connect to hcfg

   Database Connection Information

 Database server = DB2/LINUX 8.2.5
 SQL authorization ID = HPSS
 Local database alias = HCFG

db2 => set schema hpss

db2 => select num_auth_mechs, authn_mechs1_mechanism,
authn_mechs1_auth_type_key
 from server where desc_name = 'SSM System Manager'

NUM_AUTH_MECHS AUTHN_MECHS1_MECHANISM AUTHN_MECHS1_AUTH_TYPE_KEY
-------------- ---------------------- --------------------------
             1                      0                          0

  1 record(s) selected.

db2 => select authn_mech_set_num_mechs, authn_mech_set_mechs1 from
serverinterfaces where server_id = (select server_id from server where
desc_name = 'SSM System Manager') and descriptive_name = 'Administrative
Client Interface'

AUTHN_MECH_SET_NUM_MECHS AUTHN_MECH_SET_MECHS1
------------------------ --------------------
                       1                    0

  1 record(s) selected.

db2 => update server set (num_auth_mechs, authn_mechs1_mechanism,
authn_mechs1_auth_type_key) = (2, 2, 1) where desc_name = 'SSM System
Manager'
DB20000I The SQL command completed successfully.

db2 => update serverinterfaces set (authn_mech_set_num_mechs,
authn_mech_set_mechs1) = (2, 2) where server_id = (select server_id from
server where desc_name = 'SSM System Manager') and descriptive_name =
'Administrative Client Interface'
DB20000I The SQL command completed successfully.

db2 => select num_auth_mechs, authn_mechs1_mechanism,
authn_mechs1_auth_type_key
 from server where desc_name = 'SSM System Manager'

NUM_AUTH_MECHS AUTHN_MECHS1_MECHANISM AUTHN_MECHS1_AUTH_TYPE_KEY
-------------- ---------------------- --------------------------
             2                      2                          1
```

```
  1 record(s) selected.

db2 => select authn_mech_set_num_mechs, authn_mech_set_mechs1 from
serverinterfaces where server_id = (select server_id from server where
desc_name = 'SSM System Manager') and descriptive_name = 'Administrative
Client Interface'

AUTHN_MECH_SET_NUM_MECHS AUTHN_MECH_SET_MECHS1
------------------------ --------------------
                       2                    2

  1 record(s) selected.

db2 => terminate
DB20000I The TERMINATE command completed successfully.
$
```

If you're using the local HPSS password file (HPSS_UNIX_USE_SYSTEM_COMMANDS=FALSE), you need to make sure it contains entries for users *root* and *hpss*.

Now you can use either security mechanism ("unix" or "krb5") in your SSM configuration file.

5.12. HPSS IPv6 support

The people that designed IPv4 never imagined that we would need more than 2^{32} IP addresses. Nearly every computing device made today has at least one network interface. When people began to realize that there were not enough IPv4 addresses for our needs, a new protocol, IPv6, was developed that would use a 128-bit addressing scheme. There are 3.4×10^{38} IPv6 addresses. As a comparison, there are roughly 9×10^{21} stars in the observable universe. The main difficulty is to get everyone out there buying new hardware that supports the new protocol. In February 2011, the last of the major IPv4 blocks were given out. This should be incentive enough to get people to begin using IPv6.

HPSS has been updated to allow usage of IPv6 networks. In order to determine what protocol HPSS will use, the system administrator must set the following in the `env.conf` file:

Table 5.3. Protocol settings

Environment variable	Value	Meaning
HPSS_NET_FAMILY	ipv4_only	Only allow use of IPv4 (this is the default).
HPSS_NET_FAMILY	ipv6	Allow usage of both IPv4 and IPv6. Note that if IPv6 is available, HPSS will prefer it over IPv4.
HPSS_NET_FAMILY	ipv6_only	Only allow use of IPv6.

5.12.1. Usage examples

HPSS_NET_FAMILY=ipv4_only.

This is the HPSS default. HPSS services will only allow the use of the IPv4 protocol.

```
> rpcinfo -s
   program version(s) netid(s)                         service     owner
 536870913  1         tcp                             -           superuser
 536870916  1         tcp                             -           superuser
 536870915  1         tcp                             -           superuser
 536870917  1         tcp                             -           superuser
 536870920  1         tcp                             -           superuser
 536870914  1         tcp                             -           superuser
 536870912  1         tcp                             -           superuser
 536870918  1         tcp                             -           superuser
 536870919  1         tcp                             -           superuser

> lsof -P -n | grep "hpss_core 22743 " | grep LISTEN
hpss_core 22743 root 14u IPv4 296030 0t0 TCP *:42580 (LISTEN)

> nc -v -4 10.0.2.15 42580
Connection to 10.0.2.15 42580 port [tcp/*] succeeded!

> lsof -P -n | grep "nc " | grep IP
nc 24120 root 3u IPv4 366364 0t0 TCP
10.0.2.15:39038->10.0.2.15:42580 (ESTABLISHED)

> lsof -P -n | grep "hpss_core 22743 " | grep 39038
hpss_core 22743 root 15u IPv4 366365 0t0 TCP
10.0.2.15:42580->10.0.2.15:39038 (ESTABLISHED)
```

HPSS_NET_FAMILY=ipv6.

HPSS services will prefer to use the IPv6 protocol, if available.

```
> rpcinfo -s
   program version(s) netid(s)                         service     owner
 536870913  1         tcp,tcp6                        -           superuser
 536870916  1         tcp,tcp6                        -           superuser
 536870915  1         tcp,tcp6                        -           superuser
 536870917  1         tcp,tcp6                        -           superuser
 536870920  1         tcp,tcp6                        -           superuser
 536870914  1         tcp,tcp6                        -           superuser
 536870912  1         tcp,tcp6                        -           superuser
 536870918  1         tcp,tcp6                        -           superuser
 536870919  1         tcp,tcp6                        -           superuser

> lsof -P -n | grep "hpss_core 12586" | grep LISTEN
hpss_core 12586 root 14u IPv6 163012 0t0 TCP *:35941 (LISTEN)

> nc -v -4 10.0.2.15 35941
Connection to 10.0.2.15 35941 port [tcp/*] succeeded!

> lsof -P -n | grep "nc " | grep IP
nc 21611 root 3u IPv4 285809 0t0 TCP
10.0.2.15:51191->10.0.2.15:35941 (ESTABLISHED)

> lsof -P -n | grep "hpss_core 12586 " | grep 51191
hpss_core 12586 root 15u IPv6 285810 0t0 TCP
10.0.2.15:35941->10.0.2.15:51191 (ESTABLISHED)

> nc -v -6 ::1 35941
```

```
Connection to ::1 35941 port [tcp/*] succeeded!

> lsof -P -n | grep "nc " | grep IP
nc 27208 root 3u IPv6 466915 0t0 TCP
[::1]:33645->[::1]:35941 (ESTABLISHED)

> lsof -P -n | grep "hpss_core 12586 " | grep 33645
hpss_core 12586 root 15u IPv6 466916 0t0 TCP
[::1]:35941->[::1]:33645 (ESTABLISHED)
```

HPSS_NET_FAMILY=ipv6_only.

HPSS services will only allow the use of the IPv6 protocol.

```
> rpcinfo -s
   program version(s) netid(s)                         service     owner
 536870913  1         tcp6                             -           superuser
 536870916  1         tcp6                             -           superuser
 536870915  1         tcp6                             -           superuser
 536870917  1         tcp6                             -           superuser
 536870920  1         tcp6                             -           superuser
 536870914  1         tcp6                             -           superuser
 536870912  1         tcp6                             -           superuser
 536870918  1         tcp6                             -           superuser
 536870919  1         tcp6                             -           superuser

> nc -v -4 10.0.2.15 57947
nc: connect to 10.0.2.15 port 57947 (tcp) failed: Connection refused
```

Chapter 6. Installation and configuration of the Elastic (ELK) Stack

6.1. Installing the ELK stack

This section explains how to install the various parts of the ELK stack. The parts are:

- Filebeat
- Logstash
- Elastic
- Kibana

The HPSS 10.1 ELK dashboards and templates, were developed using ELK version 7.15.0.

6.1.1. Install Filebeat

Download Filebeat from https://www.elastic.co/downloads/beats/filebeat.

On the right side, select "View past releases" and then version 7.15.0.

Use the Filebeat installation instructions found on the download page, with the following modifications:

- Install Filebeat on the HPSS Core Server
- Use the **filebeat-hpss.yml** provided in the HPSS RPM instead of the default **filebeat.yml**.
- Update the **filebeat-hpss.yml** to point to the correct Logstash

The provided Filebeat template assumes Logstash is installed on the same system. However, the best practice is to run Logstash on a separate system. Change the stanza below to contain the correct host for Logstash.

```
output.logstash:
  # The Logstash hosts
  hosts: ["localhost:5044"] # Update this if Logstash will be run on a different host
  enabled: true
```

It is recommended that you start Filebeat as a service using systemd. You can find Filebeat setup instructions at https://www.elastic.co/guide/en/beats/filebeat/current/running-with-systemd.html

If you need a sample services file, use the following:

```
[Unit]
Description=Filebeat

[Service]
User=root
```

```
WorkingDirectory=<directory where Filebeat was installed>
ExecStart=filebeat -c filebeat-hpss.yml
Restart=always

[Install]
WantedBy=multi-user.target
```

6.1.2. Install Logstash

Download Logstash from https://www.elastic.co/downloads/logstash

On the right side, select "View past releases" and then version 7.15.0.

Use the Logstash installation instructions found on the download page, with the following modifications:

- Use the **logstash-hpss.yml** instead of the default **logstash.yml**.

It is recommended that you start Logstash as a service using systemd. While there are no guides for setting up Logstash with systemd, it does not differ from other processes. You can find instructions on running Logstash at https://www.elastic.co/guide/en/logstash/current/setup-logstash.html

6.1.3. Install Elastic

Download Elastic from https://www.elastic.co/downloads/elasticsearch

On the right side, select "View past releases" and then version 7.15.0.

It is recommended that you do **NOT** run Elastic on the HPSS Core Server.

Use the Elastic installation instructions found on the download page with the following modifications:

- There are no HPSS-specific Elastic settings.

To start Elastic automatically, see the instructions at https://www.elastic.co/guide/en/elasticsearch/reference/current/starting-elasticsearch.html

By default, security is disabled. To enable security, read: https://www.elastic.co/guide/en/elasticsearch/reference/current/configuring-stack-security.html

6.1.4. Install Kibana

Download Kibana from https://www.elastic.co/downloads/kibana

On the right side, select "View past releases" and then version 7.15.0.

It is recommended that you do **NOT** run Kibana on the HPSS Core Server.

Use the Kibana installation instructions found on the download page, with the following modifications:

- Load the HPSS Dashboards using Section 6.3.1, “Load the HPSS templates into Kibana”.

To start Kibana automatically, see the instructions at https://www.elastic.co/guide/en/kibana/current/start-stop.html

6.1.5. Scaling Elastic

The configuration provided in the above links are for a very basic installation of Elastic. As your system grows, you will need to scale it to meet the increased demand. The Elastic website has a wealth of information on how to scale your system. Here are two web sites to get you started on scaling Elasticsearch:

- https://www.elastic.co/guide/en/elasticsearch/reference/current/scalability.html
- https://www.elastic.co/blog/small-medium-or-large-scaling-elasticsearch-and-evolving-the-elastic-stack-to-fit

6.2. Installing the HPSS data capture components

6.2.1. HPSS data capture scripts

The HPSS data capture components are a series of scripts that call HPSS tools and translate the output into a format usable by ELK. They are designed to be run as cron jobs.

- hpssadm_hs_info
- hpssadm_devices_and_drives_list
- hpssadm_sc_list
- hpssadm_servers_list
- hpss_dump_acct
- hpss_dump_server_metrics

The HPSS data capture scripts are included with the HPSS Core Server RPM.

6.2.2. How to configure the data capture scripts

The data capture scripts should be configured as cron jobs. As the HPSS user, run **crontab -e** and add the following entries.

```
*/5 * * * * /opt/hpss/bin/hpss_dump_server_metrics -r
*/5 * * * * /opt/hpss/bin/hpss_dump_acct -r
*/5 * * * * /opt/hpss/bin/hpssadm_hs_info -r
*/5 * * * * /opt/hpss/bin/hpssadm_sc_list -r
*/5 * * * * /opt/hpss/bin/hpssadm_servers_list -r
*/5 * * * * /opt/hpss/bin/hpssadm_devices_and_drives_list -r
```

This configuration will capture all the statistics every five minutes. It is recommended that sites decide how frequently to capture data and modify the crontab to run the capture scripts based on that need. For example, a site might need to capture the server status more frequently and the account summary data less frequently.

If you change the frequency of data capture, you will need to change some of the provided Kibana templates. These lenses can be identified by the "Last 5 minutes" tag in the upper right hand corner when looking at the dashboard. For those lenses, the "Customized Time Range" for the lens needs to match the frequency of data collection. Otherwise, the lens will summarize too much information.

6.3. Setup the HPSS dashboards

6.3.1. Load the HPSS templates into Kibana

To load the HPSS templates into Kibana:

1. Create a space named HPSS in Kibana
2. Import the hpss/tools/capture_scripts/templates/hpss_kibana_templates.ndjson file

The instructions for creating a space and importing objects can be found at the following:

- https://www.elastic.co/guide/en/kibana/master/xpack-spaces.html
- https://www.elastic.co/guide/en/kibana/current/managing-saved-objects.html

The HPSS dashboard templates are included with the HPSS Core Server RPM.

6.3.2. The HPSS dashboards

The following dashboards will be imported:

Dashboard Name	Description
HPSS SSM Dashboard	Displays information from the HPSS GUI window.
Heatmaps: Transfers by COS	Heatmaps that show the activity by HPSS COS.
Heatmaps: Transfers by SC	Heatmaps that show the activity by HPSS SC.
Heatmaps: Transfers by users	Heatmaps that show the activity by HPSS user.
Status: Core Server specific information	Detailed statistics about the core server.

Status: Devices and drives	Current and status history for devices and drives information from the HPSS SSM.
Status: HPSS Servers	Current and status history for the HPSS servers from the HPSS SSM.
Status: HPSS Servers - Op State	Current and history for the HPSS server's Op State from the HPSS SSM.
Status: Storage Classes	Current and status history for the active Storage Classes from the HPSS SSM.
Status: Tape Devices	Shows information about the PVL jobs and tape mounts.
Storage: Broken down by COS	Displays the latest values for file counts and bytes stored broken down by COS.
Storage: Broken down by SC	Displays the latest values for file counts and bytes stored broken down by SC.
Storage: Summary	See summaries of disk and tape storage spaces. Also provides tape mounts per hour.
Transfers: Broken down by COS	Transfer information on reads and writes broken down by COS. This includes both the bytes transferred and the number of transfers.
Transfers: Broken down by device	Bytes written, read, and errors by device.
Transfers: Broken down by mover	Transfer information on reads and writes broken down by HPSS mover. This includes both the bytes transferred and the number of transfers.
Transfers: Operational statistics	Shows the rate of creates, deletes, and opens occurring on the system.

6.4. Captured data

The captured data is stored as indexes in ELK. The index names are based on the message id (msgid field), the year and month, and the ELK version. Here are the msgids produced by HPSS:

hpss_hs
: This index contains the output from the **hpssadm "health_status info"** command. Some of this data will reset when the SSM restarts.

hpss_sc_active
: This index contains the output from the **hpssadm "list Active Storage Classes"** command. The space values here can not be added together because Disk Storage Classes (SC) use bytes and Tape SCs use virtual volumes (VV). It is possible to write custom filters to group just disk or tape SC.

hpss_devices_and_drives
: This index contains the output from the **hpssadm "list Devices and Drives"** command. The "Bytes Read" and "Bytes Written" fields are stripped from this output. This data resets when the SSM resets. It is not reset by **hpssadm_dump_devices_and_drives**.

hpss_servers
: This index contains the output from the **hpssadm "list Servers"** command.

hpss_acct_sum
: This index contains the output from the **dump_acct_sum -v** command. The data here increments and does not reset when the servers go down.

hpss_acct_bandwith
: This index contains the output from the **dump_acct_sum -b** command. The data here increments and does not reset when the servers go down.

hpss_cos_file_count
: This index contains the output from the **dump_acct_sum** command. The data here increments and does not reset when the servers go down. This data isn't used on any dashboard. The provided templates calculated the similar values by summing the **hpss_acct_sum -v** command output.

hpss_server_metrics_mover
: This index contains the output from **hpss_server_metrics -t mover** command. The data fields reset after every call to **hpss_dump_server_metrics** are:

  - RequestProcessed
  - DataTransfers
  - RequestErrors
  - BufferSize
  - BytesMoved
  - TotalMoveTime

hpss_server_metrics_device_stats
: This index contains the output from the **hpss_server_metrics -t drives** command. The data fields reset after every call to **hpss_dump_server_metrics** are:

  - NumberOfErrors
  - BytesRead
  - BytesWritten

hpss_server_metrics_storage
: This index contains the output from the **hpss_server_metrics -t storage** command. No fields are reset in this index.

hpss_server_metrics_config
: This index contains the output from the **hpss_server_metrics -t config** command. No fields are reset in this index.

hpss_server_metrics_statfs
: This index contains the output from the **hpss_server_metrics -t statfs** command. No fields are reset in this index.

hpss_server_metrics_core

This index contains the output from the **hpss_server_metrics -t core** command. All the data fields reset after every call to **hpss_dump_server_metrics**.

There are too many fields (approximately 100) to list every one in this document. All of them are reset by the calls to **hpss_dump_server_metrics**.

hpss_server_metrics_core_limits

This index contains the output from the **hpss_server_metrics -t limits** command. No fields are reset in this index.

Chapter 7. HPSS S3 interface

HPSS offers an S3 (Simple Storage Service) interface. The S3 interface is a popular object storage interface and has been popularized by AWS S3. This document will describe the HPSS S3 interface, the goals behind the interface, its limitations, different options, deployment considerations, and troubleshooting and debugging tips.

Note that while the HPSS S3 interface provides functionality similar to the AWS S3 interface, specific aspects of how operations work may differ. This document is meant to describe the setup and usage of the HPSS S3 interface. AWS S3 documentation may serve as a guide for what to expect from the HPSS S3 interface, but anything specific to HPSS will supersede guidance from other sources.

That said, AWS provides a number of resources for understanding S3 at a high level [https://aws.amazon.com/s3/], and provides documentation on programming against an S3 endpoint [https://docs.aws.amazon.com/AmazonS3/latest/API/Welcome.html].

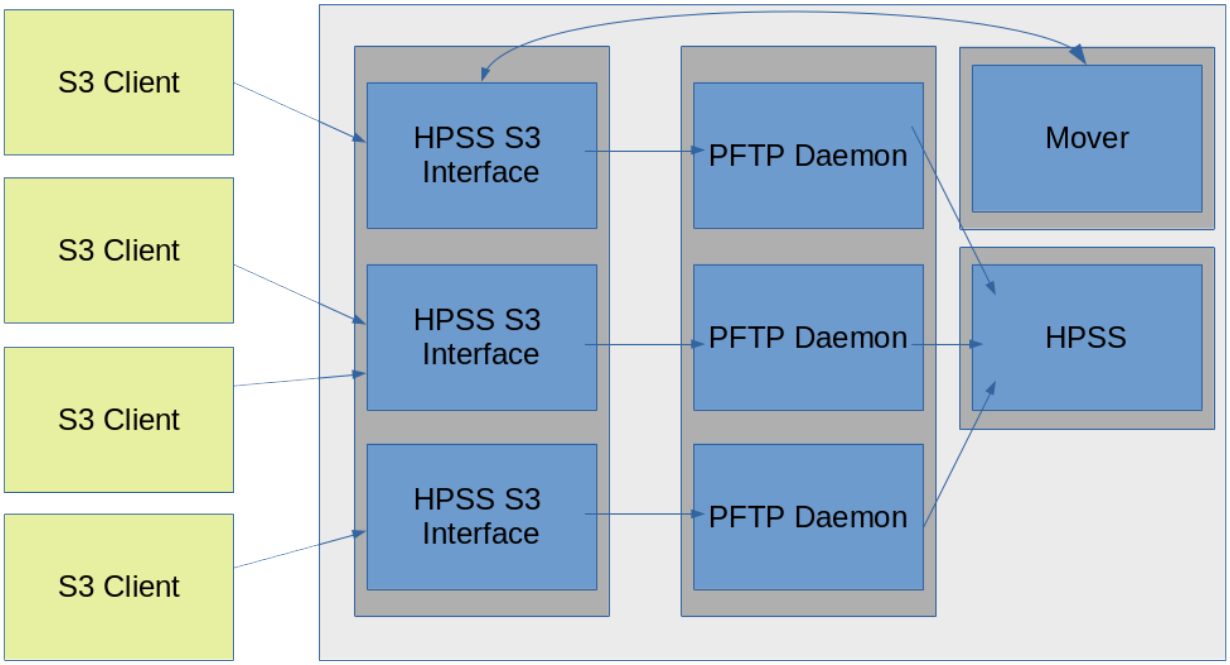
7.1. Overview

The HPSS S3 interface is an HTTP server that processes S3 REST API requests and forwards them to HPSS. The interface uses the HPSS PFTP server as its primary means of communicating with HPSS. The HPSS PFTP server provides mechanisms for using HPSS features through an FTP interface such as COS selection, file family selection, file hashing, user-defined attributes, and high-speed data transfers.

The HPSS S3 interface allows users to list, get, put, and remove HPSS files. The S3 specification has its own limitations, some of which are lower than HPSS limits. When a conflict exists between the two, the S3 limit is enforced.

A key factor of the HPSS S3 interface is that it allows for seamless integration of existing data and user interfaces and new data coming in through S3. Files already in HPSS, or added through other interfaces such as HPSSFS or HSI, are accessible through the HPSS S3 interface if that user would have read access to those files normally. This is explained in detail in the *Section 7.3, "Interoperation"* section below.

The HPSS S3 interface should be deployed on a client system with sufficient CPU to manage validating parallel checksum streams. Multiple HPSS S3 interfaces can be deployed on the same client system. Once deployed, S3 clients can contact the HPSS S3 HTTP servers and issue their S3 commands with minimal setup.



Note that the HPSS S3 interface is **compatible** with S3, rather than **compliant**. That is, there are some aspects of S3 which the HPSS S3 interface does not support and are not implemented. These are described in detail in *Section 7.6, "Unsupported Operations"*

HPSS S3 ultimately logs into HPSS using an HPSS user's username and password. S3 authentication uses ACCESS KEY and SECRET KEY for authentication. The ACCESS KEY will be the user's HPSS accounts to log in. The SECRET KEY will be generated by the administrator and used to map to a user's credentials on the server side.

A java keystore will need to be generated that holds the ACCESS KEY as the alias and the SECRET KEY as a password. The java keystore will be generated as follows:

keytool -importpass -storetype pkcs12 -alias <hpss user> -keystore <hpss keystore path>

Add the keystore path and keystore password to hpss_s3proxy.conf as s3proxy.hpss-creds and s3proxy.hpss-ks-password respectively.

For example: root@tidus [mailto:root@tidus] /opt/hpss > keytool -importpass -storetype pkcs12 -alias hpss -keystore /var/hpss/etc/hpsss3keystore Enter keystore password: hpsshpss Re-enter new password: hpsshpss Enter the password to be stored: hpsspassword Re-enter password: hpsspassword

You will add the following properties to hpss_s3proxy.conf s3proxy.hpss-creds = /var/hpss/etc/hpsss3keystore s3proxy.hpss-ks-password = hpsshpss

7.2. HPSS S3 Interface Setup

To set up the S3 interface, first set up PFTP on the client system. This is covered in the HPSS User's Guide. PFTP is used as the backend application to support S3 operations. The PFTP client should be set up to allow any users that you plan to use the S3 interface to authenticate through the PFTP client.

Next, obtain the HPSS S3 proxy binary. This is a Java binary, so a Java runtime must be installed on the system. See the HPSS Install Guide for Java runtime version requirements. There are no further software requirements to run the HPSS S3 proxy.

Determine what HPSS COS will be used for writing files. A specific HPSS COS can be specified below in the configuration file or it can be left blank or set to 0 for FTP to decide based on file size. On the server identified by jclouds.endpoint, configure the PFTP Daemon in the /var/hpss/etc/HPSS.conf file in the following section:

