English

FUJITSU Software BS2000

HSMS V12.0A

Volume 1: Functions, Management and Installation

User Guide

Edition June 2020

Comments… Suggestions… Corrections…

The User Documentation Department would like to know your opinion on this manual. Your feedback helps us to optimize our documentation to suit your individual needs.

Feel free to send us your comments by e-mail to: bs2000services@ts.fujitsu.com.

Certified documentation according to DIN EN ISO 9001:2015

To ensure a consistently high quality standard and user-friendliness, this documentation was created to meet the regulations of a quality management system which complies with the requirements of the standard DIN EN ISO 9001:2015.

Copyright and Trademarks

Table of Contents

HSMS V12.0A 2020-06 Volume 1: Functions, Management and Installation

1 Preface

HSMS (**H**ierarchical **S**torage **M**anagement **S**ystem) is a BS2000 software product for data backup which supports data management on external storage devices in a BS2000 system.

Since BS2000/OSD-BC V9.0, the public storage space of a pubset can be expanded by storage space which a net server makes available remotely in the form of Net-Storage. Here HSMS also supports cataloged Net-Storage files in BS2000.

In HSMS V10.0 and higher, a BS2000 backup server can be configured in the shared pubset network which handles the backup tasks for all other BS2000 hosts in this network. This relieves the load on the productive systems.

HSMS also enables data of networked, remote UNIX systems, network servers or workstations to be saved and, when required, restored. To permit this, these systems must be connected to the local BS2000-UFS (POSIX) via NFS.

Computer centers have to process ever increasing quantities of data. The amount of personnel and volumes required for creating and managing backup copies of the data processed has grown accordingly.
HSMS provides computer center and database administrators with a **backup function** which permits both incremental and partial backup as well as the continuation of backup volumes, and thus helps to save time and space.

In many cases, old data inventories in a system are required only in exceptional cases. Nevertheless they have to be kept on a long-term basis, either to comply with legal regulations or to provide against disaster. To this end, HSMS offers an **archival** (long-term data saving) function that is independent of the backup function. The data saved by the archival function is managed in separate archives independently of the data saved by the backup function.

The demand for online disk storages in particular has increased enormously. This results in considerable costs for devices, floor space and management. On the other hand, as a safeguard against saturation, many disk storages are not used to capacity. The **migration** function offered by HSMS is a mechanism designed to make better use of disk storage capacity: during operation, data that has not been used for some time is moved from the processing level to the background level. Migration can also be started automatically when saturation levels are reached. The data can be recalled to the processing level as required even without the user calling HSMS.

HSMS implements the basic functions of backup, archival and migration in a three-level hierarchy of external storages. The volumes of a BS2000 system – disk storages and magnetic tape cartridges – are assigned to the various hierarchical levels depending on their performance characteristics with regard to storage costs and access time requirements. The normal processing level is assigned an online background level consisting of disk storages and an offline background level consisting of magnetic tape cartridges.

In addition to the single feature pubsets (SF pubsets), HSMS supports the system-managed pubsets (SM pubsets) in a special manner as the background and storage levels with the puset's metainformation are integrated into thiese and backup and migration strategies can be controlled by means of management classes.

With its **data transfer** function, moreover, HSMS provides the option of transferring files, job variables and even catalog entries of BS2000 files to other BS2000 systems or other user IDs, using magnetic tape cartridges. Data transfer is equivalent to the
EXPORT/IMPORT function of ARCHIVE; as a result, volumes created with either HSMS or ARCHIVE are interchangeable.

HSMS includes the software product ARCHIVE. In addition, HSMS optionally interoperates with the software products MAREN and ROBAR.

For BS2000 files, a smooth changeover from ARCHIVE to HSMS is ensured by the compatibility of the archive directory files and backup volumes of the two products.

HSMS is operated via an easy-to-use user interface. HSMS statements are entered via SDF.

HSMS offers a variety of control and inquiry functions, such as:

- restricting tape processing to specially reserved times
- restarting interrupted requests
- obtaining information about the occupancy and utilization of public volume sets.

The files to be processed by the basic functions are determined by means of user-friendly selection mechanisms.

1.1 Objectives and target groups of this manual

The present manual is designed both for nonprivileged users of HSMS and for the HSMS administrator.