```
        # Uncomment next line to use the COS from the FileSize Options
        # The default is to ignore the COS specification in the Table.
        ; Set COS Based on Filesize

        # Specify Blocksizes, StripeWidths, and COSs to use based on file size
        # The table must be in strictly ascending order of filesize.
        #
        # COS has no meaning to the Non-HPSS PFTP Daemon
        # COS = 0 means allow the BitFile Server to determine the optimal COS
        #
        # BlockSize has no meaning to the HPSS PFTP Daemon
        #
        # StripeWidth = 0 means use the Bitfile Server Value (HPSS PFTP Daemon)
        #         or use the default (Non-HPSS PFTP Daemon)
        ; FileSize Options = {
                # Files greater than or equal to this value and less than
                # any other value in the table use these Settings
                # e.g., 2MB <= filesize < 10MB
                ; 1MB = {
                        ; BlockSize = 512KB
                        ; StripeWidth = 0
                        ; COS = 2
                ; }
                ; 2MB = {
                        ; BlockSize = 2MB
                        ; StripeWidth = 4
                        ; COS = 0
                ; }
                ; 10MB = {
                        ; BlockSize = 2MB
                        ; StripeWidth= 0
                        ; COS = 0
                ; }
                ; 100MB = {
                        ; BlockSize = 4MB
                        ; StripeWidth= 0
                        ; COS = 0
                ; }
                ; 1GB = {
                        ; BlockSize = 8MB
                        ; StripeWidth= 0
                        ; COS = 0
                ; }
        ; }
```

Determine or create a directory for the S3 buckets to be created in. This can either be an existing directory within HPSS or a new directory. This will become the base directory listed below in the configuration file.

Configure the hpss_s3.conf file.

Configuration Item	Description	Default
s3proxy.authorization	S3 Authorization Type	s3v4
s3proxy.endpoint	Endpoint URL	URL:PORT
s3proxy.hpss-creds	Credential keystore	/var/hpss/etc/hpsss3keystore
s3proxy.hpss-ks-password	Keystore password	hpsshpss
s3proxy.header-cache-size	HTTP Header Cache Size	32768
s3proxy.request-header-size	HTTP Request Header Size	32768
s3proxy.response-header-size	HTTP Response Header Size	32768
s3proxy.output-buffer-size	HTTP Output Buffer Size	524288
s3proxy.transfer-buffer-size	HTTP Transfer Buffer Size	524288
jclouds.pftp.hpss.cos	Default HPSS COS	<Default COS>
jclouds.ftp.basedir	Base Directory	<HPSS Mount>
jclouds.endpoint	Endpoint	PFTP Server:Port
jclouds.pftp.hpss.max.connections	Max PFTP Connections	Must match xinetd limits
hpsspftp.sockbuf.send.size	HPSS PFTP Send Socket Buffer	524288
hpsspftp.sockbuf.recv.size	HPSS PFTP Recv Socket Buffer	524288
hpsspftp.iobuf.recv.size	HPSS PFTP I/O Buffer Size	524288

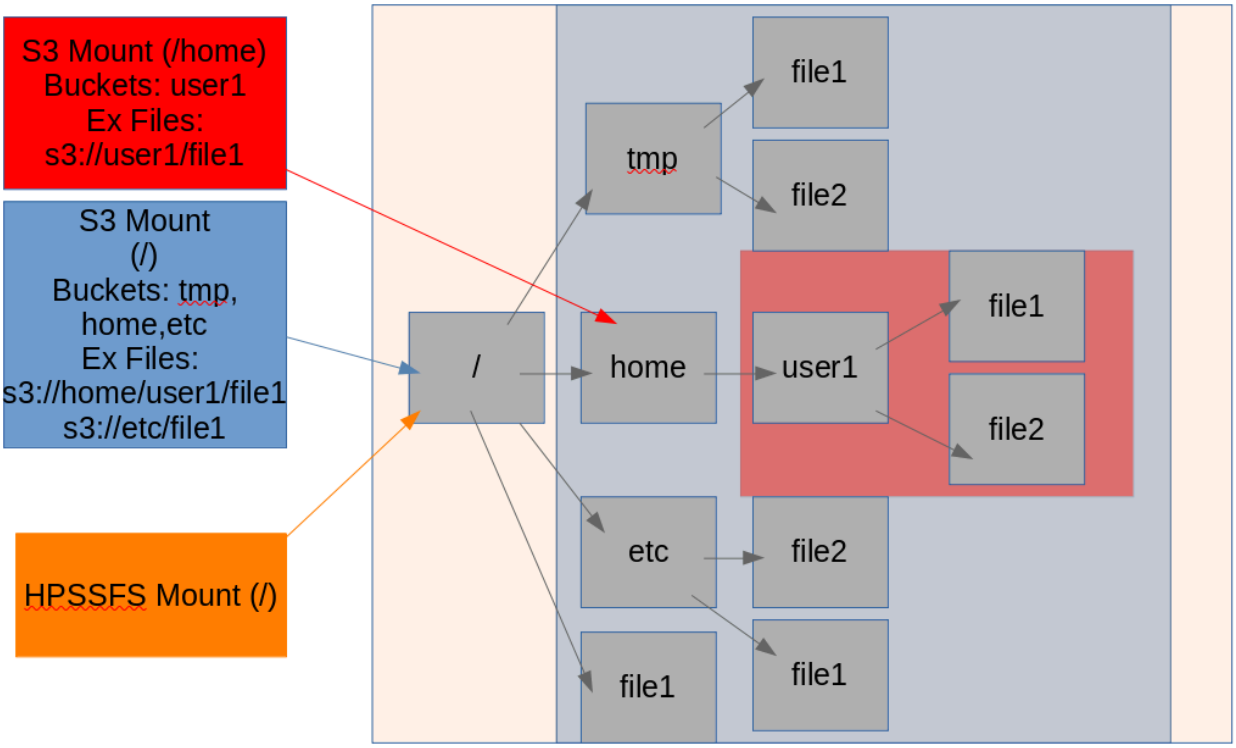
Set up the service file in systemd and start the service. Note that multiple services can be configured for different servers with different configurations, with minor tweaks to the provided service file.

```
# cp hpss_s3proxy.service /etc/systemd/system/
# systemctl daemon-reload
# systemctl enable hpss_s3proxy
# systemctl start hpss_s3proxy
# systemctl status hpss_s3proxy
```

Additionally, users allowed to authenticate to the S3 interface must be defined. Secret keys must be generated by the administrator and provided to users. A mapping file will map HPSS user IDs to secret keys for use in validating incoming requests.

7.3. Interoperation

The HPSS S3 interface is able to interoperate with existing HPSS files and workflows from other HPSS interfaces such as HPSSFS, HTAR, HSI, and PFTP.



A user who interacts with the HPSS S3 interface will see the directories in the basedir as S3 buckets. Everything underneath those buckets are considered objects to the S3 interface. Consider, for example, a path such as /home/user1/file1, where the S3 interface is mounted to /home. user1 is a treated as a bucket in this case, and the object path is "file1" inside the user1 bucket. To reference the same file with the S3 interface mounted on /, "home" is the bucket and the object name is "user1/file1". In the S3 context, "user1/" in this path is referred to as a prefix. When listing buckets which contain multiple subdirectories with files, it is useful to filter based on prefixes or use options to limit recursion.

The S3 interface mounted on /home cannot reference /tmp, /etc/ or /file1. The S3 interface mounted on / also cannot reference /file1 because it is not in a directory (bucket).

The S3 interface is not sequestered from the HPSS global namespace. The basedir configured for the mount point will point at a real HPSS directory, whether it be the root or some other directory such as a home directory. When a user lists the buckets, they will see each directory under the base directory that they have read access to. They will not see directories they do not have read access to. Similarly, when a user lists objects within a bucket, they will see all objects they have read access to, and no objects they do not have read access to.

The HPSS S3 interface does minimal local caching so objects created or deleted by other users in other applications should behave as expected for S3 users. Multi-user access across multiple S3 interfaces and applications has the same potential dangers for overwrites as other interfaces would, so care should be taken by users to avoid overwriting the same file in parallel.

S3 has a concept of "Storage Classes". This has no relation to HPSS Storage Classes. The S3 Storage Classes describe an access pattern or expected latency for retrieval (see the AWS S3 documentation for details). Within the HPSS S3 Interface, S3 objects will be designated with an S3 Storage Class that describes the access pattern or latency of the HPSS backend.

If a HPSS S3 Object is in the "Standard" S3 Storage Class that means on HPSS the top HPSS storage class containing the data is a disk. The access should be very low latency, like a standard S3 storage class.

If a HPSS S3 Object is in the "Standard-IA" S3 Storage Class that means on HPSS the top HPSS storage class containing the data is a tape, but a HPSS stage is not required to get the information from the tape. The access will be minutes or more depending upon the availability of tape drive resources and other requests against the tape.

If a HPSS S3 Object is in the "Deep Archive" S3 Storage Class that means on HPSS the top HPSS storage class containing the data is a tape, with one or more layers of disk above it. Access to this will be minutes or more depending on the availability of tape drive resources and other requests against the tape. Currently, this data can be directly retrieved via a GET operation, but in the future it will mimic the S3 Deep Archive storage class and require a S3 Restore to stage the data up to disk.

S3 Storage Class	Top HPSS Storage Class	Requires Stage?
Standard	Disk	N/A
Standard-IA	Tape	No
Deep Archive	Tape	Yes

7.4. HPSS Specific Options

HPSS supports a set of custom headers used to modify the behavior of HPSS. Most clients allow custom headers to be sent along with S3 requests.

x-hpss-cos
: HPSS Class of Service to use. This only impacts S3 put operations.

x-hpss-family
: HPSS File Family to use. This only impacts S3 put operations.

HPSS supports trashcans. When trashcans are enabled, files deleted within the HPSS S3 interface will be placed in the trashcan. The trashcan may or may not be accessible to end users depending upon the base directory specified.

7.5. S3 Client Setup

Different S3 clients will have different setup requirements. Consult client-specific documentation for general setup instructions and best practices. Below are some general tips and some examples for using the HPSS S3 with clients.

7.5.1. General

The general items that need to be configured for any S3 client will be:

1. **The S3 endpoint.** This is the URL that the client will attempt to contact for S3 services. This is typically a HTTP URL with a port number.

2. **The Access Key.** This is the access key used to authenticate to S3. This will correspond to the HPSS user name.

3. **The Secret Key.** This is the secret key used to sign requests to the S3 requests to the server. The server will verify the request using this secret key, provided to the user.

4. **Use Path Bucket URLs.** This is an older method for specifying bucket URLs, but it is widely supported across clients. Path bucket URLs are URLs where the bucket path is appended to the URL endpoint (e.g. example.com/tmp) and is distinguished from sub-domain URLs (e.g. tmp.example.com). The HPSS S3 interface only supports path bucket URLs.

Some clients may also require specifying the signature version. This is the method used to sign requests to the server with the secret key. AWS V4 (also called S3v4) signatures are the current standard. AWS V2 was the prior standard which has been removed. The HPSS S3 interface only supports the AWS v4 signature.

7.5.2. s3cmd

For s3cmd, in the `s3cmd.cfg` file, set the `access_key`, the `host_base`, the `host_bucket`, and the `secret_key`. The `send_chunk` and `recv_chunk` may be modified to tune performance. Additionally, the `host_bucket` should be set to use a post-fix bucket in the form `<ip>:<port>/(%bucket)`.

For example:

```
host_bucket = 127.0.0.1:8080/(%bucket)
```

s3cmd supports the `--add-header` option, which can be used to specify certain *Section 7.4, “HPSS Specific Options”* .

7.5.3. boto3

For boto3, the endpoint_url, aws_access_key_id, and aws_secret_access_key must be provided for s3 clients and resources.

boto3 does not support custom headers directly, but headers can be injected by using a simple snippet like:

Boto3 client example.

```
def _add_header(request, **kwargs):
    request.headers.add_header('x-hpss-cos', '1')

s3 = boto3.client(...)

event_system = s3.meta.events
event_system.register_first('before-sign.*.*', _add_header)
```

7.6. Unsupported Operations

In general, the HPSS S3 interface supports basic operations like make bucket, list bucket, delete bucket, put object, get object, and delete object. A lot of additional options have been added to S3

over time. Some of these options only make sense in a larger AWS context, some are not currently implemented but may be in the future.

The following S3 features are unsupported at this time:

- CORS
- Object Versioning
- Bucket / Object encryption
- Intelligent Tiering
- Inventory Configuration
- Bucket Lifecycle
- Bucket Metrics
- Bucket Policies
- Bucket Replication
- Bucket Tagging
- Object Tagging
- Bucket Website Configuration
- Bucket Transfer Acceleration
- Bucket Analytics Configuration
- Bucket Logging
- Bucket Notifications
- Bucket Request Payment
- Object Legal Hold
- Object Retention
- Object Torrent
- Select Object content
- Object Lambda access points

These features may have fields in otherwise supported operations which are not implemented or supported. For example **list object parts** allows parameters like RequestPayer (not supported) to specify a payer and SSECustomerKey (not supported) to specify an encryption key. While **list object parts** is supported, these additional parameters related to unsupported features are not.

The following S3 client methods are unsupported at this time:

- copy
- copy_object
- restore_object

7.7. Performance

The first place to start tuning HPSS S3 is with PFTP. Verify that a PFTP client on the system where the HPSS S3 interface is deployed can achieve the expected transfer rates. Tune the PFTP client settings in HPSS.conf to achieve the expected rates based on the network and underlying HPSS hardware. This could turn into a general HPSS tuning exercise - in that case, consult the HPSS installation guide for tips on tuning or contact HPSS support.

Once the PFTP interface is well tuned, it is time to tune the S3 interface. The HPSS S3 interface uses a high-performance jclouds module which interacts with PFTP as a PFTP client. This enables the HPSS S3 interface to achieve high performance throughput without a "store and forward" strategy of some other S3 interfaces. Additionally, the S3 interface makes use of a high performance md5 checksumming module to generate eTags, which allows PUT and GET operations to achieve hundreds of MB/s of performance with a minimal amount of CPU overhead.

Some client applications will generate their own checksum to send to the S3 server to verify. This can be time-intensive and is a data integrity vs performance trade-off if it can be turned off. The S3 interface must calculate and return an ETag which in the case of the HPSS S3 interface is MD5.

Client applications also have their own network parameters that may need to be tuned. The HPSS S3 interface usually benefits from a larger socket and buffer size.

7.8. Scaling and Load Balancing

The HPSS S3 interface can be placed behind a load balancing solution such as haproxy for redundancy, scalability, and performance. Multiple HPSS S3 instances can be placed on the same system, or can be spread across multiple client systems. The PFTP layer can be scaled out to include multiple PFTP gateways as well, independent of the HPSS S3 instances. One might imagine a solution where multiple HPSS S3 instances are configured, each with their own PFTP daemon service configured. It could also be determined that there should be multiple HPSS S3 instances per PFTP daemon service, and they could run on the same system.

One consideration for load balancing is that a client must be directed to the same server for its requests in order to support multipart puts (since the server must know about the part information to process the pieces). This can be accomplished using IP source balancing or cookie-based balancing.

IP source balancing may be a good choice if the end clients are expected to come in from a number of source IPs. Otherwise, cookie-based balancing will be needed.

A simple haproxy backend setup using source balancing across four servers might look like:

```
backend app
```

```
    balance     source
    server  hpss1 192.168.222.223:8081 check
    server  hpss2 192.168.222.224:8081 check
    server  hpss3 192.168.222.225:8081 check
    server  hpss4 192.168.222.226:8081 check
```

The end user will connect to the HA proxy IP, which will balance requests across the backend servers.

Appendix A. Glossary of terms and acronyms

ACL	Access Control List
ACSLS	Automated Cartridge System Library Software (Oracle StorageTek)
ADIC	Advanced Digital Information Corporation
accounting	The process of tracking system usage per user, possibly for the purposes of charging for that usage. Also, a log record type used to log accounting information.
ACI	AML Client Interface
AIX	Advanced Interactive Executive. An operating system provided on many IBM machines.
alarm	A log record type used to report situations that require administrator investigation or intervention.
AML	Automated Media Library. A tape robot.
AMS	Archive Management Unit
ANSI	American National Standards Institute
API	Application Program Interface
archive	One or more interconnected storage systems of the same architecture.
ASLR	Address Space Layout Randomization
attribute	When referring to a managed object, an attribute is one discrete piece of information, or set of related information, within that object.
attribute change	When referring to a managed object, an attribute change is the modification of an object attribute. This event may result in a notification being sent to SSM, if SSM is currently registered for that attribute.
audit (security)	An operation that produces lists of HPSS log messages whose record type is SECURITY. A security audit is used to provide a trail of security-relevant activity in HPSS.
AV	Account Validation
bar code	An array of rectangular bars and spaces in a predetermined pattern which represent alphanumeric information in a machine-readable format (such as a UPC symbol).
BFS	HPSS Bitfile Service
bitfile	A file stored in HPSS, represented as a logical string of bits unrestricted in size or internal structure. HPSS imposes a size limitation in 8-bit bytes based upon the maximum size in bytes that can be represented by a 64-bit unsigned integer.
bitfile segment	An internal metadata structure, not normally visible, used by the Core Server to map contiguous pieces of a bitfile to underlying storage.
Bitfile Service	Portion of the HPSS Core Server that provides a logical abstraction of bitfiles to its clients.

BBTM	Blocks Between Tape Marks. The number of data blocks that are written to a tape virtual volume before a tape mark is required on the physical media.
CAP	Cartridge Access Port
cartridge	A physical media container, such as a tape reel or cassette, capable of being mounted on and dismounted from a drive. A fixed disk is technically considered to be a cartridge because it meets this definition and can be logically mounted and dismounted.
class	A type definition in Java. It defines a template on which objects with similar characteristics can be built, and includes variables and methods specific to the class.
Class of Service	A set of storage system characteristics used to group bitfiles with similar logical characteristics and performance requirements together. A Class of Service is supported by an underlying hierarchy of storage classes.
cluster	The unit of storage space allocation on HPSS disks. The smallest amount of disk space that can be allocated from a virtual volume is a cluster. The size of the cluster on any given disk volume is determined by the size of the smallest storage segment that will be allocated on the volume, and other factors.
configuration	The process of initializing or modifying various parameters affecting the behavior of an HPSS server or infrastructure service.
COS	Class of Service
control path	For the SCSI PVR, this is a connection to the library for sending commands. Control paths can be discovered using device_scan.
Core Server	An HPSS server which manages the name space and storage for an HPSS system. The Core Server manages the name space in which files are defined, the attributes of the files, and the storage media on which the files are stored. The Core Server is the central server of an HPSS system. Each storage subsystem uses exactly one Core Server.
CRC	Cyclic Redundancy Check
CS	Core Server
daemon	A UNIX program that runs continuously in the background.
DAS	Distributed AML Server
DB2	A relational database system, a product of IBM Corporation, used by HPSS to store and manage HPSS system metadata.
DCE	Distributed Computing Environment
debug	A log record type used to report internal events that can be helpful in troubleshooting the system.
delog	The process of extracting, formatting, and outputting HPSS central log records. This process is obsolete in 7.4 and later versions of HPSS. HPSS logs are now recorded as plain text.
deregistration	The process of disabling notification to SSM for a particular attribute change.
descriptive name	A human-readable name for an HPSS server.
device	A physical piece of hardware, usually associated with a drive, that is capable of reading or writing data.

directory	An HPSS object that can contain files, symbolic links, hard links, and other directories.
dismount	An operation in which a cartridge is either physically or logically removed from a device, rendering it unreadable and unwritable. In the case of tape cartridges, a dismount operation is a physical operation. In the case of a fixed disk unit, a dismount is a logical operation.
DNS	Domain Name Service
DOE	Department of Energy
DPF	Database Partitioning Feature
drive	A physical piece of hardware capable of reading or writing mounted cartridges. The terms device and drive are often used interchangeably.
EB	Exabyte (2^{60})
EOF	End of File
EOM	End of Media
ERA	Extended Registry Attribute
ESCON	Enterprise System Connection
event	A log record type used to report informational messages (for example, subsystem starting or subsystem terminating).
export	An operation in which a cartridge and its associated storage space are removed from the HPSS system Physical Volume Library. It may or may not include an eject, which is the removal of the cartridge from its Physical Volume Repository.
FC SAN	Fiber Channel Storage Area Network
FIFO	First in first out
file	An object than can be written to, read from, or both, with attributes including access permissions and type, as defined by POSIX (P1003.1-1990). HPSS supports only regular files.
file family	An attribute of an HPSS file that is used to group a set of files on a common set of tape virtual volumes.
fileset	A collection of related files that are organized into a single easily managed unit. A fileset is a disjoint directory tree that can be mounted in some other directory tree to make it accessible to users.
fileset ID	A 64-bit number that uniquely identifies a fileset.
fileset name	A name that uniquely identifies a fileset.
file system ID	A 32-bit number that uniquely identifies an aggregate.
FTP	File Transfer Protocol
FSF	Forward Space File
FSR	Forward Space Record
Gatekeeper	An HPSS server that provides two main services: the ability to schedule the use of HPSS resources referred to as the Gatekeeping Service, and the ability to validate user accounts referred to as the Account Validation Service.

Gatekeeping Service	A registered interface in the Gatekeeper that provides a site the mechanism to create local policy on how to throttle or deny create, open and stage requests and which of these request types to monitor.
Gatekeeping Site Interface	The APIs of the gatekeeping site policy code.
Gatekeeping Site Policy	The gatekeeping shared library code written by the site to monitor and throttle create, open, and/or stage requests.
GB	Gigabyte (2^{30})
GECOS	The comment field in a UNIX password entry that can contain general information about a user, such as office or phone number.
GID	Group Identifier
GK	Gatekeeper
GSS	Generic Security Service
GUI	Graphical User Interface
HA	High Availability
HACMP	High Availability Clustered Multi-Processing - A software package used to implement high availability systems.
HADR	DB2 High Availability Disaster Recovery
halt	A forced shutdown of an HPSS server.
HBA	Host Bus Adapter
HDM	Shorthand for HPSS/DMAP.
hierarchy	See *storage hierarchy*.
HPSS	High Performance Storage System
HPSS-only fileset	An HPSS fileset that is not linked to an external file system (such as XFS).
HTP	HPSS Test Plan
IBM	International Business Machines Corporation
ID	Identifier
IDE	Integrated Drive Electronics
IEEE	Institute of Electrical and Electronics Engineers
import	An operation in which a cartridge and its associated storage space are made available to the HPSS system. An import requires that the cartridge has been physically introduced into a Physical Volume Repository (injected). Importing the cartridge makes it known to the Physical Volume Library.
I/O	Input/Output
IOD/IOR	I/O Descriptor/I/O Reply. Structures used to send control information about data movement requests in HPSS and about the success or failure of the requests.
IP	Internet Protocol
IRIX	SGI’s implementation of UNIX
JRE	Java Runtime Environment
junction	A mount point for an HPSS fileset.

KB	Kilobyte (2^{10})
KDC	Key Distribution Center
LAN	Local Area Network
LANL	Los Alamos National Laboratory
latency	For tape media, the average time in seconds between the start of a read or write request and the time when the drive actually begins reading or writing the tape.
LBP	Logical Block Protection
LDAP	Lightweight Directory Access Protocol
LFT	Local File Transfer
LLNL	Lawrence Livermore National Laboratory
LMU	Library Management Unit
Location Server	An HPSS server that is used to help clients locate the appropriate Core Server or other HPSS server to use for a particular request.
log record	A message generated by an HPSS application and handled and recorded by the HPSS logging subsystem.
log record type	A log record may be of type alarm, event, info, debug, request, security, trace, or accounting.
logging service	An HPSS infrastructure service consisting of the logging subsystem and one or more logging policies. A default logging policy can be specified, which will apply to all servers, or server-specific logging policies may be defined.
LS	Location Server
LSM	Library Storage Module
LTO	Linear Tape-Open. A half-inch open tape technology developed by IBM, HP, and Seagate.
LUN	Logical Unit Number
LVM	Logical Volume Manager
MAC	Mandatory Access Control
managed object	A programming data structure that represents an HPSS system resource. The resource can be monitored and controlled by operations on the managed object. Managed objects in HPSS are used to represent servers, drives, storage media, jobs, and other resources.
MB	Megabyte (2^{20})
MBS	Media Block Size
metadata	Control information about the data stored under HPSS, such as location, access times, permissions, and storage policies. Most HPSS metadata is stored in a DB2 relational database.
method	A Java function or subroutine.
migrate	To copy file data from a level in the file's hierarchy onto the next lower level in the hierarchy.
Migration/ Purge Server	An HPSS server responsible for supervising the placement of data in the storage hierarchies based upon site-defined migration and purge policies.

MM	Metadata Manager. A software library that provides a programming API to interface HPSS servers with the DB2 programming environment.
mount	An operation in which a cartridge is either physically or logically made readable/writable on a drive. In the case of tape cartridges, a mount operation is a physical operation. In the case of a fixed disk unit, a mount is a logical operation.
mount point	A place where a fileset is mounted in the XFS and HPSS name spaces.
Mover	An HPSS server that provides control of storage devices and data transfers within HPSS.
MPS	Migration/Purge Server
MVR	Mover
NASA	National Aeronautics and Space Administration
Name Service	The portion of the Core Server that provides a mapping between names and machine-oriented identifiers. In addition, the Name Service performs access verification and provides the Portable Operating System Interface (POSIX).
name space	The set of name-object pairs managed by the HPSS Core Server.
NERSC	National Energy Research Supercomputer Center
NIS	Network Information Service
NLS	National Language Support
notification	A notice from one server to another about a noteworthy occurrence. HPSS notifications include notices sent from other servers to SSM of changes in managed object attributes, changes in tape mount information, and log messages of type alarm or event.
NS	HPSS Name Service
NSL	National Storage Laboratory
object	See *managed object*.
ORNL	Oak Ridge National Laboratory
OS	Operating System
OS/2	The operating system (multi-tasking, single user) used on the AMU controller PC.
PB	Petabyte (2^{50})
PFTP	Parallel File Transfer Protocol
PFTPD	PFTP Daemon
physical volume	An HPSS object managed jointly by the Core Server and the Physical Volume Library that represents the portion of a virtual volume. A virtual volume may be composed of one or more physical volumes, but a physical volume may contain data from no more than one virtual volume.
Physical Volume Library	An HPSS server that manages mounts and dismounts of HPSS physical volumes.
Physical Volume Repository	An HPSS server that manages the robotic agent responsible for mounting and dismounting cartridges or interfaces with the human agent responsible for mounting and dismounting cartridges.
PIO	Parallel I/O

PIOFS	Parallel I/O File System
POSIX	Portable Operating System Interface (for computer environments).
purge	Deletion of file data from a level in the file's hierarchy after the data has been duplicated at lower levels in the hierarchy and is no longer needed at the deletion level.
purge lock	A lock applied to a bitfile which prohibits the bitfile from being purged.
PV	Physical Volume
PVL	Physical Volume Library
PVM	Physical Volume Manager
PVR	Physical Volume Repository
RAID	Redundant Array of Independent Disks
RAIT	Redundant Array of Independent Tapes
RAM	Random Access Memory
RAO	Recommended Access Order
reclaim	The act of making previously written but now empty tape virtual volumes available for reuse. Reclaimed tape virtual volumes are assigned a new Virtual Volume ID, but retain the rest of their previous characteristics. Reclaim is also the name of the utility program that performs this task.
registration	The process by which SSM requests notification of changes to specified attributes of a managed object.
reinitialization	An HPSS SSM administrative operation that directs an HPSS server to reread its latest configuration information, and to change its operating parameters to match that configuration, without going through a server shutdown and restart.
repack	The act of moving data from a virtual volume onto another virtual volume with the same characteristics with the intention of removing all data from the source virtual volume. Repack is also the name of the utility program that performs this task.
request	A log record type used to report some action being performed by an HPSS server on behalf of a client.
RISC	Reduced Instruction Set Computer/Cycles
RPC	Remote Procedure Call
RSF	Reverse Space File
RSR	Reverse Space Record
SCSI	Small Computer Systems Interface
security	A log record type used to report security-related events (for example, authorization failures).
SGI	Silicon Graphics
shelf tape	A cartridge which has been physically removed from a tape library but whose file metadata still resides in HPSS.
shutdown	An HPSS SSM administrative operation that causes a server to stop its execution gracefully.
sink	The set of destinations to which data is sent during a data transfer, such as disk devices, memory buffers, or network addresses.

SM	System Manager
SMC	SCSI Medium Changer
SME	Subject Matter Expert
SNL	Sandia National Laboratories
SOID	Storage Object ID. An internal HPSS storage object identifier that uniquely identifies a storage resource. The SOID contains a unique identifier for the object, and a unique identifier for the server that manages the object.
source	The set of origins from which data is received during a data transfer, such as disk devices, memory buffers, or network addresses.
SP	Scalable Processor
SS	HPSS Storage Service
SSD	Solid State Drive
SSH	Secure Shell
SSI	Storage Server Interface
SSM	Storage System Management
SSM session	The environment in which an SSM user interacts with the SSM System Manager to monitor and control HPSS. This environment may be the graphical user interface provided by the **hpssgui** program, or it may be the command-line user interface provided by the **hpssadm** program.
SSMSM	Storage System Management System Manager
stage	To copy file data from a level in the file's hierarchy onto the top level in the hierarchy.
start-up	An HPSS SSM administrative operation that causes a server to begin execution.
info	A log record type used to report file staging and other kinds of information.
STK	Storage Technology Corporation (Oracle StorageTek)
storage class	An HPSS object used to group storage media together to provide storage for HPSS data with specific characteristics. The characteristics are both physical and logical.
storage hierarchy	An ordered collection of storage classes. The hierarchy consists of a fixed number of storage levels numbered from level 1 to the number of levels in the hierarchy, with the maximum level being limited to 5 by HPSS. Each level is associated with a specific storage class. Migration and stage commands result in data being copied between different storage levels in the hierarchy. Each Class of Service has an associated hierarchy.
storage level	The relative position of a single storage class in a storage hierarchy. For example, if a storage class is at the top of a hierarchy, the storage level is 1.
storage map	An HPSS object managed by the Core Server to keep track of allocated storage space.
storage segment	An HPSS object managed by the Core Server to provide abstract storage for a bitfile or parts of a bitfile.
Storage Service	The portion of the Core Server which provides control over a hierarchy of virtual and physical storage resources.
storage subsystem	A portion of the HPSS name space that is managed by an independent Core Server and (optionally) Migration/Purge Server.

Storage System Management	An HPSS component that provides monitoring and control of HPSS via a windowed operator interface or command-line interface.
stripe length	The number of bytes that must be written to span all the physical storage media (physical volumes) that are grouped together to form the logical storage media (virtual volume). The stripe length equals the virtual volume block size multiplied by the number of physical volumes in the stripe group (that is, stripe width).
stripe width	The number of physical volumes grouped together to represent a virtual volume.
System Manager	The Storage System Management (SSM) server. It communicates with all other HPSS components requiring monitoring or control. It also communicates with the SSM graphical user interface (**hpssgui**) and command line interface (**hpssadm**).
TB	Terabyte (2^{40})
TCP/IP	Transmission Control Protocol/Internet Protocol
TDS	Tivoli Directory Server
TI-RPC	Transport-Independent-Remote Procedure Call
trace	A log record type used to record procedure entry/exit events during HPSS server software operation.
transaction	A programming construct that enables multiple data operations to possess the following properties: • All operations commit or abort/roll-back together such that they form a single unit of work. • All data modified as part of the same transaction are guaranteed to maintain a consistent state whether the transaction is aborted or committed. • Data modified from one transaction are isolated from other transactions until the transaction is either committed or aborted. • Once the transaction commits, all changes to data are guaranteed to be permanent.
TSA/MP	Tivoli System Automation for Multiplatforms
TSM	Tivoli Storage Manager
UDA	User-defined Attribute
UDP	User Datagram Protocol
UID	User Identifier
UPC	Universal Product Code
UUID	Universal Unique Identifier
VPN	Virtual Private Network
virtual volume	An HPSS object managed by the Core Server that is used to represent logical media. A virtual volume is made up of a group of physical storage media (a stripe group of physical volumes).
virtual volume block size	The size of the block of data bytes that is written to each physical volume of a striped virtual volume before switching to the next physical volume.

VV	Virtual Volume
XDSM	The Open Group’s Data Storage Management standard. It defines APIs that use events to notify Data Management applications about operations on files.
XFS	A file system created by SGI available as open source for the Linux operating system.
XML	Extensible Markup Language

Appendix B. References

1. **3580 Ultrium Tape Drive Setup, Operator and Service Guide** GA32-0415-00
2. **3584 UltraScalable Tape Library Planning and Operator Guide** GA32-0408-01
3. **3584 UltraScalable Tape Library SCSI Reference** WB1108-00
4. **AIX Performance Tuning Guide**
5. **Data Storage Management (XDSM)** API, ISBN 1-85912-190-X
6. **HACMP for AIX, Version 4.4: Concepts and Facilities**
7. **HACMP for AIX, Version 4.4: Planning Guide**
8. **HACMP for AIX, Version 4.4: Installation Guide**
9. **HACMP for AIX, Version 4.4: Administration Guide**
10. **HACMP for AIX, Version 4.4: Troubleshooting Guide**
11. **HPSS Error Messages Reference Manual**, current release
12. **HPSS Programmer's Reference**, current release
13. **HPSS Programmer's Reference - I/O Supplement**, current release
14. **HPSS User's Guide**, current release
15. **IBM SCSI Device Drivers: Installation and User's Guide**, GC35-0154-01
16. **IBM Ultrium Device Drivers Installation and User's Guide** GA32-0430-00.1
17. **IBM Ultrium Device Drivers Programming Reference** WB1304-01
18. **Interfacing Guide DAS**, Order no. DOC F00 011
19. **Installing, Managing, and Using the IBM AIX Parallel I/O File System**, SH34-6065-02
20. **Platform Notes: The hme FastEthernet Device Driver** 805-4449
21. **POSIX 1003.1-1990 Tar Standard**
22. **Reference Guide AMU**, Order no. DOC E00 005
23. **STK Automated Cartridge System Library Software (ACSLS) System Administrator's Guide**, PN 16716
24. **STK Automated Cartridge System Library Software Programmer's Guide**, PN 16718

25.J. Steiner, C. Neuman, and J. Schiller, "**Kerberos: An Authentication Service for Open Network Systems**," USENIX 1988 Winter Conference Proceedings (1988).

26.R.W. Watson and R.A. Coyne, "**The Parallel I/O Architecture of the High-Performance Storage System (HPSS)**," from the 1995 IEEE MSS Symposium, courtesy of the IEEE Computer Society Press.

27.T.W. Tyler and D.S. Fisher, "**Using Distributed OLTP Technology in a High-Performance Storage System**," from the 1995 IEEE MSS Symposium, courtesy of the IEEE Computer Society Press.

28.J.K. Deutsch and M.R. Gary, "**Physical Volume Library Deadlock Avoidance in a Striped Media Environment**," from the 1995 IEEE MSS Symposium, courtesy of the IEEE Computer Society Press.

29.R. Grossman, X. Qin, W. Xu, H. Hulen, and T. Tyler, "**An Architecture for a Scalable, High-Performance Digital Library**," from the 1995 IEEE MSS Symposium, courtesy of the IEEE Computer Society Press.

30.S. Louis and R.D. Burris, "**Management Issues for High-Performance Storage Systems**," from the 1995 IEEE MSS Symposium, courtesy of the IEEE Computer Society Press.

31.D. Fisher, J. Sobolewski, and T. Tyler, "**Distributed Metadata Management in the High Performance Storage System**," from the 1st IEEE Metadata Conference, April 16-18, 1996.

Appendix C. Developer acknowledgments

HPSS is a product of a government-industry collaboration. The project approach is based on the premise that no single company, government laboratory, or research organization has the ability to confront all of the system-level issues that must be resolved for significant advancement in high-performance storage system technology.

HPSS development was performed jointly by IBM Worldwide Government Industry, Lawrence Berkeley National Laboratory, Lawrence Livermore National Laboratory, Los Alamos National Laboratory, NASA Langley Research Center, Oak Ridge National Laboratory, and Sandia National Laboratories.

We would like to acknowledge Argonne National Laboratory, the National Center for Atmospheric Research, and Pacific Northwest Laboratory for their help with initial requirements reviews.

We also wish to acknowledge Cornell Information Technologies of Cornell University for providing assistance with naming service and transaction management evaluations and for joint developments of the Name Service.

In addition, we wish to acknowledge the many discussions, design ideas, implementation and operation experiences we have shared with colleagues at the National Storage Laboratory, the IEEE Mass Storage Systems and Technology Technical Committee, the IEEE Storage System Standards Working Group, and the storage community at large.

We also wish to acknowledge the Cornell Theory Center and the Maui High Performance Computer Center for providing a test bed for the initial HPSS release.

We also wish to acknowledge Gleicher Enterprises, LLC for the development of the HSI, HTAR, and Transfer Agent client applications.

Finally, we wish to acknowledge CEA-DAM (**Commissariat à l'Énergie Atomique - Centre d'Études de Bruyères-le-Châtel**) for providing assistance with development of NFS V3 protocol support.

Appendix D. `HPSS.conf` configuration file

The `HPSS.conf` configuration file contains tuning options to be used by HPSS clients and servers. For additional information, see the `HPSS.conf` manual page.

General `HPSS.conf` rules and suggestions:

- Keywords *must* be specified precisely as shown (no extra spaces). All items are case-sensitive.
- Lines are comprised of *comments*, *blank lines*, *simple specifiers*, *specifiers* and *values* - "abc = def", or *compound specifiers* and *terminators* - "def = { …}."
- A semicolon (";") is used to comment out (deactivate) an actual configuration option. To activate these options, remove the semicolon. (Suggestion)
- Statements with a pound sign ("#") sign as the first non-white character are explanatory comments. (Suggestion)
- Only ten levels of specification are allowed: Stanzas, SubStanzas, Sections, and SubSections.
- SubStanzas may only exist in Compound Stanzas. Sections may only exist in Compound SubStanzas, and SubSections may only exist in Compound Sections.
- No line may exceed 512 characters, including the "= {".
- Comments may be included by entering either a semicolon (";") or a pound sign ("#") as the first non-white character in a line. Use of either of these characters other than as the first character will not be interpreted as a comment (rather, it will be interpreted as part of the specifier or value). This means you can not put comments at the end of a line.
- Indentation is optional but is strongly encouraged (assists in diagnosis). Use "tabs" for indentation. (Strong suggestion)
- Closing braces ("}") must be used to terminate opening braces (" = {"). This *must* appear on a separate line. Maintaining indentation is recommended.
- Spaces before or after the equal sign ("=") are optional.
- Blank lines are ignored.
- If the default value works, don't fix it. (Strong suggestion)
- The non-HPSS PFTP daemon options should be left alone. (Suggestion)

The uncommented settings in `HPSS.conf.tmpl` should be suitable for a "vanilla" UNIX/UNIX HPSS system. The exceptions are "PFTP Client Interfaces" and "Network Interfaces" Stanzas. HPSS and network tuning are highly dependent on the application environment. The values specified herein are not *expected to be applicable to any installation. These two Stanzas should be tailored to suit your site configuration.*

D.1. PFTP Client Stanza

The PFTP client configuration options are in two distinct Stanzas of the HPSS.conf file (PFTP Client Stanza and PFTP Client Interfaces Stanza).

Table D.1. PFTP Client Stanza fields

Configuration type	Abbreviated description
Stanza (CMPD)	PFTP Client = { Example: `PFTP Client = {` *Optional Reserved* Stanza specifier Must be terminated with a matching "}"
SubStanza	SYSLOG Facility = <value> Values: **DAEMON**, **LOCAL0** … **LOCAL7** Example: `SYSLOG Facility = LOCAL2` *Optional* SubStanza specifying the Syslog Facility for the Multinode Daemon
SubStanza	Debug Value = <value> Values: 0 - 3 Example: `Debug Value = 1` *Optional* SubStanza specifying the Level of Debugging for the PFTP client. A larger number provides more information.
SubStanza	Protocol = <value> Values: PDATA_AND_MOVER, PDATA_ONLY (default), PDATA_PUSH Example: `Protocol = PDATA_AND_MOVER` *Optional* SubStanza specifying the default protocol. May contain any of the three protocols supported.
SubStanza	Auto Parallel Size = <value> Value: Size specified as a decimal number or "*x*MB" style notation Example: `Auto Parallel Size = 4MB` *Optional* SubStanza specifying the minimum file size to start using the "auto-parallel" features of the PFTP client.
SubStanza	PortRange = <value>

Configuration type	Abbreviated description
	Value: **StartPort-EndPort** Example: `PortRange = 10100-12100` *Optional* SubStanza specifying the TCP port range to use between the PFTP client and the Mover(s). This may be necessary when port range filters are used for security. Note: The old format (ncacn_ip_tcp[10100-12100]) is still supported for now, but will be phased out soon.
SubStanza	Parallel Block Size = <value> Value: Size specified as a decimal number or "*x*MB" style notation Example: `Parallel Block Size = 512KB` *Optional* SubStanza specifying the size of the data blocks to be used for parallel data transfers
SubStanza	Transfer Buffer Size = <value> Value: Size specified as a decimal number or "*x*MB" style notation Example: `Transfer Buffer Size = 16MB` *Optional* SubStanza specifying the PFTP client I/O buffer sizes
SubStanza	Socket Buffer Size = <value> Value: Viable socket size specified as a decimal number or "*x*MB" style notation Example: `Socket Buffer Size = 1MB` *Optional* SubStanza specifying the Pdata socket buffer sizes
SubStanza	MAX Ptran Size = <value> Value: Size specified as a decimal number or "*x*MB" style notation. NOLIMIT - Set to 250GB, the absolute max. Example: `MAX Ptran Size = 10GB` *Optional* SubStanza specifying a larger transfer size between socket open and closure. For disk COSs, the segment sizes may potentially override this specification.
SubStanza	Enable SAN3P Example: `Enable SAN3P`

Configuration type	Abbreviated description
	Optional Enable SAN3P support. Default is SAN3P disabled. Note: this will also set the protocol to PDATA_AND_MOVER.
SubStanza	TA API Debug Level Values: 0 - 4 Example: `TA API Debug Level = 0` *Optional* SubStanza specifying the level of debugging for the transfer agent. A larger number provides more information.
SubStanza	TA API Debug Log Value: Debug log file path Example: `TA API Debug Log = /var/TA/logs/PFTPC_TA_API_debug_%N_%P.log` *Optional* SubStanza specifying the path to the transfer agent log file. A **%N** will be replaced by the number of agents, and a **%P** will be replaced by the PID of the particular agent.
SubStanza	Special Features Enabled Example: `Special Features Enabled` *Optional* SubStanza for performance testing *only*. Should *not* be activated except by the appropriate personnel. Default is Off.
SubStanza	Disable stagebatch on mget Example: `Disable stagebatch on mget` *Optional* SubStanza specifying disabling using the batch stage site command to stage files prior to calling mget. Enabled unless this is active.
SubStanza	Use DirectIO Example: `Use DirectIO` *Optional* Enable Direct I/O for reads from the local file system. This is beneficial in cases where kernel caching may hinder performance.
SubStanza	DirectIO Block Size Example: `DirectIO Block Size = 512` *Optional* The block size of the local disk. Direct I/O requires that the transfer be block-aligned with the local device.

Note: All PFTP Client SubStanzas are optional.

The **PFTP Client = { … }** Stanza contains several optional specifications for the **pftp_client** executables.

The **SYSLOG Facility = value** is used to establish the syslog facility for the multinode daemon. It has no relevance to the PFTP client running without the multinode daemon.

The **Debug Value = value** is used to provide additional diagnostic information for the PFTP client. Under normal circumstances, this should be set to "0" (default) or "1". Larger values will provide additional information, but will cause users to complain.

The **Authentication Mechanism = value** is used to determine the preferred authentication mechanism for the PFTP client. If the desired mechanism is to use Kerberos credentials, this should be activated and set to **GSS**. Unless this is appropriately set, the PFTP client will *not* use Kerberos credentials even if they exist. The final determination of permitted mechanisms is performed by the PFTP server.

The **Protocol = value** SubStanza is used to specify the desired PFTP protocol. Currently, any of three values may be specified: **PDATA_AND_MOVER**, **PDATA_ONLY,** or **PDATA_PUSH**. The default specification is **PDATA_AND_MOVER**. The **PDATA_ONLY** specification provides improved performance in high latency WAN environments.

The pftp_client automatically performs conversion of **get** and **put** commands to their parallel equivalents, **pget** and **pput**. Some sites have reported degraded performance as a result of this substitution occurring with small file transfers. To accommodate this problem, the **Auto Parallel Size = value** SubStanza may be specified in the `HPSS.conf` file where the "automatic" parallel features will *not* be invoked if the size of the file is less than the value provided. The value may be specified as a decimal number (1048576) or in the format: *x*MB.

For sites with "Firewalls/ Diodes" installed, it may be necessary to provide specific ports for the data to move between the PFTP client and the Mover(s). This can be accomplished by specifying the **PortRange = value** SubStanza. The syntax of this statement is:

```
PortRange = 10100-12100
```

The old syntax of the value (ncacn_ip_tcp[10100-12100]) identical to DCE's **RPC_RESTRICTED_PORTS** environment variable is still supported for now, but will be phased out soon. At this time, the restricted ports are ignored for passive transfers. Arbitrary ports will be assigned.

Additional options are available for controlling the size of the PFTP transfer buffers, **Transfer Buffer Size**, and the buffer size for the sockets in the **PDATA_ONLY** protocol, **Socket Buffer Size**. The value may be specified as a decimal number (such as "1048576") or in the format: *x*MB.

The PFTP parallel protocol opens and closes the sockets between the PFTP client child processes and any Movers. The default value for tape was every 512 MB and for disk was the smaller of the size of 64 storage segments or 512 MB. With transfers increasing in performance into the 100 MB/second and greater range, the opening and closing of sockets is another performance problem. The **MAX Ptran Size = value** SubStanza has been provided to allow for larger transfers between socket open and closing.

in the case of disks, the 64 storage segments is still the overriding specification, so storage classes need careful specification to provide very large segments if the value associated

with **MAX Ptran Size** is large. An artificial limit of 250 GB is compiled into the PFTP client, which should not cause a great concern any time in the near future. Even at 1 GB/second, this is several minutes. The value may be specified as a decimal number (such as "4294967296") or in the format: *x*GB.

Under normal operating conditions, the **Special Features Enabled** component should remain commented out.

PFTP Client Stanza example (with suggested contents):