Even nonprivileged HSMS users should have a working knowledge of BS2000. The user should be familiar with the most important commands and, in particular, with the BS2000 Data Management System (DMS).

The HSMS administrator should additionally be familiar with the UNIX file system (UFS), system management and computer center organization.

1.2 Summary of contents

This manual is intended to familiarize the user with the HSMS functions and HSMS management. It addresses both nonprivileged BS2000 users and HSMS administrators; sections of special interest for either target group are marked accordingly.

Following an introduction to the basic functions of HSMS, the storage hierarchy and user classes are described. The archives are then presented as the basic management units of HSMS. This is followed by a detailed description of the HSMS functions.

The chapters on management and installation are aimed at the HSMS administrator; they also contain information on the migration function.

In chapter "Interoperation of HSMS with ARCHIVE" you will find information about the most important products HSMS interoperates with. In particular, it contains special information for users who are already familiar with the use of ARCHIVE. Here you will find aids for changing over from ARCHIVE to HSMS as well as overviews of the HSMS statements replacing the familiar ARCHIVE functions.

In the appendix the three sections "Introduction of HSMS in the computer center", "Effect on user applications" and "Notes on the generation of automatic procedures" provide a summary which should be distributed to all computer center users prior to the introduction of HSMS (in particular before the migration function is made available). **The restriction on reproduction therefore does not apply to this part of the appendix.** The appendix also contains information on troubleshooting.

The list of abbreviations and glossary define the most important terms used in this manual.

In the text, references to other manuals are made in abbreviated form and appear in quotation marks. The full titles of these publications can be found under “Related publications”.

There is also an index at the back of the manual.

The following documentation is available to help you learn to use the software product HSMS V12.0A within the BS2000 operating system:

HSMS V12.0A
Hierarchical Storage Management
System Volume 2: Statements
User Guide

Additional product information

Current information, version and hardware dependencies, and instructions for installing and using a product version are contained in the associated Release Notice. These Release Notices are available online at https://bs2manuals.ts.fujitsu.com.

1.3 Changes since the last edition of the manual

The manuals “HSMS V12.0A, Volumes 1 and 2” include the functional enhancements of HSMS V12.0A.

Version backup

In HSMS V12.0A a new type of backup processing is introduced: version backup.

This provides the following features:

1. A definition of a number of backup versions of a file that should be kept, with a version management only for files cataloged in the TSOSCAT of a pubset
2. A mechanism to ensure that a deleted file (possibly accidentally deleted) will be kept in the archive long enough in the previously backed up versions so that the user can take action to reconstruct the data.

For details see section "Version backup".

Keep requests

HSMS provides functionality for keeping completed requests after a host crash or restart. The amount of days that completed request will be kept can be defined with the statement
//MODIFY-HSMS-PARAMETERS OPERATION-CONTROL=*PARAMETERS(KEEP-REQUESTS=…)

For details see section "Keep requests".

1.4 Notational conventions

- In the sample program runs, user input is shown in bold type.
- Statements in the descriptive text which refer to the calling of certain functions cannot normally be entered in this form; rather they indicate the key components of the statement in question. As a rule, additional operands must be specified and the SDF data types are to be replaced by other entries.
- Document references in the text are given in abbreviated form between quotation marks. The full title to which the specified number refers is listed under that number in the related publications section and is accompanied by a brief description.
- References within this manual specify the page in question within the manual and, if appropriate, the name of the section or chapter. References to topics which are dealt with in other manuals specify only the abbreviated name of the manual. You can then consult the index of the specified manual to find the correct location in the text.

2 Basic HSMS application models

The continuously growing amounts of data handled in a computer center have led to increasing overheads for devices, volumes and, last but not least, personnel for storing, saving and managing the data.

HSMS (**H**ierarchical **S**torage **M**anagement **S**ystem) is a software product which serves to

- organize data backup and archival
- optimize the utilization of external high-speed storages in a BS2000 system
- transfer data

HSMS facilitates the saving of data and thus primarily supports the computer center administrator or system administrator. However, a selected set of functions for access to system backup copies, for data archival and for storage space optimization is available to ordinary BS2000 users also.

HSMS is able to process files cataloged in BS2000 (DMS), i.e. files from shared volumes (SF and SM pubsets), private disks, and Net-Storage (files both of the type BS2000 and of the type NODE-FILE).

HSMS can also back up, restore, and archive files from the local BS2000-UFS (POSIX) and remote nodes mounted in it. These files are only accessible via POSIX. They are thus not cataloged in BS2000 DMS.

Nonprivileged users can back up and retrieve files and job variables of other users if they are co-owners (see the “SECOS” manual [16]). Co-ownership in HSMS applies to files, job variables and archive. The co-ownership of an archive is implemented in HSMS by creating a co-owner for the associated archive directory.

HSMS supports large files and volumes (>= 32 GB). The following must be borne in mind here:

- Large files can only be restored to pubsets designed specifically for this purpose.
- Files < 32 GB which are located on pubsets which contain large volumes will be treated as normal files. They may be processed on all pubsets and also in all versions of BS2000.

HSMS V12.0 can be implemented on installations from BS2000 OSD/BC V10.0.

2.1 Basic functions

HSMS offers the user the following basic functions:

- Backup
- Version Backup
- Archival
- Migration
- data transfer on magnetic tape cartridges

With each basic function, HSMS provides easy-to-use features for the selection of the files to be processed and for processing control. HSMS is based on the software product ARCHIVE. The functions which were previously called using ARCHIVE are now also available via HSMS.

Backup

Backup is the precautionary creation, storage and management of copies of the data handled in a computer center. These copies can be used to restore data which has been lost. Data loss can be caused by operator errors such as unintentional deletion or by hardware failures.

Data backup and subsequent recovery can be used for reorganizing the data inventory or for migrating IDs to new or other pubsets. In particular, during reorganization especially the splitting of files among several areas (extends) is corrected.

HSMS saves files and job variables logically: files are read from one or more volumes and stored contiguously on other volumes, i.e. they are saved in logical units (refer to section “Data saving in BS2000”, in the “ARCHIVE” manual [2]).

HSMS permits both system backups of the entire data inventory as well as backups of individual applications (e.g. databases). As an option, HSMS saves all the files specified by the user (full backup) or only those files which are new or have been modified since the last backup (incremental backup) and, with large files, only the updated blocks (partial backup).

All types of backup support magnetic tape cartridges and hard disks as target media. HSMS can, for example, dump full backups to magnetic tape cartridge, while the interim incremental backups are written to disk. In addition, backups can be stored intermediately on disk and subsequently, at a convenient time, be moved to magnetic tape cartridges.

By default, only read access is possible to any file currently being saved, i.e. no modifications can be applied during backup. If the function “Concurrent Copy” is activated for backup, files may be modified during backup. This function is particularly useful for applications which should be interrupted either not at all or as little as possible but for which files must nevertheless be backed up.

After a backup to magnetic tape cartridge has been created, this can be automatically duplicated to other magnetic tape cartridges. This increases the security of the data: if a volume is damaged or lost, the copy can be used to reconstruct the corrupted data or restore the lost files.

HSMS also enables backup/restore of pubset copies which were created using SHC-OSD mirroring functions. The Concurrent Copy function is used for this purpose.

Nonprivileged users can back up and restore files and job variables of another user ID of which they are co-owners. For more detailed information on co-ownership, refer to the “SECOS” manual [16]).

It is possible to save additional information on the element structure when backing up PLAM libraries (except for the migrated libraries). With the aid of the saved element structure, every element present in the library can be restored individually using the HSMS statement RESTORE-LIBRARY-ELEMENTS.

Version Backup

The version backup is a variant of the backup. HSMS ensures that a number of versions of a file are kept in the version backup archive. The user defines the number of file versions to be kept for a file with the file attribute NUM-OF-BACKUP-VERS (see BS2000 command MODIFY-FILE-ATTRIBUTES). Only differential backups are performed as part of the version backup.

Archival

Archival is the long-term saving of files that are no longer required online at the processing level. It is possible to define retention periods for data that has been archived independently of the retention periods of the volumes.

Archival is used to save storage space at the processing level by virtue of the fact that the files can be deleted after archiving. This also takes some of the load off the catalog. Archival is also used for documentation purposes, e.g. if legal regulations prescribe that data must be retained for specific periods of time.

As with a backup, following archival the data can be automatically duplicated to other magnetic tapes.

Nonprivileged users can also back up and restore files of other user IDs if they are the owners (see the “SECOS” manual [16]).