```
PFTP Client = {
    # SYSLOG facility to use for all syslog messages by the Multinode Daemon.
    SYSLOG Facility = LOCAL0
    # Debugging Levels
    ; Debug Value = 1
    # Set Default Protocol
    # Values are PDATA_ONLY, PDATA_AND_MOVER, or PDATA_PUSH
    # Default is PDATA_ONLY
    ; Protocol = PDATA_AND_MOVER
    # Set Minimum File Size for auto mapping Non-parallel
    # commands to Parallel commands
    Auto Parallel Size = 4MB
    # The Port Range to be used for connections to other machines
    # Useful if Client is used to cross Network Filters
    # (The older ncacn_ip_tcp[10100-12100] format is supported for now)
      # Default = 10100-65535
    ; PortRange = 10100-12100
    # PDATA Options
    ; Parallel Block Size = 512KB
    Transfer Buffer Size = 16MB
    ; Socket Buffer Size = 1MB
    # PFTP sets an Artificial (Compiled in) Maximum of 250GB
    MAX Ptran Size = 10GB
    # Enable SAN3P
    # Note: this will also set protocol to PDATA_AND_MOVER
    ; Enable SAN3P
    # Location of temp file for parallel pipes
    # Default = /tmp
    ; Temp Pipe File Path = /tmp
    # The (optional) Agent Debug Level Stanza sets the
    # debug level (0-4, 0=no debug, 4=max debug) for the
    # Transfer Agent processes.  This overrides the command
    # line "-d" option for the agent, if specified in this file.
    TA API Debug Level = 0
    # The Debug Logfile contains debugging output. It can
    # be overridden by environment variable settings.
    ; TA API Debug Log = /var/TA/logs/PFTPC_TA_API_debug_%N_%P.log
    # Special Features
    ; Special Features Enabled
}
```

D.2. PFTP Client Interfaces Stanza

Many systems have multiple interfaces, some of which may not have access to all destination hosts. The PFTP Client Interfaces Stanza is used to specify which interfaces on the source host should be used to communicate to destination PFTP daemons, HPSS Movers, or both. This is particularly useful

if both low-speed and high-speed interfaces are available to the client host and the PFTP data transfers should use the high-speed interfaces.

Table D.2. PFTP Client Interfaces Stanza fields

Configuration type	Abbreviated description
Stanza (CMPD)	PFTP Client Interfaces = { Example: `PFTP Client Interfaces = {` *Optional Reserved* Stanza specifier Must be terminated with a matching "}"
SubStanza (CMPD)	<Hostname> <hostname.domain> = { Example: `my_host my_host.my.domain = {` Contains the hostname(s) executing the PFTP client. Must be terminated with a matching "}" Multiple Hostname SubStanzas may be in a single `HPSS.conf` file representing multiple PFTP client hosts sharing the `HPSS.conf` file
Section (CMPD)	<Name> <Name> = { Name: One or more hostnames Example: `storage storage.my.domain = {` Contains the hostname(s) executing the PFTP daemon. Must be terminated with a matching "}" Multiple Daemon Sections may be in a single Hostname SubStanza representing multiple PFTP daemon destinations which may use different characteristics
SubSection	<Name> or <Dotted IP address> Name: **Valid Interface Name** Dotted IP address: **132.175.1.1** Example: `eth0` Example: `132.175.1.1` *Optional* parameter containing the name or dot notation IP address specification for the interface on the local host (**PFTP client**) to use to connect to the Mover(s) associated with the specified PFTP daemon

The **PFTP Client Interfaces = { … }** Stanza contains several configuration options for the **pftp_client** executables.

SubStanzas refer to the hostname(s) associated with the local system where the pftp_client is being invoked.

Sections refer to the PFTP daemon hostname(s) where the PFTP daemon is being invoked.

SubSections refer to the network interface to be utilized (by the host where the PFTP client is invoked) to transfer data to the HPSS Mover systems.

*For HPSS, this is **not** necessarily the name of the Mover(s).*

SubSections specify names or dot notation IP addresses of the interfaces on the local host to be used. For HPSS, all of these interfaces must be able to connect to the Mover(s).

If and only if a specific COS is specified, these interfaces need only provide connection to the Mover(s) associated with the specific COS.

PFTP Client Interfaces Stanza rules:

- Source hostnames may contain one or more hostnames separated by white spaces (subject to the `HPSS.conf` line character limit).
- "Default" is a reserved notation to be used if the local host is not in the Stanza.
- Destination Host (FTPD Host) may contain one or more hostnames separated by white spaces (subject to the `HPSS.conf` line character limit).
- Interface Specification must be specified by interface name or IP Address dot notation.
- Interfaces **must** be able to connect to destination (HPSS Mover).

Communication failures that are not easily diagnosed will occur if the interface specification is invalid.

The following example is completely commented out. The default interface(s) will be used. This is probably typical for many sites and does not need to be modified unless multiple interfaces and asymmetric networks are involved.

PFTP Client Interfaces Stanza example:

```
; PFTP Client Interfaces = {
   # PFTP Client Host Name(s)
   ; water.clearlake.ibm.com water = {
      # Next Specification is the PFTP Daemon Host
      # water has 3 specific interfaces that can talk
      # to the HPSS Movers associated with the PFTP
      # Daemon Host "water", as well as various
      # interfaces of the form 192.2.nnn.nnn
      ; water.clearlake.ibm.com water = {
         # Interfaces on the host specified as the Client Machine
         ; 192.94.47.227
```

```
         ; 192.175.14.35
         ; 192.222.197.1
         ; eth*
      ; }
      # water has ONLY 1 interface that can talk to the HPSS
      # Movers associated with the PFTP Daemon Host "sneezy"
      ; sneezy sneezy.clearlake.ibm.com = {
         ; 192.94.47.227
      ; }
      # Use the default water interface, 192.100.101.1, to talk
      # to any other PFTP Daemons.
      ; Default = {
         ; 192.100.101.1
      ; }
   ; }

   ; sneezy sneezy.clearlake.ibm.com = {
      ; larry larry.clearlake.ibm.com = {
         ; 192.94.47.226
      ; }
      ; sneezy sneezy.clearlake.ibm.com = {
         ; 192.94.47.226
      ; }
   ; }

   # For all other Client Hosts - This allows a single HPSS.conf
   # file to be available using a common files system. This is
   # ONLY useful for cluster systems that specify "Common"
   # interfaces for multiple # nodes of the cluster (I/O Nodes)
   ; Default = {
      # Client Host Name
      ; water water.clearlake.ibm.com = {
         ; 134.253.14.227
      ; }
   ; }
; }
```

D.3. Multinode Table Stanza

The HPSS PFTP client normally forks children to provide multiple network paths between the PFTP client and any Movers. In some instances, it may be preferable to have these processes (pseudo children) running on independent nodes. In this case, it is necessary to set up a **multinoded** daemon on the independent node or host and have the PFTP client initiate one or more data transfer processes with these child processes. The **Multinode Table** Stanza is used to specify what remote hosts are to perform the "pseudo" PFTP client child processes functions.

Table D.3. Multinode Table Stanza fields

Configuration type	Description
Stanza (CMPD)	Multinode Table = { Example: `Multinode Table = {` *Optional Reserved* Stanza specifier

Configuration type	Description
	Must be terminated with a matching "}"
SubStanza	Sleep for Debugger = values Value: Time in seconds Example: `Sleep for Debugger = 15` *Optional* parameter to specify a delay in the multinode daemons to allow diagnosis. This should *only* be specified for diagnostics and will unnecessarily cause degradation if misused. Leave commented out.
SubStanza (CMPD)	<Local Hostname(s)> Example: `my_host my_host.my.domain = {` Contains the local hostname(s) this SubStanza represents Must be terminated with a matching "}"
Section (CMPD)	<Remote Hostname(s)> Example: `FTP_host PFTP_host.domain = {` Contains the hostname(s) of the system running the PFTP server Must be terminated with a matching "}"
Sub-Section	<remote_host> or <remote_host> = <dot notation interface> Example: `his_name` Example: `his_name = 100.102.103.45` Contains the hostname in either string format or dot notation IP address of the host to act as a "pseudo" PFTP child. If a secondary name is specified after the "=", the first interface is to be used for the "control" connection between the PFTP client and the multinoded hosts and the second specification is the interface to be used for the "data" connection(s) to the Mover(s). If only one value is provided, it represents BOTH the "control" and "data" connections.

The **Multinode Table = { … }** Stanza contains one or more SubStanzas specifying the names of the host initiating the PFTP session.

Each Section contains one or more names or IP addresses of remote hosts executing a multinode daemon (**multinoded**). The remote host must have appropriate entries for the **inetd** or **xinetd** superdaemon (`/etc/inetd.conf` and `/etc/services`) to initiate the multinoded.

The Sections may be either a simple Section or a valued Section. A simple SubStanza is a single name or dot notation IP address to be used for both "control" connection and "data" connection. The valued SubStanza is used to specify the name or dot notation IP address for the "control" connection (specifier) and the name or dot notation IP address for the "data" connection (value).

Multinode Table Stanza rules:

- **SubStanza** hostnames (local hosts) may contain one or more hostnames separated by white spaces (subject to the HPSS.conf line character limit.)
- **Section** hostnames (remote hosts) [and/or values] may be specified as either string-based hostnames or dot notation IP addresses. Only one entry per line.

Multinode Table example:

```
# Options read by the Multinode Daemon
Multinode Table = {
    # Diagnostic Delay
    ;  Sleep for Debugger = 15
    # Hostname of the Client
    water water.clearlake.ibm.com =
       # Name of the system running the PFTP Server
       pftp_server pftp_server.clearlake.ibm.com = {
          # Specification of the Multinode Host
          # If the Data Node is a different interface than the interface
          # specified by the Control Node; then enter the Data Node
          # Interface after the "=" otherwise the Control and Data
          # are the same.
          # Control and/or Data may be dot notation OR string hostnames.
          Water = 134.253.14.227
       }
    }
    Default = {
       Default = {
          # If the Data Node is different than the Control Node
          # Enter the Data Node after the "=" otherwise the
          # Control and Data are the same.
          # Control and/or Data may be dot notation OR string hostnames.
          larry = sneezy
       }
    }
}
```

D.4. Network Options Stanza

The network options are in the **Network Options = { … }** Stanza of the HPSS.conf file.

The **Network Options** Stanza allows different options to be specified based on source IP address [Local Interface Name(s)] and destination IP address(es). When the PFTP client, Client API, or Mover establish connections, they will search the contents of this file for entries matching source and destination IP addresses and use the options specified in the matching entry.

The configuration file entries contain values and flags to be used for applying assorted socket and network options including: whether to enable or disable TCP Delay **(TcpNoDelay)**, the socket send

sizes **(SendSpace)** and/or socket receive sizes **(RecvSpace)**, the desired write buffer size **(WriteSize)**, and **RFC1323** support ("Large" Window).

Configuring the TCP socket buffer sizes may also require changing parameters in the OS to allow for larger buffer sizes. TCP socket buffer sizes should be configured based on the client's **bandwidth delay product**, that is, the link capacity (BW) multiplied by the ping time (RTT) should be smaller than the TCP socket buffer size divided by the ping time.

```
BW * RTT <= TCP / RTT
```

Which configuration entry to use is determined based on the Local Interface Name and the destination IP address "masked" by the **NetMask** value. The calling application (PFTP client, Client API, or Mover) will apply the value of the NetMask specification in the configuration file entry to the specified destination address. A **"Default"** destination may be specified for all sources and destinations not explicitly specified in the `HPSS.conf` file.

In Linux, there are several sysctl parameters that can limit the ability of HPSS to change socket buffer sizes. These include **net.core.rmem_max**, **net.core.wmem_max**, **net.ipv4.tcp_rmem**, and **net.ipv4.tcp_wmem** which can be reviewed and modified using sysctl. The `HPSS.conf` settings control what HPSS will attempt to set socket buffer sizes to based on network address, but will be limited to the range supported by the kernel.

Table D.4. Network Options Stanza fields

Configuration type	Description
Stanza (CMPD)	Network Options = { Example: `Network Options = {` *Optional Reserved* Stanza specifier Must be terminated with a matching "}"
SubStanza	Default Write Size = <value> Example: `Default Receive = 1MB` *Optional* SubStanza specifying the default network Read socket size if not specified explicitly
SubStanza	Default Write Size = <value> Example: `Default Send Size = 1MB` *Optional* SubStanza specifying the default network write socket size if not specified explicitly
SubStanza	Default Write Size = <value> Example: `Default Write Size = 4MB` *Optional* SubStanza specifying the default write size if not specified explicitly

Configuration type	Description
SubStanza (CMPD)	<Source IP Interface Name> = { Example: `my_host my_host.my.domain = {` *Optional* SubStanza specifying an interface name. May contain one or more names separated by white spaces. May contain: "Default = {" (*Reserved* Specification) for inclusion of entries not explicitly specified Must be terminated with a matching "}"
Section (CMPD)	<Destination IP address> = { Example: `100.101.102.0 = {` *Optional* SubStanza specifying a dotted decimal address of the destination interface Only one address is allowed; however, networks and sub-networks may be chosen by the appropriate specification of the NetMask. May contain: "Default = {" (*Reserved* Specification) for inclusion of entries not explicitly specified Must be terminated with a matching "}"
SubSection	NetMask = <value> Value: Viable netmask as IP address Example: `NetMask = 255.255.255.0` *Optional* parameter to specify the dotted decimal net mask to apply to the destination IP address to determine whether the entry applies
SubSection	RFC1323 = <value> Values: 0, 1 Example: `RFC1323 = 1` *Optional* parameter to specify whether the RFC1323 option should be disabled ("0") or enabled (any other value)
SubSection	SendSpace = <value> Values: Size specified as decimal value or "xMB" format Example: `SendSpace = 256KB`

Configuration type	Description
	Optional parameter to specify the value to be used for the socket sending buffer space
SubSection	RecvSpace = <value> Values: Size specified as decimal value or "xMB" format Example: `RecvSpace = 256KB` *Optional* parameter to specify the value to be used for the socket receive buffer space
SubSection	WriteSize = <value> Values: Size specified as decimal value or "xMB" format Example: `WriteSize = 1MB` *Optional* parameter used to set the size to be used for each individual write request to the network
SubSection	TcpNoDelay = <value> Values: 0, 1 Example: `TcpNoDelay = 1` *Optional* parameter Indicates whether the TCP Delay option should be disabled (0) or enabled (any other value)

SendSpace and **RecvSpace**. Controls the size of the receive and send buffers for TCP/IP sockets. Internally, HPSS servers and clients attempt to set these buffers' sizes explicitly, but other utilities may not. Typically, the RecvSpace and the SendSpace are equal; however, this is not mandated. Setting either of these values in excess of the system maximum will result in a value less than or equal to the system maximum. The maximum can be observed or changed on AIX using the **"no"** command and, respectively, observing or setting the **sb_max** parameter. There is no portable mechanism for determining the system maximum. Consequently, the specified values may be reduced until an acceptable value is obtained. This process involves a bit-shift operation (divide by two).

RFC1323. Controls whether large TCP window sizes are used. Usually turned on (1) for higher throughput networks (for example, SP/x switch or Gigabit Ethernet) and turned off (0) for lower throughput networks (such as 10/100 Mb Ethernet or FDDI). Large windows provide for increased performance over some networks, but may have a negative performance impact on others.

currently the ability to enable or disable **RFC 1323** support in this manner is specific to AIX. (An equivalent setting for Solaris is the **tcp_wscale_always** flag, set with the command **"ndd /dev/tcp tcp_wscale_always"**.)

The **WriteSize** allows the size of the individual write requests to the TCP/IP connections to be configured. The default behavior (if no entry in the file matches a connection or if zero is entered for the value of this field) is that the size of the write request is the size of the data buffer. On some

networks (for example, the SP/x switch), improved performance has been measured by using a smaller value (such as 32 KB) for the size of the individual writes to the network. If no entry is found that matches a network connection or the value specified is zero, HPSS will query an environment variable, **HPSS_TCP_WRITESIZE**, and use that value, if set and nonzero, for the write size.

The **TcpNoDelay** option determines whether HPSS will enable or disable the algorithm that tries to improve performance from small network writes. This algorithm attempts to coalesce small writes to a TCP/IP connection so they can be sent in a single packet by delaying physical writes to the network. HPSS typically disables this algorithm so that delays are not experienced while sending Mover Protocol and parallel data transfer headers. However, if this causes a performance degradation on a specific network (for example, causes smaller than optimal packet sizes for large transfers), this can be disabled for data transfer connections.

Network Options Stanza Specific rules:

- The first matching entry found in the file will be used to determine the network options used for that connection.
- Multiple "Source Interface Name" SubStanzas may be included within the "Network Options" Stanza. A "Default" Source Interface Name SubStanza may be specified.
- The Source Interface Name SubStanza may specify one or more names (subject to the `HPSS.conf` line character limit, including the "= {").

Do not include the quotes when specifying Default. Destination IP address must be specified in decimal dot notation. Multiple Sections may be included in any SubStanza. A "Default" Destination Interface Name Section may be specified.

Do not include the quotes when specifying Default. The NetMask must be specified in decimal dot IP address notation. All SubSections must be specified in every Section.

NOTE: Tuning is a "fine art" and may vary dramatically within any network configuration and may change with only very minor network configuration modifications. Values provided below are not necessarily "good" numbers.

Network Options Stanza example:

```
#  HPSS Network Options
Network Options = {
    ; Default Receive Size = 1MB
    ; Default Send Size = 1MB
    ; Default Write Size = 4MB
    # My Interface specification(s) using Interface Name
    # Notation
    my_host my_host.domain = {
        # Destination IP address in dot notation
        100.101.102.103 = {
            # The netmask to be applied to the Dest. IP address
            Netmask = 255.255.255.0
            # Use large IP Windows
            RFC1323 = 1
            # Socket Transmission Size.
            SendSpace = 1048576
            # Socket Receive Size.
```

```
            RecvSpace =  1MB
            # The overall buffer size to use for writing.
            WriteSize = 2MB
            # The TCP No Delay Flag is disabled
            TCPNoDelay = 0
        }
        # Default Destination - options to be used for destinations
        # NOT explicitly specified.
        Default = {
            NetMask = 255.255.255.0
            RFC1323 = 1
            SendSpace = 512KB
            RecvSpace =  512KB
            WriteSize = 256KB
            TCPNoDelay = 1
        }
    }
    # Values to be used for source hosts not explicitly specified
    Default  = {
        #  Destination IP address in dot notation
        200.201.202.203 = {
            NetMask = 255.255.255.0
            RFC1323 = 1
            SendSpace = 1048576
            RecvSpace =  1MB
            WriteSize = 2MB
            TCPNoDelay = 0
        }
        # Default Destination - options to be used for destinations
        # NOT explicitly specified.
        Default = {
            NetMask = 255.255.255.0
            RFC1323 = 1
            SendSpace = 256KB
            RecvSpace =  128KB
            WriteSize = 512KB
            TCPNoDelay = 0
        }
    }
}
```

D.5. PFTP Daemon Stanza

A large number of options are available for configuring the PFTP daemon and tuning its performance. These options were previously specified in the `ftpaccess` file or via command-line switches. These options have now been consolidated into the PFTP Daemon Stanza in the `HPSS.conf` file. The options are described below:

Table D.5. PFTP Daemon Stanza description

Configuration type	Abbreviated description
Stanza (CMPD)	PFTP Daemon = { *Optional* Reserved_Stanza specifier on client machines *only*. Required Stanza on all HPSS PFTP server systems.

Configuration type	Abbreviated description
	Must be terminated with a matching "}"
SubStanza	Allow Core Files *Optional* SubStanza specifying that the system should save core files if the PFTP daemon crashes. By default, **xinetd** disables core files.
SubStanza	Core File Directory = <value> Value: **Pathname** Example: `Core File Directory = /var/hpss/adm/core/PFTP_Daemon` *Optional* SubStanza to specify the directory where the PFTP server should put core files
SubStanza	TA API Debug Level Values: 0 - 4 Example: `TA API Debug Level = 0` *Optional* SubStanza specifying the level of debugging for the transfer agent. A larger number provides more information.
SubStanza	TA API Debug Log Value: Debug log file path Example: `TA API Debug Log = /var/TA/logs/PFTPD_TA_API_debug_%N_%P.log` *Optional* SubStanza specifying the path to the transfer agent log file. A **%N** will be replaced by the number of agents, and a **%P** will be replaced by the PID of the particular agent.
SubStanza	SYSLOG Facility = <value> Value: DAEMON, LOCAL0 … LOCAL7 Example: `SYSLOG Facility = LOCAL0` Replaces **-s<string>** option *Optional* SubStanza specifying the syslog facility for the HPSS PFTPD. The default syslog facility is **DAEMON** (reference: `/usr/include/sys/syslog.h`). Alternatives are **LOCAL0** - **LOCAL7**. Incorrect specification will default back to **DAEMON**. To make use of the alternates, modify `/etc/syslog.conf` to use the alternate facility. Note, the file specified in the `/etc/syslog.conf` must exist prior to initialization/refresh of the **syslogd**.
SubStanza	Use Foreign LDAP for Cross Realm

Configuration type	Abbreviated description
	Example: `Use Foreign LDAP for Cross Realm` *Optional* SubStanza specifying that LDAP lookups should use the LDAP server in the foreign realm. Few sites want this option.
SubStanza	Realms are Equivalent Value: off (default), on Example: `Realms are Equivalent = on` *Optional* SubStanza specifying that user in realm A is the entity as in realm B
SubStanza	FTP Base Directory = <value> Value: **Pathname** Example: `FTP Base Directory = /var/hpss` Replaces **-D<string>** option *Optional* SubStanza setting the **{FTPBaseDir}** path. Default: `/var/hpss`. This directory must contain several subdirectories including: `adm`, `bin`, `daemon`, and `etc`. Specific files and subdirectories are located in each of these subdirectories - `etc`: `ftpaccess`, [`ftpbanner`], and `ftpusers`. `adm`: [`daemon.syslog`], [`hpss_ftpd.log`], [`xferlog`]. `daemon`: `ftpd/ftp.pids-hpss_class`. [ ] implies optional others are required. `etc/passwd` is optional for FTP if **Use the KDC Registry** or **Use Extended Registry Attributes** is specified.
SubStanza	FTP Access File = <value> Value: **filename** Example: `FTP Access File = myftpaccess` Replaces **-F<string>** option *Optional* SubStanza setting the {**FTP_FtpAccessFile**}. Default: **ftpaccess**. Located in the directory `{FTPBaseDir}/etc`.
SubStanza	Disable Slash Home Directory Example: `Disable Slash Home Directory` Replaces **-Z** option *Optional* SubStanza disabling use of "/" (forward slash) as the user's home directory. Normally, this should be active for security reasons.
SubStanza	Disable Access if no Home Directory Example: `Disable Access if No Home Directory`

Configuration type	Abbreviated description
	Replaces **-H** option *Optional* SubStanza disallowing login for users whose home directory does not exist or is not properly configured. The default behavior is to put the user in the "/" (forward slash) directory. Normally, this should be active for security reasons.
SubStanza	HPSS FTP Principal = <value> Value: Appropriate HPSS Principal Name Example: `HPSS FTP Principal = hpssftp` Replaces **-P** option and **hpss_option PRINC name** in `ftpaccess` *Optional* SubStanza specifying the HPSS principal representing **hpssftp** (HPSS only)
SubStanza	Disallow Passive Connections Example: `Disallow Passive Connections` *Optional* SubStanza disabling passive connections
SubStanza	Sleep for Debugger = <value> Example: `Sleep for Debugger = 5` Replaces **-z** option *Optional* SubStanza specifying the number of seconds for the HPSS PFTP daemon to sleep at initialization. Useful when attempting to attach to the daemon with the debugger. **NOTE**: leaving this active will cause significant degradation to the PFTP service.
SubStanza	Must Have Credentials Example: `Must Have Credentials` Replaces **-a** option *Optional* SubStanza mandating authentication with Kerberos credentials, disabling{Username}/{Password} authentication. This also disables the **user** command.
SubStanza	Allow CCC Command Example: `Allow CCC Command` *Optional* Kerberos option not relevant to the HPSS PFTP daemon

Configuration type	Abbreviated description
SubStanza	PFTP IO Buffer Size = <value> Example: `PFTP IO Buffer Size = 4MB` Replaces **-b<string>** option *Optional* SubStanza setting the preferred IO Buffer Size for the PFTP server
SubStanza	Debug Value = <value> Example: `Debug Value = 3` Replaces **-d** option(s) *Optional* SubStanza specifying the level of debugging desired (1 - 4). Used internally to determine the quantity and detail of syslog messages from the PFTP daemon.
SubStanza	Non-Parallel HostName = <value> Example: `Non-Parallel HostName = aixrahe.sandia.gov` Replaces **-h** option and **hpss_option HOSTname** in `ftpaccess` *Optional* SubStanza specifying the network interface to be used for data transferred between the PFTPD and the Movers when performing non-parallel transfers. Sets the **HPSS_API_HOSTNAME** environment variable for the Client API (HPSS only).
SubStanza	PFTP Debug Port = <value> Example: `PFTP Debug Port = 6666` Replaces **-p<port>** option *Optional* SubStanza specifying a port to be used by the HPSS PFTP daemon. Used only when initiating the daemon manually (rather than using **inetd/ xinetd**). May be left on; it will not interfere with normal operations.
SubStanza	Default Time Out = <value> Example: `Default Time Out = 1500` Replaces **-t** option and **hpss_option DTO time** in `ftpaccess` *Optional* SubStanza specifying the default timeout in seconds
SubStanza	Default Umask = <value> Example: `Default Umask = 077` Replaces **-u** option and **hpss_option UMASK octal** in `ftpaccess`

Configuration type	Abbreviated description
	Optional SubStanza specifying the default umask in octal
SubStanza	Client API Verbose Value = <value> Example: `Client API Verbose Value = 1` Replaces **-v** option(s) *Optional* SubStanza specifying the level of HPSS Client API Logging to use (1 - 7). The Client API will perform logging specified by the **HPSS_DEBUG** environment variable in a file specified by the **HPSS_DEBUGPATH** environment variable. (Default name is `/var/hpss/ftp/adm/hpss_ftpd.log`.) The default value is **1** (HPSS only).
SubStanza	Disallow User Setting of COS Example: `Disallow User Setting of COS` Replaces **-C** option *Optional* SubStanza to disable the ability of clients to explicitly set the Class of Service for new files (via the **"site setcos"** command). Not recommended.
SubStanza	Maximum Time Out = <value> Value: Time in seconds Example: `Maximum Time Out = 86400` Replaces **-T** option and **hpss_option MTO time** in `ftpaccess` *Optional* SubStanza specifying the maximum timeout in seconds
SubStanza	Use Extended Registry Attributes Example: `Use Extended Registry Attributes` Replaces **-X** option *Optional* SubStanza specifying use of the LDAP registry for authentication (bypassing the **passwd** file) and use of the **HPSS.homedir** and **HPSS.gecos** Extended Registry Attributes (ERAs) for the user's home directory and accounting fields (if they are filled)
SubStanza	Print Performance Data Example: `Print Performance Data` *Optional* SubStanza specifying the printing of additional performance numbers (non-HPSS PFTP daemon only)

Configuration type	Abbreviated description
SubStanza	Number of Ports = <value> Values: 1 - 64 Example: `Number of Ports = 16` *Optional* SubStanza specifying the maximum stripe width allowed (non-HPSS PFTP daemon only)
SubStanza	PortRange = <value> Example: `PortRange = 10100-12100` *Optional* SubStanza specifying the port range to be used for the non-HPSS PFTP daemon which is necessary for parallel transfers. This is ignored for passive listings. The old format (ncacn_ip_tcp[10100-12100]) is still supported at present, but might be phased out in a future version of HPSS.
SubStanza	Socket Buffer Size = <value> Values: Viable Socket Sizes Example: `Socket Buffer Size = 1MB` *Optional* SubStanza specifying the socket buffer size (non-HPSS PFTP daemon only)
SubStanza	Set COS Based on Filesize Example: `Set COS Based on Filesize` *Optional* SubStanza specifying to set the COS from the FileSize Options table. Default: Ignore the COS in the table.
SubStanza (Compound)	FileSize Options = { Example: `FileSize Options = {` *Optional* SubStanza specifier Must be terminated with a matching "}" See notes below
Section (Compound)	<value> = { Example: `1MB = {` *Optional* Section specifier. Must be terminated with a matching "}" See notes below

Configuration type	Abbreviated description
SubSection	BlockSize = <value> Example: `BlockSize = 512KB` *Optional* SubSection specifying the size of data blocks to be used based on file size. (Has no meaning for the HPSS PFTP daemon.)
SubSection	StripeWidth = <value> Example: `StripeWidth = 0` *Optional* SubSection specifying the stripe width to be used based on file size. **0** (zero) means to use the Core Server Value (HPSS PFTP daemon) or use the default (non-HPSS PFTP daemon).
SubSection	COS = <value> Example: `COS = 2` *Optional* SubSection specifying the Class of Service to be used based on file size. **0** (zero) means to allow the Core Server to determine the optimal COS. (Has no meaning for the non-HPSS PFTD daemon.)
SubStanza	<nodename> Service Name = <canonicalname> Example: `sunrahe Service Name = sunrahe.sandia.gov` *Optional* SubStanza specifying the service name to be used by the PFTP daemon node when acquiring credentials. Needed when the servername in the keytab is different from that obtained by **gethostname()**. Use multiple entries when this file is common to multiple PFTP daemons. Useful particularly for clusters and systems having multiple names. One or the other of **host/servicename@realm** or **ftp/servicename@realm** must exist in the Kerberos KDC and in the `/etc/v5srvtab` for the PFTP server executing on the machine mymachine.ssm.com. (Non-HPSS PFTP daemon only.)
SubStanza	Use System Password Files = <value> Example: `Use System Password Files = TRUE` SubStanza specifying that the system password files (`/etc/passwd`, `/etc/group`, `/etc/shadow`) should be used. Should be specified explicitly. TRUE and FALSE are case-sensitive.
SubStanza	PFTP Password File = <value> Value: **Pathname/Filename** Example: `PFTP Password File = /var/hpss/etc/passwd` *Optional* SubStanza used to specify the file containing the user's password information. `/var/hpss/etc/passwd` is the default.

Configuration type	Abbreviated description
SubStanza	PFTP Shadow File = <value> Value: **Pathname/Filename** Example: `PFTP Shadow File = /var/hpss/etc/shadow` *Optional* SubStanza used to specify the file containing the user's protected password information. This should be specified if USERNAME/PASSWORD authentication is in effect.
SubStanza	PFTP Group File = <value> Value: **Pathname/Filename** Example: `PFTP Group File = /var/hpss/etc/groups` *Optional* SubStanza used to specify the file containing the group information for PFTP clients. Default is `/var/hpss/etc/group`.
SubStanza	Primary Authentication Type = <value> Values: **krb5** (default)**, spkm** (Not supported?)**, unix** Example: `Primary Authentication Type = krb5` *Optional* SubStanza used to specify the default authentication type
SubStanza	Primary Authenticator = <value> Values: <auth_type>:<auth_file> where <auth_type> = auth_keytab, auth_keyfile, auth_key, auth_password, auth_none Example: `Primary Authenticator = auth_keytab:/var/hpss/etc/hpss.keytab` *Optional* SubStanza used to specify the file containing the information to authenticate/authorize the **hpssftp** principal
SubStanza	Site Configuration File = <value> Values: **Pathname/Filename** Example: `Site Configuration File = /var/hpss/etc/site.conf` *Optional* SubStanza used to specify the site configuration file to use with the Primary Authentication Mechanism
SubStanza (Compound)	Client Authentication Mechanisms = { Must be terminated with a matching "}"

Configuration type	Abbreviated description
Section (Compound)	<Type> = { Types: **GSS, USER_PASS** Must be terminated with a matching "}"
SubSection	Mapfile Specifier = <value> Values: **Pathname/filename** Example: `Mapfile Specifier = /var/hpss/etc/MapfileName` *Optional* SubStanza used to specify a file containing username mappings. A different file can exist for each authentication type. This file provides the ability to authenticate as one user and be authorized as another user (entity account). These files *must* be protected for security reasons. These files should be owned by *root* and readable and writable by *root* only.
SubSection	Default Authorization Mechanism = <value> Values: **LDAP**, **UNIX**, **DBAS** (not implemented) Example: `Default Authorization Mechanism = LDAP` *Optional* SubStanza used to specify the authorization mechanism desired. The PFTP daemon does authorization internally. If the HPSS system is configured to use LDAP and the PFTP server is configured to use UNIX, the end user will have to be in both authorization facilities. If the LDAP bind type is GSSAPI, the Primary Authorization Mechanism must be "krb5". If LDAP is specified for USER_PASS, a site method must also be specified. **DBAS** has *not* been implemented in PFTP or in HPSS.
SubSection	Use Site Auth Method = <value> Values: CryptoCard, KRB5KDC, SecurID, NIST Example: `Use Site Auth Method = CRYPTOCARD` *Optional* SubStanza used to specify a site-specific authentication mechanism to be used instead of the UserName/Password mechanism. This option requires a specific recompile of the hpss_pftpd with site-specific modules linked in. **NOTE:** if this option is specified, the UserName/Password Mechanism will *not* use a standard password.
SubStanza	{Hostname} Service Name = {servicename} Example: `mymachine Service Name = fire.clearlake.ibm.com` *Optional* SubStanza used to specify alternate service names for the Kerberos service principals. The value after the equal sign is appended

Configuration type	Abbreviated description
	to either "host" or "ftp" to form a service by the name like: host/fire.clearlake.ibm.com@realm. This is very useful for computing clusters and multi-homed systems using a Kerberized PFTP server.

All SubStanzas are optional. If the optional SubStanza **FileSize Options = {** is included, one or more **<value> = {** Sections must be included with mandatory SubSections **BlockSize = …**, **StripeWidth = …**, and **COS = …**.

Each **<value> = {** Section defines the beginning of a range of file sizes to which its settings apply. That range begins with **value** and extends to the next larger **value** included in the **FileSize options = {** SubStanza. In the example below, the settings in the **1MB = {** Section below apply to files with sizes in the range [1MB, 2MB).

PFTP Daemon Stanza example:

```
PFTP Daemon = {
    # Allow the Daemon to take Core Dumps
    ; Allow Core Files
    # Directory to put core files in (Default = .)
    ; Core File Directory = /var/hpss/adm/core
    # The (optional) Agent Debug Level Stanza sets the
    # debug level (0-4, 0=no debug, 4=max debug) for the
    # Transfer Agent processes.  This overrides the command
    # line "-d" option for the agent, if specified in this file.
    TA API Debug Level = 0
    # The Debug Logfile contains debugging output. It can
    # be overridden by environment variable settings.
    ; TA API Debug Log = /var/TA/logs/PFTPD_TA_API_debug_%N_%P.log
    #  Specify the SYSLOG facility to use for all syslog messages
    #  except Authentication Messages.
    #  Values: DAEMON, LOCAL<0-7>
    #  Replaces -sstring option.    Default = DAEMON
    SYSLOG Facility = LOCAL0
    #  Use another cell's LDAP for cross realm authorization
    ;  Use Foreign LDAP for Cross Realm
    #  User in realm A is the same entity as user in realm B
    ;  Realms are Equivalent = On
    #  Specify the Base Directory for PFTP Files
    ;  FTP Base Directory = /var/hpss
    #  Specify the name of the ftpaccess file
    #  This becomes {BaseDir}/etc/{ftpaccess} based on the FTP Base Directory
    ;  FTP Access File = ftpaccess
    #  Do NOT allow / to be Home Directory (HPSS Only)
    Disable Slash Home Directory
    #  Terminate session if Home Directory does NOT Exist (HPSS Only)
    Disable Access if no Home Directory
    #  What Principal to use for HPSS FTP (HPSS Only)
    ; HPSS FTP Principal = hpssftp
    #  Disallow Passive Connections (HPSS Only)
    ; Disallow Passive Connections
    #  Delay for xx seconds to allow dbx attach to the Daemon.
    ; Sleep for Debugger = 5
```

```
#  For Credentials-based PFTP, Deny username/password authentication
; Must Have Credentials
```

```
#  Allow the CCC Command to the GSS Daemon (non-HPSS Only)
#  This allow for the Control channel to be in clear text
; Allow CCC Command
#  Set the IO Buffer Size for the HPSS PFTP Daemon
; PFTP IO Buffer Size = 1MB
#  Specify the Level of Debugging Desired.
; Debug Value = 1
#  For non-Parallel Transfers, specify the Interface (by Name)
#  to use between the PFTP Daemon and the Movers (HPSS Only)
#  Replaces -h option and "hpss_option HOST name" in ftpaccess
; Non-Parallel HostName = aixrahe.sandia.gov
#  Specify the Port to be used for Manual PFTP Daemon Startup
; PFTP Debug Port = 6666
#  Specification in seconds for the Default Timeout
; Default Time Out = 1500
#  Specify (in octal) the Default umask
; Default Umask = 077
#  Specification of the Level of HPSS Client API logging to use ( 0 - 7 )
; Client API Verbose Value = 0
#  Do NOT allow the user to specify Classes of Service (HPSS Only)
; Disallow User Setting of COS
#  Specification in seconds for the Maximum Timeout
; Maximum Time Out = 86400
#  Use the Extended Registry Attributes if they Exist (+HPSS.conf+)
; Use Extended Registry Attributes
#  Print additional Performance Numbers (non-HPSS PFTP Daemon Only)
; Print Performance Data
#  Specify the Maximum Stripe Width Allowed (non-HPSS PFTP Daemon Only)
; Number of Ports = 16
#  The Port Range to be used for the non-HPSS Parallel FTP Daemon
#  which is necessary for Parallel Transfers (non-HPSS PFTP Daemon Only)
; PortRange = 10100-12100
#  The Socket Buffer Size (non-HPSS PFTP Daemon Only)
; Socket Buffer Size = 1MB
#  Uncomment next line to use the COS from the FileSize Options
#  The default is to ignore the COS specification in the Table.
; Set COS Based on Filesize
#  Specify Blocksizes, StripeWidths, and COSs to use based on file size
#  COS has no meaning to the Non-HPSS PFTP Daemon
#  COS = 0 means allow the BitFile Server to determine the optimal COS
#  BlockSize has no meaning to the HPSS PFTP Daemon Only
#  StripeWidth = 0 means use the Bitfile Server Value (HPSS PFTP Daemon)
#      or use the default (Non-HPSS PFTP Daemon)
```

```
; FileSize Options = {
#  Files greater than or equal to this value and less than
#  any other value in the table use these Settings
#  e.g., 2MB <= filesize < 10MB
; 1MB = {
    ; BlockSize = 512KB
    ; StripeWidth = 0
    ; COS = 2
; }
; 2MB = {
    ; BlockSize = 2MB
    ; StripeWidth = 4
    ; COS = 0
```

```
    ; }
    ; 10MB = {
        ; BlockSize = 2MB
        ; StripeWidth= 0
        ; COS = 0
    ; }
    ; 100MB = {
        ; BlockSize = 4MB
        ; StripeWidth= 0
        ; COS = 0
    ; }
    ; 1GB = {
        ; BlockSize = 8MB
        ; StripeWidth= 0
        ; COS = 0
    ; }
    ; }

    # Use the System Password file routines (TRUE or FALSE)
    # The Default for PFTP is FALSE (Case Sensitive!)
    Use System Password Files = FALSE

    #  Path and Name for the PFTP Password File
    PFTP Password File = /var/hpss/etc/passwd

    #  Path and Name for the PFTP Shadow Password File
    # NOTE: PFTP does not currently use the value. It is used ONLY to
    # change how the password is looked up! If the site is using
    # /etc/passwd and the system running the PFTP Daemon utilizes
    # some form of "Shadow" password file to authenticate PFTP users,
    # this should be uncommented.
    # Do NOT remove the part after the "=" sign.
    ;  PFTP Shadow File = /etc/security/passwd

    #  Path and Name for the PFTP Group File
    ;  PFTP Group File = /etc/group

    # Primary Authentication Type for the FTP Daemon
    # This mechanism will be used to authenticate "hpssftp" with HPSS
    # using the PFTP Daemon.
    # Default: krb5 - Options: krb5, spkm(Not supported?), unix
    ; Primary Authentication Mechanism = krb5
```

```
    # Primary Authenticator for the FTP Daemon
    # This mechanism will be used to authenticate "hpssftp" with HPSS
    # using the PFTP Daemon.
    # Format: <auth_type>:<auth_file>
    #         where <auth_type> = auth_keytab, auth_keyfile, auth_key,
    #                             auth_passwd, auth_none
    # Default: auth_keytab:/var/hpss/etc/hpss.keytab
    ; Primary Authenticator = auth_keytab:/var/hpss/etc/hpss.keytab

    # Supported Client Authentication Parameters
    # These parameters will be used to authenticate/authorize
    # the FTP client (end-user) with the FTP Daemon
    # Valid Values: GSS, USER_PASS
    #
    # Mapfile Specifier specifies the type and required information
    #       for Mapping one user name to another.  The types include
```

```
    #       "FILE:", "LDAP:", and "DBAS:" The default type is "FILE:"
    #       For "FILE:" specify the path and name of the file after the ":"
    #       "LDAP" and "DBAS" are NOT currently supported.
    #
    # Default Authorization Mechanism specifies the location of the
    #       pw_struct info.  This may be "UNIX", "LDAP", or "DBAS"
    #       "DBAS" is NOT currently supported.
    #       This probably needs to be specified if a name mapping occurs.
    #
    # If the LDAP bind type is GSSAPI, LDAP may be specified only
    # if the Primary Authentication Mechanism for hpssftp is krb5.
    # LDAP may be specified for USER_PASS only if a site auth method
    # is also specified.
    # Use Site Auth Method is used to specify that the Authentication
    # Mode is Site defined and written for USER_PASS only; e.g.,
    # Crypto will actually use a Crypto Card to Authenticate.
    # NOTE: if this is specified, it is impossible
    # to allow both username/password AND username/"site method"
    # simultaneously. Current methods are KRB5KDC, CryptoCard,
    # SecurId, and NIST
    #
    Client Authentication Mechanisms = {
      ; GSS = {
            ; Mapfile Specifier = FILE:/var/hpss/etc/KRB2UnixMapfile
            ; Default Authorization Mechanism = LDAP
      ; }
      USER_PASS = {
            ; Use Site Auth Method = CryptoCard
            ; Mapfile Specifier = FILE:/var/hpss/etc/Unix2UnixMapfile
            Default Authorization Mechanism = UNIX
      }
    }

    #  Keytab Hostname Mapping Section
    #  Syntax:
    #      machinename Service Name = canonicalname
    #  "machinename" is the name returned by a call to gethostname()
    #  in the PFTP Daemon.
    #  "canonicalname" is the machine name associated with the Kerberos
    #  service - usually the fully qualified name as generated by the
    #  PFTP Client
    #  Specify "{machinename} Service Name" to set the service name to be used
    #  for the PFTP Daemon machine when acquiring creds.  This is needed
    #  when the servername; e.g., host/machinename@realm is different
    #  between the keytab file and the host/machinename obtained where
    #  the machinename is obtained by gethostname() (Non-HPSS PFTP Daemon)
    #  Specify nultiple "machinename Service Name" entries if this
    #  file is common to multiple PFTP Daemon servers.
    ; aixrahe.sandia.gov Service Name = aixrahe.sandia.gov
    ; sunrahe Service Name = sunrahe.sandia.gov
}
```

D.6. Transfer Agent Stanza

A large number of options are available for configuring the transfer agent and tuning its performance.

Table D.6. Transfer Agent Stanza description

Configuration type	Abbreviated description
Stanza (CMPD)	Transfer Agent = { *Reserved* Stanza specifier Must be terminated with a matching "}"
SubStanza	Agent Debug Level = <value> Value: 0 - 4 Example: `Agent Debug Level = 1` *Optional* Sets the debug level (where 0=no debug, 4=max debug) for transfer agent processes
SubStanza	Debug Log File = <value> Value: **Pathname/filename** Example: `Debug Logfile = /var/TA/logs/agent_debug_%N_%P.log` *Optional* Specifies the debugging output file. It can be overridden by environment variable settings.
SubStanza	Enable Core Dump = <value> Value: **YES/NO** Example: `Enable Core Dump = YES` *Optional* Determines whether the agent will enable full core dumps
SubStanza	Working Directory = <value> Value: **Pathname** Example: `Working Directory = /var/TA/cores` *Optional* The absolute pathname of the directory to which the agent will "cd" upon startup, so that core and other important files will be in a known location. The default, if not specified, is the $TMPDIR environment variable setting, or "/tmp" as an absolute fallback.
SubStanza	Nodeset File = <value> Value: **Pathname/filename** Example: `Nodeset File = /usr/local/etc/nodeset.conf` *Optional* File containing named sets of nodes that can be referred to via the "SET:setname" notation

Configuration type	Abbreviated description
SubStanza	Node Affinity File = <value> Value: **Pathname/filename** Example: `Node Affinity File = /usr/local/etc/node_affinity.conf` *Optional* File used to specify groups of nodes that are able to communicate in a network whose topology does not support full interconnection
SubStanza	Shared FS File = <value> Value: **Pathname/filename** Example: `Shared FS File = /usr/local/etc/shared_fs.conf` *Optional* File used to specify the list of shared file system mount points and associated nodes or NodeSets
SubStanza	Agent File = <value> Value: **Pathname/filename** Example: `Agent File = /usr/local/etc/agent.conf` *Optional* File containing the list of client hosts and the list of nodes that can be used as agents from each client
SubStanza	Disabled Node File = <value> Value: **Pathname/filename** Example: `Disabled Node File = /usr/local/etc/disabled_node.conf` *Optional* File used to disable selection of nodes that are temporarily unavailable. It overrides entries in the Agent File.
SubStanza	Audit Logfile = <value> Value: **Pathname/filename** Example: `Audit Logfile = /var/hpss/log/PMTA_AuditLog` *Optional* File used to specify the location of the transfer agent audit log
SubStanza	Debug Logfile = <value> Value: **Pathname/filename** Example: `Debug Logfile = /var/TA/log/pmta_Audit_%N_%P.log` *Optional* Contains an audit trail of all PMTA activity
SubStanza	Agent Auth Mechanism = <value>

Configuration type	Abbreviated description
	Value: **none (default), ident, kerberos** Example: `Agent Auth Mechanism = none` *Optional* The authentication mechanism used by the agent to verify its parent's identity
SubStanza	Agent Authenticator Type = <value> Value: **none (ident), token (future), kerberos (future)** Example: `Agent Authenticator Type = none` *Optional* The mechanism-specific type of authenticator used for verifying the parent's identity
SubStanza	Agent Authenticator File = <value> Value: **Pathname/Filename** Example: `Agent Authenticator File = /usr/local/etc/ident_hosts.conf` *Optional* The absolute pathname of the authentication-specific authenticator file. For "ident", this file contains a list of trusted host patterns that are allowed to launch the transfer agents
SubStanza	SYSLOG Facility = <value> Values: **off**, **LOCAL0** … **LOCAL7** Example: `SYSLOG Facility = LOCAL0` *Optional* Controls logging to syslog
SubStanza	Allow Uid Mapping = <value> Values: **YES, NO (default)** Example: `Allow Uid Mapping = YES` *Optional* Specifies whether the same user can have different UIDs on different machines within the site
SubStanza	Uid Mapfile = <value> Value: **Pathname/filename** Example: `Uid Mapfile = /usr/local/etc/uid_mapfile` *Optional* The name of the mapping file when "Allow Uid Mapping" is set to "YES"
SubStanza	Allow Gid Mapping = <value>

Configuration type	Abbreviated description
	Values: **YES, NO (default)** Example: `Allow Gid Mapping = YES` *Optional* Specifies whether the same group can have different GIDs on different machines within the site
SubStanza	Gid Mapfile = <value> Value: **Pathname/filename** Example: `Gid Mapfile = /usr/local/etc/gid_mapfile` *Optional* The name of the mapping file when "Allow Gid Mapping" is set to "YES"
SubStanza	Umask = <value> Value: **Octal 000 - 777** Example: `Umask = 002` *Optional* Umask value that the transfer agent should set when it starts up
SubStanza	Connections per NIC = <value> Value: **1 (default) - 8** Example: `Connections per NIC = 4` *Optional* The number of data connections that should be opened for each network interface. This should be set to the same value on all agent nodes.
SubStanza	Max Local IO Count = <value> Value: **1 - 8 (default = 4)** Example: `Max Local IO Count = 4` *Optional* The maximum number of concurrent local file I/O ops
SubStanza	File Create Mode = <value> Value: **Octal 000 - 777** Example: `File Create Mode = 640` *Optional* The permissions that should be specified when the transfer agent creates files
SubStanza	Stripe Size = <value> Value: **128KB - 256MB**

Configuration type	Abbreviated description
	Example: `Stripe Size = 16MB` *Optional* The default number of bytes that each agent should write within each stripe of data. The total stripe length is (Stripe Size × number of agents). The value may contain the K, KB, M, or MB suffix. This value may be overridden by the application.

All Stanza and Sub Components are optional.

Transfer Agent Stanza example:

```
# Parallel Multinode Transfer Agent (PMTA) Section
Transfer Agent = {
        # The (optional) Agent Debug Level Stanza sets the
        # debug level (0-4, 0=no debug, 4=max debug) for the
        # Transfer Agent processes.  This overrides the command
        # line "-d" option for the agent, if specified in this file.
        Agent Debug Level = 0

        # The Debug Logfile contains debugging output. It can
        # be overridden by environment variable settings.
        ; Debug Logfile = /var/TA/logs/agent_debug_%N_%P.log

        # Enable Core Dump - this option determines whether the agent will
        # enable full core dumps. Allowable values for this option are YES and
        # NO.  The Default is YES
        Enable Core Dump = yes