Migration

The migration function offered by HSMS makes it possible to optimize the utilization of highspeed disk storages. In particular in systems with frequently changing applications it is sound practice to move data that is currently not required (inactive data) from the processing level to a background level, keeping the files accessible via their catalog entries. This helps to avoid pubspace saturation or reaching the user ID-specific pubspace limit without noticeably reducing the availability of the files (can also be automated for pubsets).

Whenever the Data Management System (DMS) attempts to access a migrated file, an HSMS request is generated automatically; the file is then recalled to the processing level without user intervention.

Migration permits more efficient utilization of expensive high-speed external storages without increasing personnel requirements. Since only those pages are migrated which are actually occupied, storage space is saved at the background level in comparison with the processing level.

Files which are stored on the disks of a disk storage system and are mirrored remotely using SHC-OSD should not be migrated if the remote host is unable to access the
S2 processing level.

With the SM2 application the migration and recall activities can be traced and the migration parameters can be set to their optimum values.

Nonprivileged users can also migrate and recall files of which they are co-owners (see the “SECOS” manual [16]).

> The basic migration function is not available for files on Net-Storage and private disk, and for files of BS2000-UFS (POSIX) and remote nodes S0 mounted in these.

Data transfer (export/import)

HSMS can be used to transfer files and job variables to other BS2000 systems or other user IDs and to exchange volumes. To do this, the data is written to tape cartridges (exported) and then loaded (imported) to the desired system or user ID. If necessary, you can also work with archive directories.
Moreover, it is possible to transfer only the file catalog entries.

Overview of the HSMS basic functions

BACKUP	ARCHIVAL	MIGRATION	EXPORT
System service and user control (protection against destruction)	User procedure (company and legal regulations)	System and user control	User procedure
All files and job variables	Selected files and job variables	Inactive files	Selected files and job variables
Short-term storage of a copy (e.g. 6 weeks) Version Backup: permanent retention of multiple versions of a file	Long-term storage of a copy (and deletion of the original)	Medium-term swapping out of files	Swapping out for transfer
RESTORE	RESTORE	RECALL	IMPORT
In case of loss of the original or for reorganization	In exceptional cases only	For processing	For processing
Controlled by HSMS statements	Controlled by HSMS statements	Controlled by HSMS statements and automatically when access is attempted	Controlled by HSMS statements

2.2 Storage hierarchy

HSMS implements migration in a three-level storage hierarchy which, among other features, offers support of both multiple public volume sets (MPVS, SF pubsets), shared SF pubsets, SM pubsets, and shared SM pubsets.

HSMS distinguishes between the three storage levels S0, S1, and S2, which are also addressed as such in the HSMS statements and BS2000 commands.

SF pubsets can be assigned either to processing level S0 or to storage or background level S1 depending on their properties such as availability, access time, and costs. The storage and background level S2 is implemented using tape media.

As far as HSMS is concerned, SM pubsets already have an internal storage hierarchy. But they can also be assigned as S1 level in the SF environment (S1-SM pubsets).

Net-Storage and private disks can be backed up and used for data saving under HSMS, but are not assigned to any of the storage levels.

Storage level S0

The S0 storage level is the normal processing level. It is implemented by pubsets which users can access online. S0 is managed by the Data Management System (DMS). It is at this level that the data which can be handled using the basic HSMS functions is generated and processed.
If the system comprises disk storages with different characteristics, the high-speed disk storage types, i.e. the ones with short access times, should be assigned to storage level S0.

Storage level S1

The online background level S1 is also implemented by disk storages organized in SF pubsets, S1-SM pubsets, or volume sets of an SM pubset. The data on storage level S1 is managed by HSMS (HSMS files, of course, also being managed by DMS). It is to the S1 storage level that data from S0 is moved for migration.

High-capacity disk storages which can have longer access times than those on storage level S0, on which the data is stored at lower cost (to which compressed storage also contributes), are primarily assigned to storage level S1.

In the SF environment, storage level S1 can be assigned either globally for the entire system or specifically to individual S0 pubsets (see figure 1). It consists of one or more SF pubsets or S1-SM pubsets.

By contrast, in an SM pubset a volume set can be defined as S1 level. In HSMS V11.0 and higher, the S1 level can also be extended to all volume sets of the SM pubset which are under HSMS management. This extension is only possible in BS2000 OSD/BC V11.0 and higher. Irrespective of the number of volume sets available, the S1 level is only available in this SM pubset on a local pubset basis.

Storage level S1 is not mandatory in a system under HSMS management because the background level S2 is sufficient for standard HSMS operation.

Storage level S1 is not available for processing node files of the BS2000-UFS (POSIX) and remote nodes S0 mounted there.

Storage level S2

Storage level S2 consists of magnetic tape cartridges. The storage costs are considerably lower than for S0 and S1. On the other hand, operator or at least robot intervention is generally required to make level S2 data accessible. Access times for S2 storages are therefore of a different magnitude than those for the other storage levels.

S2 data can be used for migration. Optimum utilization of S2 volumes is achieved by “continuing” the magnetic tape cartridges.

The data on storage level S2 is managed by HSMS. HSMS allows the periods of time during which the volumes of storage level S2 may be processed to be defined on a function or archive-specific basis.

Overview of HSMS storage levels

storage level	Volume	Access	Access time
S0 processing level	Disks	Online	Very short
S1 background level	Disks	Online	Short
S2 background level	Magnetic tape cartridges	Offline	Long

The pubsets in a BS2000 configuration can be assigned to the various storage levels. For example, a pubset environment under HSMS management could look like this:

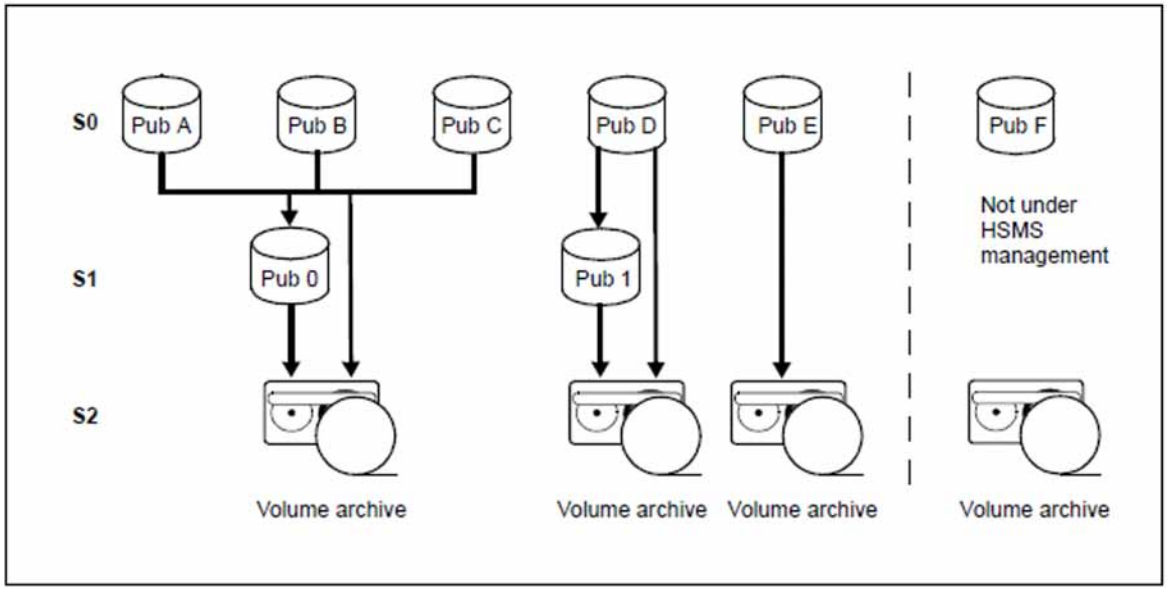


Figure 1: Example of a multiple public volume set under HSMS

In this example, the global S1 pubset 0 is assigned to pubsets A, B and C.
In contrast, the S1 pubset assigned to pubset D is pubset 1. Pubset E does not have an S1 pubset, which means that backups can be made only to S2.

One or more volume sets can be assigned to each SM pubset in a BS2000 configuration as an S1 storage level. If the SM pubset outside the S1 volume set is damaged and must be set up again, existing full volume sets can be integrated into the SM pubset reconfiguration.

In addition, an SM pubset can be set up as storage level S1 within the SF environment. To do this, it is placed under HSMS control as an SM pubset and then defined either globally like an SF pubset or as storage level S1 on a pubset-specific basis. For details, see section "Assigning an SM pubset to storage level S1".