        # The optional "Working Directory" setting is the absolute
        # pathname of the directory to which the agent will "cd" upon
        # startup, so that core files, etc. will be in a known location.
        # The default, if not specified, is the $TMPDIR environment
        #  variable setting, or "/tmp" as an absolute fallback.
        Working Directory = /var/TA/cores

        # The (optional) NodeSet File contains named sets of
        # Nodes that can be referred to via the "SET:setname"
        # notation.
        Nodeset File = /usr/local/etc/nodeset.conf

        # The (optional) Node Affinity file is used to specify
        # groups of nodes are able to communicate in a
        # network whose topology does not support full interconnection
        Node Affinity File = /usr/local/etc/node_affinity.conf

        # The Shared Filesystem file is used to specify the
        # list of shared filesystem mount points and associated
        # nodes or NodeSets
        Shared FS File = /usr/local/etc/shared_fs.conf

        # The Agent file contains the list of client hosts and
        # the list of nodes that can be used as Agents from each client
        Agent File = /usr/local/etc/agent.conf

        # The Disabled Node file is used to disable selection of
```

```
# nodes that are temporarily unavailable. It overrides
# entries in the Agent file
Disabled Node File = /usr/local/etc/disabled_node.conf

# The Audit Logfile contains an audit trail of all PMTA activity
; Audit Logfile = /var/TA/logs/pmta_Audit_%N_%P.log

# "Agent Auth Mechanism", if specified, is the authentication
# mechanism used by the Agent to verify its parent's identity.
# Valid settings are:
#   none   (default)
#   ident
#   kerberos
; Agent Auth Mechanism = none
```

```
# "Agent Authenticator Type" is the mechanism-specific type of
# authenticator used for verifying the parent's identity
# Legal settings are:
#   none  [for ident authentication]
#   token [reserved for future use]
#   kerberos [kerberos authentication - TBD]
; Agent Authenticator Type = none

# The "Agent Authenticator File", if specified, is the absolute
# pathname of the authentication-specific authenticator file.
# For "ident", this file contains a list of trusted host patterns
# that are allowed to launch the Transfer Agents
; Agent Authenticator File = /usr/local/etc/ident_hosts.conf

# The SYSLOG Facility parameter controls logging to syslog. Legal
# values are "off" or "LOCALn" where n=0-7.
SYSLOG Facility = off

# The optional "Allow Uid Mapping" setting defines whether the same user
# can have different UIDs on different machines within the site.
# The default value is NO
Allow Uid Mapping = NO

# The optional "Uid Mapfile" setting is the name of the
# mapping file when "Allow Uid Mapping" is set to "YES"
; Uid Mapfile = /usr/local/etc/uid_mapfile

# The optional "Allow Gid Mapping" setting defines whether the same group
# can have different GIDs on different machines within the site.
# The default value is NO
Allow Gid Mapping = NO

# The optional "Gid Mapfile" setting is the name of the mapping
# file used when "Allow Gid Mapping" is set to "YES"
; Gid Mapfile = /usr/local/etc/gid_mapfile

# The optional "Umask" setting is the value that the Transfer Agent
# should set when it starts up.  It is specified as a 3-digit
# octal value.
; Umask = 002

# The optional "Connections Per NIC" setting specifies the number of
# data connections that should be opened for each network interface.
# This should be set to the same value on all agent nodes.
```

```
        # The default is one, and the max allowed is 8.
        ; Connections Per NIC = 4

        # The max number of concurrent local file I/O ops
        # 0 < value <= 8, Default = 4
        ; Max Local IO Count = 4

        # The optional "File Create Mode" setting specifies the permissions
        # that should be specified when the Transfer Agent creates files.
        # It is specified as an octal value
        File Create Mode = 640

        # The optional "Stripe Size" setting is used to specify the default
        # number of bytes that each Agent should write within each stripe of
        # data.  The total stripe length is (Stripe Size * number of Agents).
        # The value may contain K, KB, M, or MB suffix, and must be within
        # the range (128K - 256MB)
        # This value may be overridden by the application.
        Stripe Size = 16MB

        # The optional "Size To Agent Count" SubStanza is used to define the
        # number of agents to be used for a transfer, based upon the transfer
        # size
        # Entry format for each interval is:
        #       AgentCount      MinSize MaxSize
        Size To Agent Count = {
                 0                0     700MB
                 1              700MB     2GB
                 2                2GB    40GB
                 4               40GB   200GB
                 8              200GB   500GB
                16              500GB
        # Note: omitted MaxSize value means 2^64-1
        } # end of Size To Agent Count Section
}
#  end of Transfer Agent Section
```

D.7. Stanzas reserved for future use

The following Stanza names (specifiers) are reserved for future implementation in HPSS and should not be used by application developers.

- Local File Path
- PSI

Appendix E. hpss_env_defs.h

The HPSS environment variables are defined in `/opt/hpss/include/hpss_env_defs.h`. These environment variables can be overridden in `/var/hpss/etc/env.conf` or in the local environment.

*Note: The following contents of the `hpss_env_defs.h` may be inconsistent with the latest HPSS release. Refer to the output from running the **hpss_set_env** command to see the default settings and/or what is currently set in your HPSS system.*

```
static env_t   hpss_env_defs[] = {

/*
 ************************************************************************
 *      HPSS_ROOT               - Root pathname for HPSS Unix top level
 *      HPSS_HOST               - Machine host name
 *      HPSS_NODE_TYPE          - Node type of current machine
 *      HPSS_PATH_INSTALL       - Pathname for HPSS installation
 *      HPSS_PATH_BIN           - Pathname for HPSS executables
 *      HPSS_PATH_MSG           - Pathname for HPSS message catalog
 *      HPSS_PATH_SLASH_BIN     - Pathname for /bin
 *      HPSS_PATH_SLASH_ETC     - Pathname for /etc
 *      HPSS_PATH_USR_BIN       - Pathname for /usr/bin
 *      HPSS_PATH_USR_SBIN      - Pathname for /usr/sbin
 *      HPSS_PATH_VAR           - Pathname for HPSS var directory
 *      HPSS_USER               - HPSS user name
 *      HPSS_USERROOT           - Root user id
 *      HPSS_PATH_DB_INSTALL    - Pathname for DB bin
 *      HPSS_NET_FAMILY         - Default network protocol to be used
 *      HPSS_SOCK_KEEPIDLE_SECS - Default tcp_keepidle seconds to be used
 *      HPSS_SOCK_KEEPIDLE_CNT   - Default keepalive count to be used
 *      HPSS_SOCK_KEEPIDLE_INTVL - Default keepalive interval to be used
 *
 ************************************************************************
 */
        { "HPSS_ROOT",                  "/opt/hpss",            NULL},
        { "HPSS_HOST",                  "%H",                   NULL},
        { "HPSS_NODE_TYPE",             NULL,                   NULL},
        { "HPSS_PATH_INSTALL",          "/opt/hpss",            NULL},
        { "HPSS_PATH_BIN",              "${HPSS_PATH_INSTALL}/bin",
          NULL},
        { "HPSS_PATH_MSG",            "${HPSS_PATH_INSTALL}/msg/En_US",
          NULL},
        { "HPSS_PATH_SLASH_BIN",        "/bin",                 NULL},
        { "HPSS_PATH_SLASH_ETC",        "/etc",                 NULL},
        { "HPSS_PATH_USR_BIN",          "/usr/bin",             NULL},
        { "HPSS_PATH_USR_SBIN",         "/usr/sbin",            NULL},
        { "HPSS_PATH_VAR",              "/var/hpss",            NULL},
        { "HPSS_USER",                  "hpss",                 NULL},
        { "HPSS_USERROOT",              "root",                 NULL},
        { "HPSS_PATH_DB_INSTALL",       "${HPSS_PATH_VAR}/hpssdb", NULL},
        { "HPSS_SYSTEM",                "%S",                   NULL},
        { "HPSS_SYSTEM_VERSION",        "%V",                   NULL},
        { "HPSS_HOST_FULL_NAME",        "%L",                   NULL},
        { "HPSS_NET_FAMILY",            "ipv4_only",            NULL},
        { "HPSS_SOCK_KEEPIDLE_SECS",    "7200",                 NULL},
```

```
        { "HPSS_SOCK_KEEPIDLE_CNT",     "9",                    NULL},
        { "HPSS_SOCK_KEEPIDLE_INTVL",   "75",                   NULL},
/*
 ************************************************************************
 * HPSS Group names
 *      HPSS_GRP_NAME             - HPSS group name
 *      HPSS_GRP_NAME_SERVER      - HPSS Server group name
 *      HPSS_GRP_NAME_CLIENT      - HPSS Client group name
 ************************************************************************
 */
        { "HPSS_GRP_NAME",              "hpss",                         NULL},
        { "HPSS_GRP_NAME_SERVER",       "hpsssrvr",                     NULL},
        { "HPSS_GRP_NAME_CLIENT",       "hpss_client",                  NULL},
/*
 ************************************************************************
 * HPSS Server Principal names
 *
 *      HPSS_PRINCIPAL            - Principal name for SEC Server
 *      HPSS_PRINCIPAL_CORE       - Principal name for CORE Server
 *      HPSS_PRINCIPAL_DMG        - Principal name for GHI DMAPI user
 *      HPSS_PRINCIPAL_FTPD       - Principal name for FTP Daemon
 *      HPSS_PRINCIPAL_GK         - Principal name for Gatekeeper Server
 *      HPSS_PRINCIPAL_HPSSD      - Principal name for Startup Daemon
 *      HPSS_PRINCIPAL_LS         - Principal name for Location Server
 *      HPSS_PRINCIPAL_MOUNTD     - Principal name for Mount Daemon
 *      HPSS_PRINCIPAL_MPS        - Principal name for Migration/Purge Server
 *      HPSS_PRINCIPAL_MVR        - Principal name for Mover
 *      HPSS_PRINCIPAL_PVL        - Principal name for PVL
 *      HPSS_PRINCIPAL_PVR        - Principal name for PVR
 *      HPSS_PRINCIPAL_RAIT       - Principal name for RAIT Engine
 *      HPSS_PRINCIPAL_SSM        - Principal name for SSM
 *      HPSS_PRINCIPAL_ADM_USER   - Principal name for primary HPSS
 *                                  administrator principal
 ************************************************************************
 */
        { "HPSS_PRINCIPAL",             NULL,                   NULL},
        { "HPSS_PRINCIPAL_CORE",        "hpsscore",             NULL},
        { "HPSS_PRINCIPAL_DMG",         "hpssdmg",              NULL},
        { "HPSS_PRINCIPAL_FTPD",        "hpssftp",              NULL},
        { "HPSS_PRINCIPAL_GK",          "hpssgk",               NULL},
        { "HPSS_PRINCIPAL_HPSSD",       "hpsssd",               NULL},
        { "HPSS_PRINCIPAL_LS",          "hpssls",               NULL},
        { "HPSS_PRINCIPAL_FS",          "hpssfs",               NULL},
        { "HPSS_PRINCIPAL_MOUNTD",      "hpssmntd",             NULL},
        { "HPSS_PRINCIPAL_MPS",         "hpssmps",              NULL},
        { "HPSS_PRINCIPAL_MVR",         "hpssmvr",              NULL},
        { "HPSS_PRINCIPAL_PVL",         "hpsspvl",              NULL},
        { "HPSS_PRINCIPAL_PVR",         "hpsspvr",              NULL},
        { "HPSS_PRINCIPAL_RAIT",        "hpssrait",             NULL},
        { "HPSS_PRINCIPAL_SSM",         "hpssssm",              NULL},
        { "HPSS_PRINCIPAL_ADM_USER",    "${HPSS_USER}",         NULL},
/*
 ************************************************************************
 * HPSS Server Principal UID's
 *
 *      HPSS_PRINCIPAL_CORE_UID     - Principal UID for CORE Server
 *      HPSS_PRINCIPAL_FTPD_UID     - Principal UID for FTP Daemon
 *      HPSS_PRINCIPAL_GK_UID       - Principal UID for Gatekeeper Server
 *      HPSS_PRINCIPAL_HPSSD_UID    - Principal UID for Startup Daemon
```

```
 *      HPSS_PRINCIPAL_LS_UID       - Principal UID for Location Server
 *      HPSS_PRINCIPAL_MPS_UID      - Principal UID for Migration/Purge Server
 *      HPSS_PRINCIPAL_MVR_UID      - Principal UID for Mover
 *      HPSS_PRINCIPAL_NS_UID       - Principal UID for Name Server
 *      HPSS_PRINCIPAL_PFSD_UID     - Principal UID for PFS Daemon
 *      HPSS_PRINCIPAL_PVL_UID      - Principal UID for PVL
 *      HPSS_PRINCIPAL_PVR_UID      - Principal UID for PVR
 *      HPSS_PRINCIPAL_RAIT_UID     - Principal UID for RAIT Engine
 *      HPSS_PRINCIPAL_SS_UID       - Principal UID for Storage Server
 *      HPSS_PRINCIPAL_SSM_UID      - Principal UID for SSM
 *
 * NOTE: Principal UID must be in the format of "-uid <number of uid>"
 *       For example:
 *      { "HPSS_PRINCIPAL_BFS_UID"          "-uid 1234",        NULL},
 *
 ******************************************************************************
 */
       { "HPSS_PRINCIPAL_CORE_UID",    "301",                  NULL},
       { "HPSS_PRINCIPAL_FTPD_UID",    "302",                  NULL},
       { "HPSS_PRINCIPAL_GK_UID",      "303",                  NULL},
       { "HPSS_PRINCIPAL_HPSSD_UID",   "304",                  NULL},
       { "HPSS_PRINCIPAL_LS_UID",      "305",                  NULL},
       { "HPSS_PRINCIPAL_FS_UID",      "306",                  NULL},
       { "HPSS_PRINCIPAL_MPS_UID",     "307",                  NULL},
       { "HPSS_PRINCIPAL_MVR_UID",     "308",                  NULL},
       { "HPSS_PRINCIPAL_PVL_UID",     "309",                  NULL},
       { "HPSS_PRINCIPAL_PVR_UID",     "310",                  NULL},
       { "HPSS_PRINCIPAL_RAIT_UID",    "311",                  NULL},
       { "HPSS_PRINCIPAL_SSM_UID",     "312",                  NULL},
/*
 ******************************************************************************
 * HPSS Server Executable Names
 *
 *      HPSS_EXEC_ACCT          - executable name for Accounting
 *      HPSS_EXEC_CORE          - executable name for CORE Server
 *      HPSS_EXEC_FTPD          - executable name for FTPD
 *      HPSS_EXEC_GK            - executable name for Gatekeeper Server
 *      HPSS_EXEC_HPSSD         - executable name for Start Daemon
 *      HPSS_EXEC_LS            - executable name for Location Server
 *      HPSS_EXEC_MOUNTD        - executable name for Mount Daemon
 *      HPSS_EXEC_MPS           - executable name for Migration/Purge Server
 *      HPSS_EXEC_MVR           - executable name for Mover
 *      HPSS_EXEC_MVR_TCP       - executable name for Mover TCP
 *      HPSS_EXEC_PFSD          - executable name for PFS Daemon
 *      HPSS_EXEC_PVL           - executable name for PVL
 *      HPSS_EXEC_PVR_OPER      - executable name for PVR Operator
 *      HPSS_EXEC_PVR_STK       - executable name for PVR STK
 *      HPSS_EXEC_PVR_AML       - executable name for PVR AML
 *      HPSS_EXEC_PVR_SCSI      - executable name for PVR SCSI
 *      HPSS_EXEC_RAIT          - executable name for RAIT Engine
 *      HPSS_EXEC_RAIT_TCP      - executable name for RAIT Engine TCP
 *      HPSS_EXEC_SSMSM         - executable name for SSM Storage Manager
 *
 ******************************************************************************
 */
       { "HPSS_EXEC_ACCT",             "${HPSS_PATH_BIN}/hpss_acct",
         NULL},
       { "HPSS_EXEC_CORE",             "${HPSS_PATH_BIN}/hpss_core",
         NULL},
```

```
        { "HPSS_EXEC_FTPD",             "${HPSS_PATH_BIN}/hpss_pftpd",
          NULL},
        { "HPSS_EXEC_GK",               "${HPSS_PATH_BIN}/hpss_gk",
          NULL},
        { "HPSS_EXEC_HPSSD",            "${HPSS_PATH_BIN}/hpss_sd",
          NULL},
        { "HPSS_EXEC_LS",               "${HPSS_PATH_BIN}/hpss_ls",
          NULL},
        { "HPSS_EXEC_MOUNTD",           "${HPSS_PATH_BIN}/hpss_mnt",
          NULL},
        { "HPSS_EXEC_MPS",              "${HPSS_PATH_BIN}/hpss_mps",
          NULL},
        { "HPSS_EXEC_MVR",              "${HPSS_PATH_BIN}/hpss_mvr",
          NULL},
        { "HPSS_EXEC_MVR_TCP",          "${HPSS_PATH_BIN}/hpss_mvr_tcp",
          NULL},
        { "HPSS_EXEC_PVL",              "${HPSS_PATH_BIN}/hpss_pvl",
          NULL},
        { "HPSS_EXEC_PVR_OPER",         "${HPSS_PATH_BIN}/hpss_pvr_operator",
          NULL},
        { "HPSS_EXEC_PVR_STK",          "${HPSS_PATH_BIN}/hpss_pvr_stk",
          NULL},
        { "HPSS_EXEC_PVR_AML",          "${HPSS_PATH_BIN}/hpss_pvr_aml",
          NULL},
        { "HPSS_EXEC_PVR_SCSI",         "${HPSS_PATH_BIN}/hpss_pvr_scsi",
          NULL},
        { "HPSS_EXEC_RAIT",             "${HPSS_PATH_BIN}/hpss_rait_engine",
          NULL},
        { "HPSS_EXEC_RAIT_TCP",         "${HPSS_PATH_BIN}/hpss_rait_engine_tcp",
          NULL},
        { "HPSS_EXEC_SSMSM",            "${HPSS_PATH_BIN}/hpss_ssmsm",
          NULL},
/*
 ******************************************************************************
 * Utilities need by SSM
 *
 *      HPSS_EXEC_ACL_EDIT              - Pathname for the acl_edit utility
 *      HPSS_EXEC_CDSCP                 - Pathname for the cdscp utility
 *      HPSS_EXEC_DELOG                 - Pathname for the delog utility
 *      HPSS_EXEC_RECLAIM               - Pathname for the reclaim utility
 *      HPSS_EXEC_REPACK                - Pathname for the repack utility
 *
 ******************************************************************************
 */
        { "HPSS_EXEC_ACL_EDIT",         "${HPSS_PATH_SLASH_BIN}/acl_edit",
          NULL},
        { "HPSS_EXEC_CDSCP",            "${HPSS_PATH_SLASH_BIN}/cdscp",
          NULL},
        { "HPSS_EXEC_DELOG",            "${HPSS_PATH_BIN}/hpss_delog",
          NULL},
        { "HPSS_EXEC_RECLAIM",          "${HPSS_PATH_BIN}/reclaim",
          NULL},
        { "HPSS_EXEC_REPACK",           "${HPSS_PATH_BIN}/repack",
          NULL},
/*
 ******************************************************************************
 * Logging Unix files
 *
 *      HPSS_PATH_LOG                   - unix path name for logging files
```

```
 *      HPSS_LOG_MSG_LEN                - split syslog messages which exceed this length
 ************************************************************************
 */
        { "HPSS_PATH_LOG",              "${HPSS_PATH_VAR}/log", NULL},
        { "HPSS_LOG_MSG_LEN",           "260000"              , NULL},
/*
 ************************************************************************
 * Accounting Unix files
 *
 *      HPSS_PATH_ACCT                  - unix path name for accounting files
 *      HPSS_UNIX_ACCT_CHECKPOINT       - checkpoint file
 *      HPSS_UNIX_ACCT_REPORT           - report file
 *      HPSS_UNIX_ACCT_COMMENTARY       - commentary file
 ************************************************************************
 */
        { "HPSS_PATH_ACCT",             "${HPSS_PATH_VAR}/acct",
          NULL},
        { "HPSS_UNIX_ACCT_CHECKPOINT",  "${HPSS_PATH_ACCT}/acct_checkpoint",
          NULL},
        { "HPSS_UNIX_ACCT_REPORT",      "${HPSS_PATH_ACCT}/acct_report",
          NULL},
        { "HPSS_UNIX_ACCT_COMMENTARY",  "${HPSS_PATH_ACCT}/acct_commentary",
          NULL},
/*
 ************************************************************************
 * MPS Unix files
 *
 *      HPSS_PATH_MPS                   - unix path name for MPS files
 *      HPSS_UNIX_MPS_REPORT            - report file
 ************************************************************************
 */
        { "HPSS_PATH_MPS",              "${HPSS_PATH_VAR}/mps", NULL},
        { "HPSS_UNIX_MPS_REPORT",       "",                     NULL},
/*
 ************************************************************************
 * Gatekeeper Unix files
 *
 *      HPSS_PATH_GK                    - unix path name for Gatekeeping files
 *      HPSS_UNIX_GK_SITE_POLICY        - site policy file
 ************************************************************************
 */
        { "HPSS_PATH_GK",               "${HPSS_PATH_VAR}/gk",  NULL},
        { "HPSS_UNIX_GK_SITE_POLICY",   "${HPSS_PATH_GK}/gksitepolicy",
          NULL},
/*
 ************************************************************************
 *  Database Info
 *
 *      HPSS_GLOBAL_DB_NAME     - Default name of the Global Data Base
 *      HPSS_SUBSYS_DB_NAME     - Default name for the Subsystem Data Base
 *      HPSS_MM_SCHEMA_NAME     - Default schema name
 *      HPSS_MM_STMT_CACHE      - Sets the length of the MMLIB statement cache
 *      HPSS_SERVER_DB_GROUP    - DB auth group identity used by HPSS servers
 *      HPSS_SERVER_DB_KEYTAB   - DB connection keytab used by HPSS servers
 ************************************************************************
 */
        { "HPSS_DB_INSTANCE_OWNER",     "hpssdb",               NULL},
        { "HPSS_GLOBAL_DB_NAME",        "cfg",                  NULL},
        { "HPSS_SUBSYS_DB_NAME",        "subsys",               NULL},
```

```
        { "HPSS_MM_SCHEMA_NAME",        "HPSS",                 NULL},
        { "HPSS_MM_STMT_CACHE",         "75",                   NULL},
        { "HPSS_SERVER_DB_GROUP",       "hpsssrvr",             NULL},
        { "HPSS_SERVER_DB_KEYTAB",      "${HPSS_PATH_ETC}/mm.keytab",
          NULL},
/*
 ****************************************************************************
 * Descriptive Names
 *
 *      %H is evaluated by hpss_Getenv to be the short form of the hostname,
 *      the hostname up to but not including the first ".", if any.
 *      We use HPSS_HOST_TMP to fill in the host name in the Startup
 *      Daemon and Mover descriptive names because hpss_Getenv
 *      can only evaluate %H when it is the only content of the environment
 *      variable.  We can't embed %H in the descriptive name.  We don't
 *      use HPSS_HOST for this purpose because the user may redefine
 *      it.
 *
 *      HPSS_DESC_CORE                  - Descriptive name - Core Server
 *      HPSS_DESC_FTPD                  - Descriptive name - FTP Daemon
 *      HPSS_DESC_GK                    - Descriptive name - Gatekeeper Server
 *      HPSS_DESC_HPSSD                 - Descriptive name - Startup Daemon
 *      HPSS_DESC_LS                    - Descriptive name - Location Server
 *      HPSS_DESC_MM                    - Descriptive name - Metadata Monitor
 *      HPSS_DESC_MOUNTD                - Descriptive name - Mount Daemon
 *      HPSS_DESC_MPS                   - Descriptive name - MPS
 *      HPSS_DESC_MVR                   - Descriptive name - Mover
 *      HPSS_DESC_PFSD                  - Descriptive name - PFSD
 *      HPSS_DESC_PVL                   - Descriptive name - PVL
 *      HPSS_DESC_PVR_OPER              - Descriptive name - PVR - Operator
 *      HPSS_DESC_PVR_STK               - Descriptive name - PVR - STK
 *      HPSS_DESC_PVR_STK_RAIT          - Descriptive name - PVR - STK RAIT
 *      HPSS_DESC_PVR_AML               - Descriptive name - PVR - AML
 *      HPSS_DESC_PVR_SCSI              - Descriptive name - PVR - SCSI
 *      HPSS_DESC_RAIT                  - Descriptive name - RAIT Engine
 *      HPSS_DESC_SSMSM                 - Descriptive name - SSM System Manager
 ****************************************************************************
 */
        { "HPSS_HOST_TMP",              "%H",                   NULL},
        { "HPSS_DESC_CORE",             "Core Server",          NULL},
        { "HPSS_DESC_FTPD",             "FTP Daemon",           NULL},
        { "HPSS_DESC_GK",               "Gatekeeper",           NULL},
        { "HPSS_DESC_HPSSD",            "Startup Daemon (${HPSS_HOST_TMP})",    NULL},
        { "HPSS_DESC_LS",               "Location Server",      NULL},
        { "HPSS_DESC_MM",               "Metadata Monitor",     NULL},
        { "HPSS_DESC_MOUNTD",           "Mount Daemon",         NULL},
        { "HPSS_DESC_MPS",              "Migration/Purge Server",
          NULL},
        { "HPSS_DESC_MVR",              "Mover (${HPSS_HOST_TMP})",     NULL},
        { "HPSS_DESC_PVL",              "PVL",                  NULL},
        { "HPSS_DESC_PVR_OPER",         "Operator PVR",         NULL},
        { "HPSS_DESC_PVR_STK",          "STK PVR",              NULL},
        { "HPSS_DESC_PVR_AML",          "AML PVR",              NULL},
        { "HPSS_DESC_PVR_SCSI",         "SCSI PVR",             NULL},
        { "HPSS_DESC_RAIT",             "RAIT Engine (${HPSS_HOST_TMP})",       NULL},
        { "HPSS_DESC_SSMSM",            "SSM System Manager",   NULL},