2.3 Archives

Archives are the basic management units for the data managed by HSMS. HSMS stores all data saved by either backup, archival or migration in archives, keeping separate archives for each of the basic functions:

- backup archives for backup
- version backup archives
- long-term archives for archival
- migration archives for migration.
- shadow archives for the copies of save files. The copies are created automatically for backups and archivals.

HSMS makes a distinction between *public* archives, which are available to all users, and *private* archives, which may be accessed by the archive owner only.

The HSMS administrator can assign public archives as default system archives:

- system-wide
- for specific SF pubsets
- for SM pubsets.

All users can use these system archives for any of the HSMS basic functions. They are addressable under the following symbolic names:

- SYSBACKUP for the default system backup archive
- SYSVERSION for the default system version archive (only pubset specific)
- SYSARCHIVE for the default long-term system archive
- SYSMIGRATE for the default system migration archive.

Data managed in the archive is stored in save files which can be located on S1 pubsets, private disks, or magnetic tape cartridges. In order to save memory space, the data can be compressed before it is written to the save file.

To increase data security, HSMS offers a function which can be used to copy save files – or excerpts of save files – within the same archive or to another archive.

Summary

To sum up, the data managed by HSMS and the storage media can be classified according to two criteria.

- Using volume *availability* as a criterion, the storage media are assigned to the different levels of the storage hierarchy made up of S0, S1 and S2.
- Using *utilization* as a criterion, the data inventory is subdivided into management units formed by the default system archives and private archives.

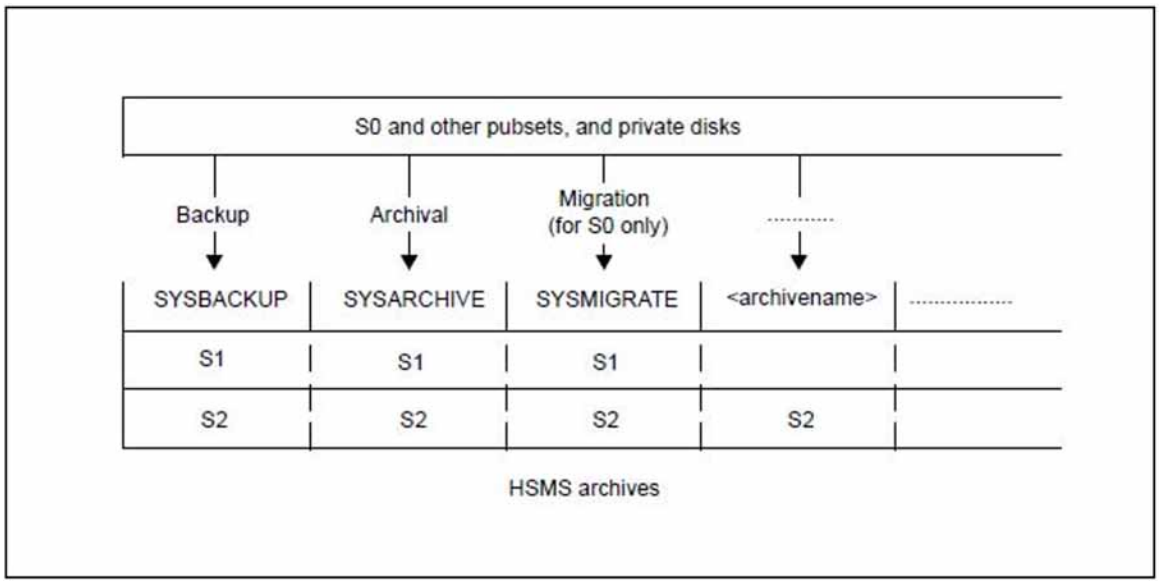


Figure 2: HSMS storage management structure

HSMS offers a set of inquiry functions for the management of data, storage units, archives etc. For instance, HSMS can be used to ascertain the occupancy of pubsets in order to forestall possible saturation.

2.4 Co-ownership

Owners of objects can use co-owner protection to specify for which of their objects they want to designate co-owners and the access conditions that co-owners must fulfil in order to perform administrative accesses. The object owner is the user ID that sets up the object. In HSMS, these objects can be files and job variables and archives. The co-ownership of an archive is implemented in HSMS by creating a co-owner for the associated archive directory. Furthermore, HSMS supports co-ownership for input and output files and the job variables to be monitored.