/*
 ****************************************************************************
```

```
* System Manager Specific
*
*      HPSS_PATH_SSM           - unix path name for data server files
*      HPSS_SSM_ALARMS         - File to store SSM Alarms/Events
*                                NULL -> SM will store internally
*      HPSS_SSM_ALARMS_DISPLAY - Number of SSM Alarms/Events to display/store
*      HPSS_SSM_MAX_IDLE_SYSLOG- Max number of minutes syslog file can
*                                be idle before getting concerned
*      HPSS_SSM_ALARMS_GET     - Number of SSM Alarms/Events to get at one time
*      HPSS_SSM_COUNTRY        - Country for Java internationalization
*      HPSS_SSM_LANGUAGE       - Language for Java internationalization
*      HPSS_SSM_SERVER_LISTEN_PORT
*                              - Port the SM is listening on for client RPCs
*                                If 0, port will be chosen by portmapper
*      HPSS_SSM_TIMING_DEBUG   - Turns on/off ssm timing debug logging
*                                If 0, timing debug will be off
*      HPSS_HELP_FILES_PATH    - HPSS Help files install path
*      HPSS_HELP_URL_TYPE      - HPSS Help files URL type
*      HPSSGUI_SM_HOST_NAME    - host SM is on
*      HPSSADM_SM_HOST_NAME    - host SM is on
*      HPSSGUI_SM_PORT_NUM     - port SM is on
*      HPSSADM_SM_PORT_NUM     - port SM is on
*      HPSSGUI_RPC_PROT_LEVEL  - rpc protection level used for SM communication
*      HPSSADM_RPC_PROT_LEVEL  - rpc protection level used for SM communication
*      HPSSSSM_UI_WAIT_TIME    - Time the GUI will wait at the SM for updates
*      HPSSSSM_UI_MO_RATE      - Time the GUI will wait between MO update tries
*      HPSSSSM_UI_LIST_RATE    - Time the GUI will wait between List update tries
*      HPSSSSM_UI_ALARM_RATE   - Time the GUI will wait between Alarm update tries
*      HPSS_SSM_SEC_MECH       - security mechanism used for SM communication
*      HPSSGUI_USER_CFG_PATH   - Directory which holds GUI config files
*      HPSSADM_USER_CFG_PATH   - Directory which holds ADM config files
*
*      HPSS_SSMUSER_JAVA_POLICY- Java policy file for SSM GUI and hpssadm
*
*      JAVA_CLS1               - location of hpss ssm Java classes
*      HPSS_SSM_CLASSPATH      - runtime search path for Java classes
*
*   The SM attempts to throttle the connection attempts to other servers. It
*   will attempt to reconnect to each server every
*   HPSS_SM_SRV_CONNECT_INTERVAL_MIN seconds until the number of failures for
*   that server has reached HPSS_SM_SRV_CONNECT_FAIL_COUNT. After the failure
*   count has been reached the SM will only try to reconnect to the server
*   every HPSS_SM_SRV_CONNECT_INTERVAL_MAX seconds until a successful i
*   connection is made at which time the connection interval for the server
*   will be set back to HPSS_SM_SRV_CONNECT_INTERVAL_MIN.
*
*      HPSS_SM_SRV_CONNECT_FAIL_COUNT  - Number of connection failures to a
*                      server before the HPSS_SM_SRV_CONNECT_INTERVAL_MAX
*                      interval takes affect
*      HPSS_SM_SRV_CONNECT_INTERVAL_MIN - Interval between attempting server
*                      connections when HPSS_SM_SERVER_CONNECT_FAIL_COUNT
*                      has not yet been reached (seconds)
*      HPSS_SM_SRV_CONNECT_INTERVAL_MAX - Interval between server  connections
*                      when HPSS_SM_SERVER_CONNECT_FAIL_COUNT has been
*                      reached without a successful connection (seconds)
*      HPSS_SM_SRV_LIST_UPDATE_INTERVAL - Frequency at which the SM updates
*                      the server list (seconds)
*      HPSS_SM_SRV_MONITOR_THREADS      - Number of threads created to monitor
*                      server connections
```

```
*      HPSS_SM_SRV_QUEUE_SIZE          - Request Queue Size used by the System
*                      Manager server interface - default of 0 means that
*                      there will be 20 slots in the server interface
*                      request queue to be used when the server interface
*                      threadpool is completely full. The queue is used to hold
*                      RPC requests from servers until a thread is available
*                      to process the request.
*                      Note that if the request queue has any entries in it
*                      it means that all the threads in the server thread
*                      pool are busy and the SM response will be degraded.
*                      If this happens then it would be good to increase
*                      the number of threads available to the server interface
*                      using the HPSS_SM_SRV_TPOOL_SIZE variable.
*                      Increasing the size of the queue will not help with
*                      performance.
*      HPSS_SM_SRV_TPOOL_SIZE          - Thread Pool Size used by the System
*                      Manager server interface. If the Thread Pool is
*                      exhausted then server RPC requests will be queued in the
*                      server RPC Request Queue to wait for a thread to become
*                      available. When the thread pool is exhausted SM
*                      performance may be degraded. Increase this value if
*                      that is the case.  Typically 1 thread per HPSS server
*                      should be adequate. But a few extra wouldn't hurt.
*      HPSS_SM_SRV_MAX_CONNECTIONS      - Number of HPSS server connections
*                      to maintain at once. If this number of connections is
*                      exceeded, then old connections will be close to
*                      maintain this number of connections
*
************************************************************************
*/
       { "HPSS_PATH_SSM",              "${HPSS_PATH_VAR}/ssm", NULL},
       { "HPSS_SSM_ALARMS",            NULL,                   NULL},
       { "HPSS_SSM_ALARMS_DISPLAY",    "2000",                 NULL},
       { "HPSS_SSM_MAX_IDLE_SYSLOG",   "60",                   NULL},
       { "HPSS_SSM_ALARMS_GET",        "500",                  NULL},
       { "HPSS_SSM_COUNTRY",           "US",                   NULL},
       { "HPSS_SSM_LANGUAGE",          "en",                   NULL},
       { "HPSS_SSM_SERVER_LISTEN_PORT","0",                    NULL},
       { "HPSS_SSM_TIMING_DEBUG",      "0",                    NULL},
       { "HPSS_HELP_FILES_PATH", "${HPSS_PATH_INSTALL}/ssmhelp", NULL},
       { "HPSS_HELP_URL_TYPE",         "file:",                NULL},
       { "HPSSGUI_SM_HOST_NAME",       "${HPSS_HOST}",         NULL},
       { "HPSSADM_SM_HOST_NAME",       "${HPSS_HOST}",         NULL},
       { "HPSSGUI_SM_PORT_NUM",        "536870913:1",          NULL},
       { "HPSSADM_SM_PORT_NUM",        "536870913:1",          NULL},
       { "HPSSGUI_RPC_PROT_LEVEL",     "${HPSS_RPC_PROT_LEVEL}",
         NULL},
       { "HPSSADM_RPC_PROT_LEVEL",     "${HPSS_RPC_PROT_LEVEL}",
         NULL},
       { "HPSSSSM_UI_WAIT_TIME",       "${SSM_UI_WAIT_TIME}",  NULL},
       { "HPSSSSM_UI_MO_RATE",         "${SSM_UI_MO_RATE}",    NULL},
       { "HPSSSSM_UI_LIST_RATE",       "${SSM_UI_LIST_RATE}",  NULL},
       { "HPSSSSM_UI_ALARM_RATE",      "${SSM_UI_ALARM_RATE}", NULL},
       { "HPSS_SSM_SEC_MECH",          NULL,                   NULL},
       { "HPSSGUI_USER_CFG_PATH",      "${HOME}/hpss-ssm-prefs",
         NULL},
       { "HPSSADM_USER_CFG_PATH",      "${HOME}/hpss-ssm-prefs",
         NULL},
       { "HPSS_SSMUSER_JAVA_POLICY",
```

```
          "${HPSS_PATH_VAR}/ssm/java.policy.ssmuser",          NULL},
        { "JAVA_CLS1",
          "${HPSS_ROOT}/bin/hpss.jar",                         NULL},
        { "HPSS_SSM_CLASSPATH",
          "${JAVA_CLS1}",                                      NULL},
        { "HPSS_SM_SRV_CONNECT_FAIL_COUNT",     "3",           NULL},
        { "HPSS_SM_SRV_CONNECT_INTERVAL_MIN",   "20",          NULL},
        { "HPSS_SM_SRV_CONNECT_INTERVAL_MAX",   "60",          NULL},
        { "HPSS_SM_SRV_LIST_UPDATE_INTERVAL",   "5",           NULL},
        { "HPSS_SM_SRV_MONITOR_THREADS",        "10",          NULL},
        { "HPSS_SM_SRV_QUEUE_SIZE",             "0",           NULL},
        { "HPSS_SM_SRV_TPOOL_SIZE",             "100",         NULL},
        { "HPSS_SM_SRV_MAX_CONNECTIONS",        "50",          NULL},

/*
 ****************************************************************************
 * CLAPI Specific
 *
 *      HPSS_API_HOSTNAME           - Used to control the network interface
 *                                    selected for data transfers.
 *      HPSS_API_DEBUG_PATH         - Used to direct debug output messages
 *      HPSS_API_MAX_CONN           - Defines the number of connections that
 *                                    are supported by the Client API within
 *                                    a single client process
        HPSS_API_MAX_OPEN           - Defines the maximum number of open files
 *      HPSS_API_DEBUG              - Used to enable debug messages
 *      HPSS_API_RETRIES            - Used to control the number of retries
 *                                    when operations fail with a "retryable"
 *                                    return code
 *      HPSS_API_BUSY_DELAY         - Used to control the number of seconds
 *                                    to delay between retry attempts.
 *      HPSS_API_BUSY_RETRIES       - Used to control the number of retries
 *                                    to be performed when a request fails
 *                                    because the Core Server does not
 *                                    currently have an available thread
 *                                    to handle the request
 *      HPSS_API_TOTAL_DELAY        - Used to control the number of seconds
 *                                    to continue retrying a request
 *      HPSS_API_LIMITED_RETRIES    - Used to control the number of retry
 *                                    attempts before a limited retry error
 *                                    operation fails
 *      HPSS_API_DMAP_WRITE_UPDATES - Used to control the frequency of the
 *                                    cache invalidates that are issued to
 *                                    the DMAPI file system while writing
 *                                    to a file that is mirrored in HPSS
 *      HPSS_API_REUSE_CONNECTIONS  - Used to control whether TCP/IP
 *                                    connections are to left open as long
 *                                    as an HPSS file is open or are closed
 *                                    after each read or write operation.
 *      HPSS_API_USE_PORT_RANGE     - Used to control whether HPSS Movers
 *                                    should the configured port range
 *                                    when making TCP/IP connection for
 *                                    read or write operations for the client
 *      HPSS_API_RETRY_STAGE_INP    - Used to control whether retries are
 *                                    attempted on when trying to open files
 *                                    in a Class of Service that is
 *                                    configured for background staging on
 *                                    open
 *      HPSS_API_DISABLE_CROSS_REALM- Used to control cross-realm traversal
```

```
 *      HPSS_API_DISABLE_JUNCTIONS  - Used to control junction traversal
 *      HPSS_API_AUTHN_MECH         - Used to control the select of an
 *                                    authentication mechanism
 *      HPSS_API_RPC_PROT_LEVEL     - Used to control the select of an
 *                                    RPC protection level
 *
 *      HPSS_API_SAN3P              - Used to control whether SAN3P is
 *                                    on or off.
 *      HPSS_API_TRANSFER_TYPE      - Used to control the transfer type.
 *          -- TCP         - TCP Protocol
 *          -- MVRSELECT   - Mover selects protocol
 *
 *      HPSS_API_OBJ_BUF_LIMIT      - Maximum number of objects that can
 *                                    be returned.
 *      HPSS_API_XML_BUF_LIMIT      - Maximum number of xml strings that can be
 *                                    returned.
 *      HPSS_API_XMLOBJ_BUF_LIMIT   - Maximum number of xml/object entries that
 *                                    can be returned.
 *      HPSS_API_XMLSIZE_LIMIT      - Maximum length of an XML string buffer.
 *      HPSS_API_XMLREQUEST_LIMIT   - Maximum length of an XML string request.
 *                                    (xmlsize * number of xml buffers)
 *
 *
 ***************************************************************************
 */
        { "AIXTHREAD_COND_DEBUG",       "OFF",                  NULL},
        { "HPSS_API_HOSTNAME",          "${HPSS_HOST}",         NULL},
        { "HPSS_API_DEBUG_PATH",        "stdout",               NULL},
        { "HPSS_API_MAX_CONN",          "0",                    NULL},
        { "HPSS_API_MAX_OPEN",          "4096",                 NULL},
        { "HPSS_API_DEBUG",             "0",                    NULL},
        { "HPSS_API_RETRIES",           "4",                    NULL},
        { "HPSS_API_BUSY_DELAY",        "15",                   NULL},
        { "HPSS_API_BUSY_RETRIES",      "3",                    NULL},
        { "HPSS_API_TOTAL_DELAY",       "0",                    NULL},
        { "HPSS_API_LIMITED_RETRIES",   "1",                    NULL},
        { "HPSS_API_DMAP_WRITE_UPDATES","20",                   NULL},
        { "HPSS_API_REUSE_CONNECTIONS", "0",                    NULL},
        { "HPSS_API_USE_PORT_RANGE",    "0",                    NULL},
        { "HPSS_API_RETRY_STAGE_INP",   "1",                    NULL},
        { "HPSS_API_DISABLE_CROSS_REALM","0",                   NULL},
        { "HPSS_API_DISABLE_JUNCTIONS", "0",                    NULL},
        { "HPSS_API_AUTHN_MECH",        "${HPSS_CLIENT_AUTHN_MECH}",
          NULL},
        { "HPSS_API_RPC_PROT_LEVEL",    "${HPSS_RPC_PROT_LEVEL}",
          NULL},
        { "HPSS_API_SAN3P",             "on",                    NULL},
        { "HPSS_API_TRANSFER_TYPE",     "MVRSELECT",             NULL},
        { "HPSS_API_OBJ_BUF_LIMIT",     "4096",                  NULL},
        { "HPSS_API_XML_BUF_LIMIT",     "131072",                NULL},
        { "HPSS_API_XMLOBJ_BUF_LIMIT",  "2048",                  NULL},
        { "HPSS_API_XMLSIZE_LIMIT",     "131072",                NULL},
        { "HPSS_API_XMLREQUEST_LIMIT",  "204800",                NULL},
/*
 ***************************************************************************
 * HDM Specific
 *
 *      HPSS_PATH_HDM      - The HDM's base path
 ***************************************************************************
```

```
 */
       { "HPSS_PATH_HDM",              "${HPSS_PATH_VAR}/hdm/hdm1", NULL},
/*
 ************************************************************************
 * Core Server Specific
 *
 *      HPSS_CORE_LARGE_SEG_THRESHOLD  - Defines a "large" tape segment
 *      HPSS_CORE_LOG_XFER_PERF        - "1" (or "on") to enable internal
 *                                       transfer performance logging
 *      HPSS_CORE_REPACK_OUTPUT        - "on" to segregate repack tapes
 *      HPSS_CORE_DISKSEGCACHE_MEM_PERCENT
 *                                     - max percentage of total system memory
 *                                       occupied by the disk segment cache
 *      HPSS_CORE_TAPE_CACHE_IDLE_TIME - Tape VV MD cache retention time (secs)
 *      HPSS_CORE_SEG_CACHE_IDLE_TIME  - Segment MD cache retention time (secs)
 *      HPSS_CORE_ROTATE_TAPE_PVRS     - "on" to enable PVR rotation
 *
 *      The HPSS_ACCT_xx environment variables are used by the accounting
 *      code inside the core server.  If the default values are changed
 *      here, they should also be changed in bfs_acct.h.
 *
 *      HPSS_ACCT_DELAY                - Delay between accounting thread loops.
 *      HPSS_ACCT_RECS_BW_RAW          - Number of acctlogbandwidth records
 *                                       to select per transaction.
 *      HPSS_ACCT_RECS_CAP_RAW         - Number of acctlogcapacity records
 *                                       to select per transaction.
 *      HPSS_CORE_CREATE_ALLOW_SUBTYPE - Skip tape subtype check when creating
 *                                       tape resources.
 *      HPSS_CORE_TAOS_SCHEDULE_PERCENT- Percent of queue that must be unscheduled
 *                                       before calling TAOS.
 *      HPSS_CORE_VERIFY_SECTION_LIMIT - Number of sections to group together for
 *                                       verify requests
 ************************************************************************
 */
       { "HPSS_CORE_LARGE_SEG_THRESHOLD",      "300", NULL},
       { "HPSS_CORE_LOG_XFER_PERF",            "0",   NULL},
       { "HPSS_CORE_REPACK_OUTPUT",            "on",  NULL},
       { "HPSS_CORE_DISKSEGCACHE_MEM_PERCENT", "1.0", NULL},
       { "HPSS_CORE_TAPE_CACHE_IDLE_TIME",     "300", NULL},
       { "HPSS_CORE_SEG_CACHE_IDLE_TIME",      "60",  NULL},
       { "HPSS_CORE_ROTATE_TAPE_PVRS",         "off", NULL},
       { "HPSS_CORE_RUMBLE_THREADS",           "50",  NULL},

       { "HPSS_ACCT_DELAY",                    "10",     NULL},
       { "HPSS_ACCT_RECS_BW_RAW",              "100000", NULL},
       { "HPSS_ACCT_RECS_CAP_RAW",             "100000", NULL},
       { "HPSS_CORE_CREATE_ALLOW_SUBTYPE",     "FALSE",  NULL},
       { "HPSS_CORE_TAOS_SCHEDULE_PERCENT",    "50.0",   NULL},
       { "HPSS_CORE_VERIFY_SECTION_LIMIT",     "50",   NULL},

/*
 ************************************************************************
 * LOG Specific
 *
 *      HPSSLOG_PERROR       - Also log messages to stderr
 *      HPSS_INFRA_LOG_TYPES - Default types of infrastructure messages
 *                             to be logged
 *      HPSS_INFRA_LOG_CONF  - The infrastructure logging configuration file
 *
```

```
 ****************************************************************************
 */
        { "HPSSLOG_PERROR",             NULL,                   NULL},
        { "HPSS_INFRA_LOG_TYPES",       "CS_ALARM:CS_EVENT",    NULL},
        { "HPSS_INFRA_LOG_CONF",        "${HPSS_PATH_TMP}/$$.log.conf",
          NULL},
/*
 ****************************************************************************
 * LS Specific
 *      LS_DISABLE_LOCAL_CACHE  - Set this to disable the ls cache
 *      LS_LOCAL_CACHE_FILENAME - Set the name of the cache file
 ****************************************************************************
 */
        { "LS_DISABLE_LOCAL_CACHE",  NULL,             NULL},
        { "LS_LOCAL_CACHE_FILENAME", ".hpss_ls_cache", NULL},

/*
 * GHI Specific
 *
 *      HPSS_GHI_PATH      - unix pathname for HPSS/GHI I/F files
 *      HPSS_GHI_LOG_PATH  - GHI path for log files
 *      HPSS_GHI_ETC_PATH  - GHI path for config files
 *      HPSS_GHI_TMP_PATH  - GHI path for temporary files
 *      HPSS_GHI_CONF      - GHI configuration file
 *      HPSS_GHI_FSCONF    - GHI file system config file
 ****************************************************************************
 */
        { "HPSS_GHI_PATH",      "${HPSS_PATH_VAR}/ghi",                 NULL},
        { "HPSS_GHI_LOG_PATH",  "${HPSS_GHI_PATH}/log",                 NULL},
        { "HPSS_GHI_ETC_PATH",  "${HPSS_GHI_PATH}/etc",                 NULL},
        { "HPSS_GHI_TMP_PATH",  "${HPSS_GHI_PATH}/tmp",                 NULL},
        { "HPSS_GHI_CONF",      "${HPSS_GHI_ETC_PATH}/ghi.conf",        NULL},
        { "HPSS_GHI_FSCONF",    "${HPSS_GHI_ETC_PATH}/ghi_%s.conf",     NULL},

/*
 ****************************************************************************
 * FTPD Specific
 *
 *      HPSS_PATH_FTP          - FTP daemon ./etc pathname
 *      HPSS_FTPD_CONTROL_PORT - FTP daemon control default port ID
 *      HPSS_FTP_RESERVED      - FTP reserved port IDs
 *      HPSS_FTP_BLOCK_SIZE    - FTP block size
 *      HPSS_FTP_MAXUSERS      - FTP maximum number of connections
 ****************************************************************************
 */
        { "HPSS_PATH_FTP",              "${HPSS_PATH_VAR}/ftp", NULL},
        { "HPSS_FTPD_CONTROL_PORT",     "4021",                 NULL},
        { "HPSS_FTP_RESERVED",          "1025",                 NULL},
        { "HPSS_FTP_BLOCK_SIZE",        "4",                    NULL},
        { "HPSS_FTP_MAXUSERS",          "MAXUSERS",             NULL},

/*
 ****************************************************************************
 * RAIT Engine Specific
 *
 *      HPSS_RAIT_BUFFER_COUNT      - Number of buffers used per transfer
 *      HPSS_RAIT_TASK_THREAD_COUNT - Number of parity threads per transfer
 ****************************************************************************
 */
```

```
        { "HPSS_RAIT_BUFFER_COUNT",      "6",                   NULL},
        { "HPSS_RAIT_TASK_THREAD_COUNT", "6",                   NULL},

/*
 ************************************************************************
 * MPS Specific
 *
 *      HPSS_MPS_PURGE_PARALLELISM      - Number of purge worker threads per
 *                                        SClass
 *      HPSS_MPS_MIGR_PURGE_PAUSE_SECS  - Number of seconds to wait before
 *                                        starting migration/purge on MPS
 *                                        startup.