The co-owner of an archive can then treat this archive as if it were their own. The only thing they cannot do is modify the archive attributes.

A co-owner is a user ID which is different from that of the owner. However, for the object in question this user ID possesses the same rights as the owner.

In general, the following applies to co-owners: all read, write and execute accesses to files are controlled by means of the rules of the traditional file security mechanisms:

- If a file is protected by USER-ACCESS, ACCESS or BASIC-ACL, a co-owner has the same read, write and execute rights as the file owner.
- If a file is protected by guards, access is controlled via the evaluation of access conditions that are defined in STDAC guards.

Co-owners can be defined, modified or viewed using the BS2000 commands ADD-/MODIFY-COOWNER-PROTECTION-RULE and ADD-/MODIFY-/REMOVE-/SHOW-ACCESS-CONDITIONS.

For more detailed information on co-ownership, refer to the “SECOS” manual [16].

2.5 User classes

HSMS distinguishes between three classes of users for the utilization and control of its functions: nonprivileged users, HSMS administrators and subsystem administrators. In this manual, the user classes to which the individual functions are available are indicated where appropriate.

Nonprivileged users (“STANDARD PROCESSING” privilege)

Nonprivileged users may *back up* files and job variables of which they are the owner or co-owner in the following archives:

- a public system archive (to which ACCESS=*WRITE applies)
- their own archive
- a public archive belonging to another user if the data which is to be backed up belongs to this another user.
- another user’s archive, for which the user is a co-owner of the corresponding archive directory. This archive does not have to be public.

Nonprivileged users may *archive* files and job variables of which they are the owner or co-owner. They may archive these files in a system archive or in a user archive, provided they are authorized to access this archive.

Nonprivileged users may also *migrate* files of which they are the owner or co-owner to a public system archive. They are also permitted to *transfer* files and job variables of which they are the owner or co-owner.

Finally, nonprivileged users may also *set up their own archives*, which can be private or public.

HSMS administrator (“HSMS-ADMINISTRATION” privilege)

The HSMS administrator is authorized to make unrestricted use of all functions and features offered by HSMS. The HSMS administrator works under a user ID to which the “HSMS administration” privilege has been assigned. By default, these are the user IDs SYSHSMS and TSOS.

The HSMS administrator is responsible for system backups, i.e. system-wide backups of the data of all users. Once HSMS has been invoked, the HSMS administrator has access to the files of all users on all pubsets. He or she is also responsible for managing the storage hierarchy and creating the default system archives for the basic HSMS functions as well as for controlling tape access operations for HSMS.

In addition, the HSMS administrator is responsible for setting up shadow archives. Shadow archives are used to administer the copies of save files which are automatically created on magnetic tape cartridges for backups and archivals.

Subsystem administrator (“SUBSYSTEM-MANAGEMENT” privilege)

The subsystem administrator is authorized to start and terminate HSMS.

2.6 HSMS in an SM pubset environment

In BS2000, two pubset types can be found: single feature pubsets (SF pubsets) and system managed pubsets (SM pubsets). Both are called via their catalogue ID.

An SF pubset consists of one or more disks whose basic attributes (e.g. disk format, allocation unit, availability) must agree. An SM pubset, on the other hand, can be made up of so-called volume sets with different attributes. The essential attributes only have to agree within each volume set.

When a user defines volume-set-specific attributes for a file on an SM pubset, the system determines a volume set of the SM pubset which matches these attributes and stores the file there. Above all this enables a file to be moved to a volume offering different performance within the same SM pubset without having to change the file name.

For more detailed information on SM pubsets, see the “SMS” manual [18].

An SM pubset under HSMS control is an SM pubset that is managed by HSMS. The HSMS metadata is located on the SM pubset itself. An SM pubset not under HSMS control can only be processed by HSMS in a few cases.

The description below is based on the assumption that the SM pubset concerned is under HSMS control. For simplicity’s sake, it is therefore just called an SM pubset. Wherever there is room for doubt, we expressly indicate whether the SM pubset is under HSMS control or not.

An SM pubset is a closed container that holds data and metadata. The files are located either on the S0 processing level or on the S1 or S2 background levels, depending on whether they contain active or inactive data.