 ************************************************************************
 */
        { "HPSS_MPS_PURGE_PARALLELISM",      "2",               NULL},
        { "HPSS_MPS_MIGR_PURGE_PAUSE_SECS",  "300",             NULL},

/*
 ************************************************************************
 * MVR Specific
 *
 *      HPSS_MVR_DISABLE_RESERVATIONS   - Disable SCSI reservations for the
 *                                        devices that correspond to the
 *                                        specified comma separated list of
 *                                        HPSS device identifiers.
 *      HPSS_MVR_DEV_PROGRESS_SECS      - Number of seconds between checks
 *                                        for registered updates for device
 *                                        bytes read/written.
 *      HPSS_MVR_LOCATE_BLK_THRESH      - Within this block threshold the tape
 *                                        mover will read the section header
 *                                        rather than locate directly to data
 ************************************************************************
 */
        { "HPSS_MVR_DISABLE_RESERVATIONS",   NULL,               NULL},
        { "HPSS_MVR_DEV_PROGRESS_SECS",      NULL,               NULL},
        { "HPSS_MVR_LOCATE_BLK_THRESH",      "0",                NULL},

/*
 ************************************************************************
 * Security Registry & Service Location Specific
 *
 *      HPSS_AUTHZ_SERVICE_CONF  - File for valid authorization mechanisms
 *      HPSS_SEC_EP_CONF         - File containing the local endpoints
 *      HPSS_KRB5_AUTHN_MECH     - Kerberos authentication mechanism
 *      HPSS_KRB5_KEYTAB_FILE    - The path for the kerberos keytab file
 *      HPSS_UNIX_AUTHN_MECH     - HPSS Unix authentication mechanism
 *      HPSS_UNIX_KEYTAB_FILE    - The path for the HPSS Unix keytab file
 *      HPSS_PRIMARY_AUTHN_MECH  - The primary authentication mechanism to use
 *      HPSS_CLIENT_AUTHN_MECH   - The client authentication method to use
 *      HPSS_SEC_SITE_CONF       - File containing connect information for
 *                                 local security registry and location
 *                                 service
 *      HPSS_SEC_MUTUAL_AUTH     - Whether to support mutual authentication (true/false)
 *      HPSS_AUTHN_TYPES         - Supported authentication types
 *      HPSS_AUTHZ_TYPES         - Supported authorization types
 *      HPSS_SITE_LOCATION       - Site Location
 *      KRB5_INSTALL_PATH        - Kerberos installation path
 *                                 no default - platform dependent
```

```
 *      KRB5_KDC_DIR             - Kerberos directory containing local config
 *                                 files for KDC
 *      KRB5_KDC_HOST            - Host for Kerberos KDC (just used by mkhpss)
 *
 ***************************************************************************
 */
        { "HPSS_SEC_REALM_ADMIN",       "admin/admin",          NULL},
        { "HPSS_KRB5_AUTHN_MECH",       "krb5",                 NULL},
        { "HPSS_KRB5_KEYTAB_FILE",
                                "auth_keytab:${HPSS_PATH_ETC}/hpss.keytab",
          NULL},
        { "HPSS_UNIX_AUTHN_MECH",       "unix",                 NULL},
        { "HPSS_UNIX_KEYTAB_FILE",
                                "auth_keytab:${HPSS_PATH_ETC}/hpss.unix.keytab",
          NULL},
        { "HPSS_PRIMARY_AUTHN_MECH",    "${HPSS_UNIX_AUTHN_MECH}",
          NULL},
        { "HPSS_PRIMARY_AUTHENTICATOR", "${HPSS_UNIX_KEYTAB_FILE}",
          NULL},
        { "HPSS_CLIENT_AUTHN_MECH",     "${HPSS_PRIMARY_AUTHN_MECH}",
          NULL},
        { "HPSS_AUTHZ_SERVICE_CONF",    "${HPSS_PATH_ETC}/authz.conf",
          NULL},
        { "HPSS_SEC_EP_CONF",           "${HPSS_PATH_ETC}/ep.conf",
          NULL},
        { "HPSS_SEC_SITE_CONF",         "${HPSS_PATH_ETC}/site.conf",
          NULL},
        { "HPSS_SEC_MUTUAL_AUTH",       "TRUE",                 NULL},
        { "KRB5_CONFIG",                "${HPSS_PATH_SLASH_ETC}/krb5.conf",
          NULL},
        { "HPSS_AUTHN_TYPES",           "krb5,unix",            NULL},
        { "HPSS_AUTHZ_TYPES",           "ldap,unix",            NULL},

        { "HPSS_UNIX_AUTHN_MASTER_KEYFILE",
          "${HPSS_PATH_ETC}/unix.master.key",
          NULL},
        { "HPSS_SITE_LOCATION",         "USA",                  NULL},
        { "KRB5_INSTALL_PATH",          KRB5_INSTALL_PATH,      NULL},
        { "KRB5_KDC_DIR",               "${HPSS_PATH_VAR}/krb5kdc",
          NULL},
        { "KRB5_KDC_HOST",              "",                     NULL},
        { "LDAP_INSTALL_PATH",          LDAP_INSTALL_PATH,      NULL},
        { "HPSS_UNIX_CRED_HOME_DIR",    "/home",                NULL},
        { "HPSS_USE_XREALM_LDAP",       "0",                    NULL},

/*
 ***************************************************************************
 * RPC Specific
 *
 *      HPSS_RPC_PORT_RANGE      - Range of TCP/IP ports to use for RPCs
 *      HPSS_LCG_SERVER_RPT_PORT_RANGE  - Specific port range to be used by
 *                                        Location, Core and Gatekeeper for
 *                                        clients outside firewall
 *      HPSS_RPC_SOCK_SNDBUF_SZ  - The RPC socket send buffer size
 *      HPSS_RPC_SOCK_RCVBUF_SZ  - The RPC socket receive buffer size
 *      HPSS_RPC_SOCK_IO_SZ      - The RPC socket I/O size to be used
 *      HPSS_RPC_SOCK_NDELAY     - The RPC socket Nagle setting to be used
 *      HPSS_RPC_PROG_NUM_RANGE  - The range for RPC program numbers
 *      HPSS_RPC_PROT_LEVEL      - Default RPC protection level to be used
```

```
 *
 ******************************************************************************
 */
        { "HPSS_RPC_PORT_RANGE",         NULL,                    NULL},
        { "HPSS_LCG_SERVER_RPC_PORT_RANGE", NULL,                 NULL},
        { "HPSS_RPC_SOCK_SNDBUF_SZ",     NULL,                    NULL},
        { "HPSS_RPC_SOCK_RCVBUF_SZ",     NULL,                    NULL},
        { "HPSS_RPC_SOCK_IO_SZ",         NULL,                    NULL},
        { "HPSS_RPC_SOCK_NDELAY",        NULL,                    NULL},
        { "HPSS_RPC_PROG_NUM_RANGE",     "0x20000000-0x20000200",NULL},
        { "HPSS_RPC_PROT_LEVEL",         "connect",               NULL},

/*
 ******************************************************************************
 * SCSI PVR Specific
 *
 *      HPSS_SCSI_DIAG             - Log low level trace to aux log in
 *                                   HPSS_PATH_TMP/pvr.<pid>.diag and turn on
 *                                   SCSI layer diagnostics
 *
 *      HPSS_SCSI_AVOID_CROSS_ZONE - Number of times to pause a mount waiting
 *                                   on a pending dismount before going ahead
 *                                   with a cross zone move.
 *
 ******************************************************************************
 */
        { "HPSS_SCSI_DIAG",              "off",                   NULL},
        { "HPSS_SCSI_AVOID_CROSS_ZONE", NULL,                     NULL},

/*
 ******************************************************************************
 * RTM Settings
 *      HPSS_RTM_USE_FULL_PATH
 *          Control full path reporting in RTM (rtmu) and SSM (HPSS GUI
 *          and hpssadm.pl).  Valid values include:
 *          default - Report full path from root of roots in RTM;
 *                    Report relative path in SSM.
 *          off     - Report relative path for both RTM and SSM.
 *          on      - Report full path from root of roots for both RTM
 *                    and SSM.
 *          Notes:
 *             (1) Full path reporting can be a more expensive operation,
 *                 with an additional request to determine the full path,
 *                 but caching is used to mitigate follow-on requests.
 *             (2) For a change to take effect in SSM (HPSS GUI and
 *                 hpssadm.pl), the SSM server must be restarted.
 ******************************************************************************
 */
        { "HPSS_RTM_USE_FULL_PATH",      "default",               NULL},

/*
 ******************************************************************************
 * Installation & Miscellaneous
 *
 *      HPSS_PATH_ADM            - Path where administrative files are placed
 *      HPSS_PATH_CORE           - Path where subsystem core files are placed
 *      HPSS_PATH_TMP            - Path where temporary files are placed
 *      HPSS_PATH_ETC            - Path where runtime config files are placed
 *      HPSS_ENV_CONF            - The path to the environment override file
```

```
 *      HPSS_HPSS_CONF            - The path to the HPSS configuration file
 *      HPSS_COPYRIGHT            - File containing HPSS copy right info
 *      HPSS_MAGIC_CONF_PATH      - Path to magic's configuration file
 *      HPSS_PTHREAD_STACK        - Minimum stack size for HPSS pthreads
 *      HPSS_PTHREAD_STACK_MAX    - Maximum stack size for HPSS pthreads
 *      HPSS_SAN3P_LINUX_DMMPATH - Use Linux DM Multipath for SAN3P
 ************************************************************************
 */
       { "HPSS_PATH_ADM",              "${HPSS_PATH_VAR}/adm", NULL},
       { "HPSS_PATH_CORE",             "${HPSS_PATH_ADM}/core",NULL},
       { "HPSS_PATH_TMP",              "${HPSS_PATH_VAR}/tmp", NULL},
       { "HPSS_PATH_ETC",              "${HPSS_PATH_VAR}/etc", NULL},
       { "HPSS_PATH_CRED",             "${HPSS_PATH_VAR}/cred", NULL},
       { "HPSS_ENV_CONF",              "${HPSS_PATH_ETC}/env.conf",
         NULL},
       { "HPSS_HPSS_CONF",             "${HPSS_PATH_ETC}/HPSS.conf",
         NULL},
       { "HPSS_COPYRIGHT",             "${HPSS_ROOT}/copyright",
         NULL},
       { "HPSS_MAGIC_CONF_PATH",       "${HPSS_PATH_ETC}/magic.conf",
         NULL},
       { "HPSS_PTHREAD_STACK",         NULL,                    NULL},
       { "HPSS_PTHREAD_STACK_MAX",     NULL,                    NULL},
       { "HPSS_SAN3P_LINUX_DMMPATH",   "off",                   NULL},
/*
 ************************************************************************
 * HPSS Group names
 *      HPSS_GROUP             - HPSS group name
 *      HPSS_GROUP_DB          - HPSS DB group name
 *      HPSS_GROUP_LDAP        - HPSS LDAP group name
 *      HPSS_GROUP_SERVER      - HPSS Server group name
 *      HPSS_GROUP_CLIENT      - HPSS Client group name
 *      HPSS_GROUP_SYSTEM      - HPSS group name for 'local' 'system'
 *
 ************************************************************************
 */
       { "HPSS_GROUP",             "${HPSS_GRP_NAME}",         NULL},
       { "HPSS_GROUP_DB",          "hpssdb",                   NULL},
       { "HPSS_GROUP_LDAP",        "ldap",                     NULL},
       { "HPSS_GROUP_SERVER",      "${HPSS_GRP_NAME_SERVER}",  NULL},
       { "HPSS_GROUP_CLIENT",      "${HPSS_GRP_NAME_CLIENT}",  NULL},
       { "HPSS_GROUP_SYSTEM",      "system",                   NULL},
/*
 ************************************************************************
 * HPSS Group Default IDs
 *      Used by mkhpss to create HPSS associated GROUPS.  These may need to
 *      be adjusted for NIS environments before running mkhpss/HPSS.  All
 *      GIDs must be available and consistant across all server/mover nodes
 *
 *      HPSS_GID          - HPSS group gid
 *      HPSS_GID_DB       - HPSS DB group gid
 *      HPSS_GID_LDAP     - HPSS DB group gid
 *      HPSS_GID_SERVER   - HPSS Server group gid
 *      HPSS_GID_CLIENT   - HPSS Client group gid
 *      HPSS_GID_SYSTEM   - HPSS group gid for 'local' 'system'
 *
 ************************************************************************
 */
       { "HPSS_GID",               NULL,                       NULL},
```

```
        { "HPSS_GID_DB",            NULL,                          NULL},
        { "HPSS_GID_LDAP",          NULL,                          NULL},
        { "HPSS_GID_SERVER",        NULL,                          NULL},
        { "HPSS_GID_CLIENT",        NULL,                          NULL},
        { "HPSS_GID_SYSTEM",        "0",                           NULL},
/*
 ******************************************************************************
 * HPSS User Names
 *      Similar to PRINCIPAL list above, but used by mkhpss to setup
 *      additional non-server ids.  PRINCIPAL implies DCE/Kerberos, while
 *      USER relates more to UNIX based systems.
 *
 *      HPSS_USER            - Already defined
 *      HPSS_USER_DB         - User name for HPSS DB
 *      HPSS_USER_LDAP       - User name for LDAP server
 *      HPSS_USER_ROOT       - User name for HPSS local 'root'
 *
 ******************************************************************************
 */

/* Non-Server User Names */
     /* { "HPSS_USER",             "Already defined",           NULL}, */
        { "HPSS_USER_DB",          "${HPSS_DB_INSTANCE_OWNER}", NULL},
        { "HPSS_USER_LDAP",        "ldap",                      NULL},
        { "HPSS_USER_LDAP_DB",     "ldapdb",                    NULL},
        { "HPSS_USER_ROOT",        "root",                      NULL},
/*
 ******************************************************************************
 * HPSS Server Unix UID's
 *      Used by mkhpss when setting up HPSS accounts/authentication info.
 *      Note that all UIDs must be available and consistant across all server,
 *      mover, and VFS client nodes.  These must be adjusted before the code
 *      is compiled and mkhpss/HPSS is run.
 *
 *      HPSS_UID          - Unix UID for HPSS
 *      HPSS_UID_DB       - Unix UID for HPSS DB2 Instance
 *      HPSS_UID_LDAP     - Unix UID for HPSS LDAP
 *      HPSS_UID_LDAP_DB  - Unix UID for HPSS LDAP DB
 *      HPSS_UID_ROOT     - Unix UID for HPSS local 'root'
 *
 ******************************************************************************
 */

/* Non-Server UIDs */
        { "HPSS_UID",              NULL,                          NULL},
        { "HPSS_UID_DB",           NULL,                          NULL},
        { "HPSS_UID_LDAP",         NULL,                          NULL},
        { "HPSS_UID_LDAP_DB",      NULL,                          NULL},
        { "HPSS_UID_ROOT",         "0",                           NULL},

};
```

Appendix F. The /var/hpss files

The `/var/hpss` directory tree is the default location of a number of HPSS configuration files, log files, and other files needed by the servers.

The directories and configuration files can be created with the **mkhpss** utility or hand-created. Be very careful when using **mkhpss** utility as selecting the wrong option can damage the already partially configured HPSS system. The log files and other files are created by the servers.

The directories in `/var/hpss` include:

acct. The usual directory to hold accounting report files and checkpoint files. This location is specified in the accounting policy. The report and checkpoint files are created by the accounting report utility; see the *Generating an accounting report* section in the *HPSS Management Guide*. The administrator must regularly remove unneeded reports to prevent the `/var/hpss` file system from filling.

adm. This directory contains one log file and one subdirectory:

- **hpssd.failed_server.** A file of log entries for server startups and terminations. Created and maintained by the Startup Daemon.
- **core**. The default directory where HPSS servers put "core" files if they terminate abnormally. The system administrator must clean out old core files regularly to avoid filling up `/var/hpss`.

cred. The default directory where Kerberos credential files for HPSS servers are placed.

doc. The default directory where the HPSS documentation is installed.

etc. The default directory where many UNIX configuration files are placed. These files include:

- **HPSS.conf.** A configuration file for ftp and other HPSS client applications. Can be initialized from the template file `/opt/hpss/config/templates/HPSS.conf.tmpl`. See Appendix D: *Appendix D, `HPSS.conf` configuration file* for more details.
- **authz.conf.** Created by **mkhpss**. Lists HPSS SEC libraries and functions for performing initialization of authorization manager: LDAP or UNIX. Can be initialized from the template file `/opt/hpss/config/templates/authz.conf.template`. See the *Security services configuration* section in the *HPSS Management Guide* for details.
- **env.conf.** Created as an empty file by **mkhpss**. Contains site-specific environment variables for HPSS. Some utilities and servers in HPSS may automatically source this file upon startup for necessary environment variables. For others, environment variables like HPSS_PATH_VAR may need to be defined first.
- **ep.conf.** Contains the endpoints of the HPSS Location Servers. Used by HPSS client application programs to determine the address at which to contact HPSS. Created by the utility `/opt/hpss/config/hpss_bld_ep`. This utility should be executed by root after the Location Server has been configured. It takes no arguments.

- **hpss.keytab.** Created by **mkhpss**. Contains the username and password for HPSS servers to use for Kerberos authentication with a keytab file. Can be created manually by using **ktutil** to add each of the HPSS server principals to the file. See your Kerberos documentation for the usage of **ktutil**.

- **hpss.unix.keytab.** Created by **mkhpss**. Contains the username and password for HPSS servers to use for UNIX authentication with a keytab file. Can be created manually by using the **hpss_unix_keytab** utility to add each of the HPSS server principals to the file. See the HPSS man page for the usage of **hpss_unix_keytab**.

- **mm.keytab.** Created by **mkhpss**. The keytab for user *hpss*. Used by utilities which read the DB2 databases. Can be created manually with the `/opt/hpss/config/hpss_mm_keytab` utility. The syntax is

  ```
  hpss_mm_keytab -f /var/hpss/etc/mm.keytab hpss
  ```

  The program will prompt for the UNIX password of user *hpss*. Can be created manually with the `/opt/hpss/config/hpss_mm_keytab` utility. The syntax is

  ```
  hpss_mm_keytab -f /var/hpss/etc/mm.keytab hpss
  ```

  The program will prompt for the UNIX password of user *hpss*.

- **site.conf.** Created by **mkhpss**. Contains the site name, realm name, realm ID, authorization mechanism and authorization URL to utilize. Can be initialized from the template file. `/opt/hpss/config/templates/site.conf.template`. See the *Security services configuration* section in the *HPSS Management Guide* for details.

- **rc.db2**. Created by **mkhpss**. Script for starting DB2. Can be initialized from the template file `/opt/hpss/config/templates/rc.db2.template`. Change the line in the file:

  ```
  export DB2INSTANCE="%<DB2INSTANCE>%"
  ```

  to:

  ```
  export DB2INSTANCE="hpssdb"
  ```

  or to whatever your site has named your DB2 instance.

- **rc.krb**. Created by **mkhpss**. Script for starting the Kerberos servers. Can be initialized from the template file `/opt/hpss/config/templates/rc.krb5.template`. (Looks like you have to replace the lines for KRB5_INSTALL_PATH and KRB5_KDC_PROFILE in here.)

- **passwd.** Created by **mkhpss**. A local HPSS-only password file for use with UNIX authentication and authorization. Optionally, the system password file can be used instead. If a lot of users are added or removed from the system, it may be necessary to remove backups of this file generated by the HPSS user management utilities.

- **group.** Created by **mkhpss**. A local HPSS-only group file for use with UNIX authentication and authorization. Optionally, the system group file can be used instead. If a lot of users are added or removed from the system, it may be necessary to remove backups of this file generated by the HPSS user management utilities.

- **shadow.** Created by **mkhpss**. A local HPSS-only shadow file for use with UNIX authentication and authorization. Optionally, the system shadow file can be used instead. If a lot of users are added

or removed from the system, it may be necessary to remove backups of this file generated by the HPSS user management utilities.

- **unix.master.key.** Created by **mkhpss**. The file can also be generated by running the **hpss_unix_keygen** utility. It contains a hexadecimal key in the format `0x######## 0x########`. The key is used by HPSS during UNIX authentication to provide keytab services. Password hashes are wrapped using the master key and compared to the corresponding entry in the keytab file. The `unix.master.key` file should exist on the core server and also on remote process servers, such as a remote PVR, a Mover, or a RAIT Engine. The original file and all copies must be protected and must be identical.

- **hpss_san3p.conf.** Contains fiber connectivity information for SAN3p devices and clients. This file typically resides on a Mover machine and is used to indicate the association between client machines, using IP addresses, and SAN3p devices. A template can be found in the `./config/templates/hpss_san3p.conf.template` file. See the instructions in the template for details about how to update the file for specific SAN configurations. See the *Devices and drive management* chapter of the *HPSS Management Guide* for more information on SAN3p devices.

ftp. Files for FTP configuration. See the *FTP Daemon configuration* section in the *HPSS Management Guide* for more information.

gk. The recommended directory for the site policy configuration file used by the Gatekeeper if the Gatekeeper is configured to do gatekeeping services. The file is usually named `gksitepolicy`. It is created by the system administrator. See *Section 3.7.3, "Gatekeeper"*.

hpssdb. The directory to hold the DB2 instance configuration information and *CFG* database tables.

krb5kdc. The directory for Kerberos configuration files. These include:

- **kadm5.keytab**. The keytab for the Kerberos administrative user

- **kdc.conf**. Kerberos configuration file. See the Kerberos documentation. See also the template file `/opt/hpss/config/templates/kdc.conf.template`.

log. By default, HPSS log files should reside in `/var/hpss/log`. Because HPSS logs to syslog, the system syslog configuration should be updated to place HPSS logs in this location. See the *Logging* section of the *HPSS Management Guide* for details.

mps. The directory in which the MPS places its migration/purge reports, if it is configured to produce these reports. Reports are generated every 24 hours. The system administrator will need to remove these reports regularly when no longer needed to avoid filling up the `/var/hpss` file system.

ssm. The usual directory for SSM configuration files. These include:

- A file to store alarms and events if SSM is configured to buffer alarm and event messages in a disk file (as opposed to keeping them in memory). This pathname is defined by the HPSS_SSM_ALARMS environment variable.

- An optional subdirectory to hold keytabs for **hpssadm** users on the system. The directory and keytab files are created by the system administrator.

- **login.conf**. (Optional) Contains information required by SSM authentication. A copy of this file is included in `/opt/hpss/bin/hpss.jar` and it should need no customization. However, a template is provided in `/opt/hpss/config/templates/login.conf.template` should sites need it.

- **ssm.conf.** Configuration file for SSM client programs (**hpssgui** and **hpssadm**). Can be initialized from the template file `/opt/hpss/config/templates/ssm.conf.template`. See the instructions inside the template file.

See the *Using SSM* chapter of the *HPSS Management Guide* for details.

tmp. The Startup Daemon uses `/var/hpss/tmp` as the default location for HPSS server lock files, creating a lock file for each server it starts on the node.

HPSS may also write diagnostic log files and disk allocation maps in this directory, when configured to do so. The lock files are very small, but the log files and maps can be very large.

The diagnostic files are named `gasapi.diag`, `secapi.diag`, or `<pid>.diag`. These files can be truncated to free up the disk space they occupy, or they can be removed from the system.

As HPSS runs, it checks whether `/var/hpss/tmp/gasapi.diag` and `/var/hpss/tmp/secapi.diag` exist. If one of them does, HPSS processes will begin appending diagnostic data to the file and continue to do so as long as they continue running.

To truncate one of the `.diag` files (but leave the process writing it running, and possibly writing more data to the file in the future), simply redirect the output of a silent command (that is, one that produces no output) to the file:

```
% /bin/echo -n "" > /var/hpss/tmp/secapi.diag
```

To remove a `.diag` file completely so that HPSS processes will no longer write diagnostic data to it, it is necessary to shut down all processes that have it open for writing. For `secapi.diag` and `gasapi.diag`, this will typically mean shutting down all of HPSS, then removing the file. Once all the standard HPSS servers are shut down through SSM:

```
% rc.hpss -m stop
% rc.hpss -d stop
% rm /var/hpss/tmp/secapi.diag /var/hpss/tmp/gasapi.diag
```

Per-process diagnostic files are generated by sending a HUP signal to the process of interest:

```
% kill -HUP <pid>
```

To truncate `<pid>.diag` and free up the disk space it occupies:

```
% /bin/echo -n "" > /var/hpss/tmp/<pid>.diag
```

To remove `<pid>.diag` completely, determine which server is running with that process id and shut it down, then:

```
% rm /var/hpss/tmp/<pid>.diag
```

If a `.diag` file is simply removed using **rm** while a process still has it open, the directory entry disappears but the disk space occupied is not released. Thus, it is possible to wind up with a disk full of unreachable and unreleasable data. The only way to recover is to find the process that has the phantom files open and shut them down. Once this is done, the orphaned space will be released and returned to the system for reuse.