An SM pubset has the following attributes:

- Each SM pubset under HSMS control contains just one SM pubset object of the DMS data management system, which is represented by processing levels S0 and S1. An SM pubset consists of several volume sets which together form a pool. The mandatory S0 processing level is made up of one or more volume sets which are not reserved by HSMS; the files for processing must be located on this level.
- Storage and background levels are required to save and migrate files of processing level S0 within the SM pubset:
  - S1 contains files stored and migrated to disk; the S1 level is either formed by a volume set within the SM pubset or by all volume sets under HSMS management.
  - S2 contains files stored and migrated to tape; the S2 level consists of a tape pool which is implemented by a backup archive directory or a MAREN pool.
- To enable the files to be administered on the different levels, the SM pubset environment is described by metadata. The metadata is located in the pool on a specially identified volume set, known as the control volume set. The metadata consists, for instance, of data for the description of the volume set configuration, user catalog, guards catalog, pointers to file catalogs, catalogs for special objects, such as job variables, migrated files, files on private volumes etc. The metadata is stored either in HSMS files on the SM pubset or in HSMS files on the default pubset of SYSHSMS, depending on whether they are system or pubset-specific.

To be able to process an SM pubset as a “closed container”, the term “environment” is used in HSMS. An environment contains data and metadata. An SM pubset under HSMS control is by definition an environment. A different SM pubset is also a different environment.

The data and metadata stored in an SM pubset only relate to objects of that SM pubset. If, during backup/restore or migrate/recall, reference is made to an archive that is defined in an SM pubset, only files and job variables that belong to that SM pubset can be processed. Nevertheless, files backed up in one environment can still be restored to another environment.

The exceptions to this rule are exporting and importing. Exporting and importing are functions that move files from one configuration to another. In other words, these are global functions, i.e. they cannot be allocated to a particular SM pubset environment. These functions are always processed in an SF environment with no restriction on the data to be processed. The metadata (jobs, control file, ...) is physically stored in the SF environment, while the associated data may be stored in any environment.

Archiving and restoring are long-term functions in which the configuration of the environment can temporarily change. They can either be allocated to a computer center configuration, or not. Therefore, these activities can be global or local, i.e. they can be allocated to an SF environment or an SM environment. The metadata (archive, jobs, control file, ...) is physically stored in the related environment, while the associated data may be stored in any environment.

All activities on an SM pubset can even be carried out on another computer or the same computer with another home pubset after the SM pubset has been imported. Interrupted jobs can also be restarted.

The total set of SF pubsets is regarded as a single environment. The data and metadata on SF pubsets can be managed by HSMS.

2.7 BS2000 backup server in a shared pubset environment

A backup server can be defined explicitly in the SPVS network to process HSMS requests. Processing takes place on the backup server regardless of whether the server is the master or slave host of the shared pubset. The aim of this is to relieve the load on productive systems.

Details of configuration and the method of operation are provided in section "Backup server".

2.8 Backing up remote computers using HSMS

The rapid transition from centralized data processing to company-wide distributed information technology has resulted in a substantial increase in the number of remote computers – in particular workstations and PCs – involved. These remote computers are now extremely powerful. Large volumes of data can be stored on their hard disks.

HSMS enables files from remote networked UNIX computers to be backed up, archived and, if necessary, restored without the need for a buffer in BS2000 Migration, import and export are not supported for remote computers.

A file system of a UNIX workstation or parts of it can be mounted in the BS2000-UFS (POSIX) using NFS if an NFS server is installed on the UNUX client and the corresponding directory is exported. In this way HSMS accesses the files of the remote UNIX system which are to be saved or restored. From the HSMS viewpoint the remote UNIX is a passive node, i.e. no separate backup software is required on UNIX to perform the backup.

2.8.1 Backing up UNIX workstations

Files on this type of workstation are organized in UNIX file systems (UFS). A UNIX file system is a hierarchical tree structure consisting of files and directories. The origin of this hierarchy is the *root* directory whose path name is the slash (/). From here the directory structure proceeds downwards. Each directory subordinated to the root directory can have subdirectories or files of its own. Files constitute the lowest level of this hierarchy. From a file no further branch is possible.

Directories are also referred to as the nodes of a file system; they contain the names of files and subdirectories subordinated to them. The names of the files and directories are user-defined but have to comply with certain conventions.

For further information about the UNIX file system please refer to the “NFS” manual [11].

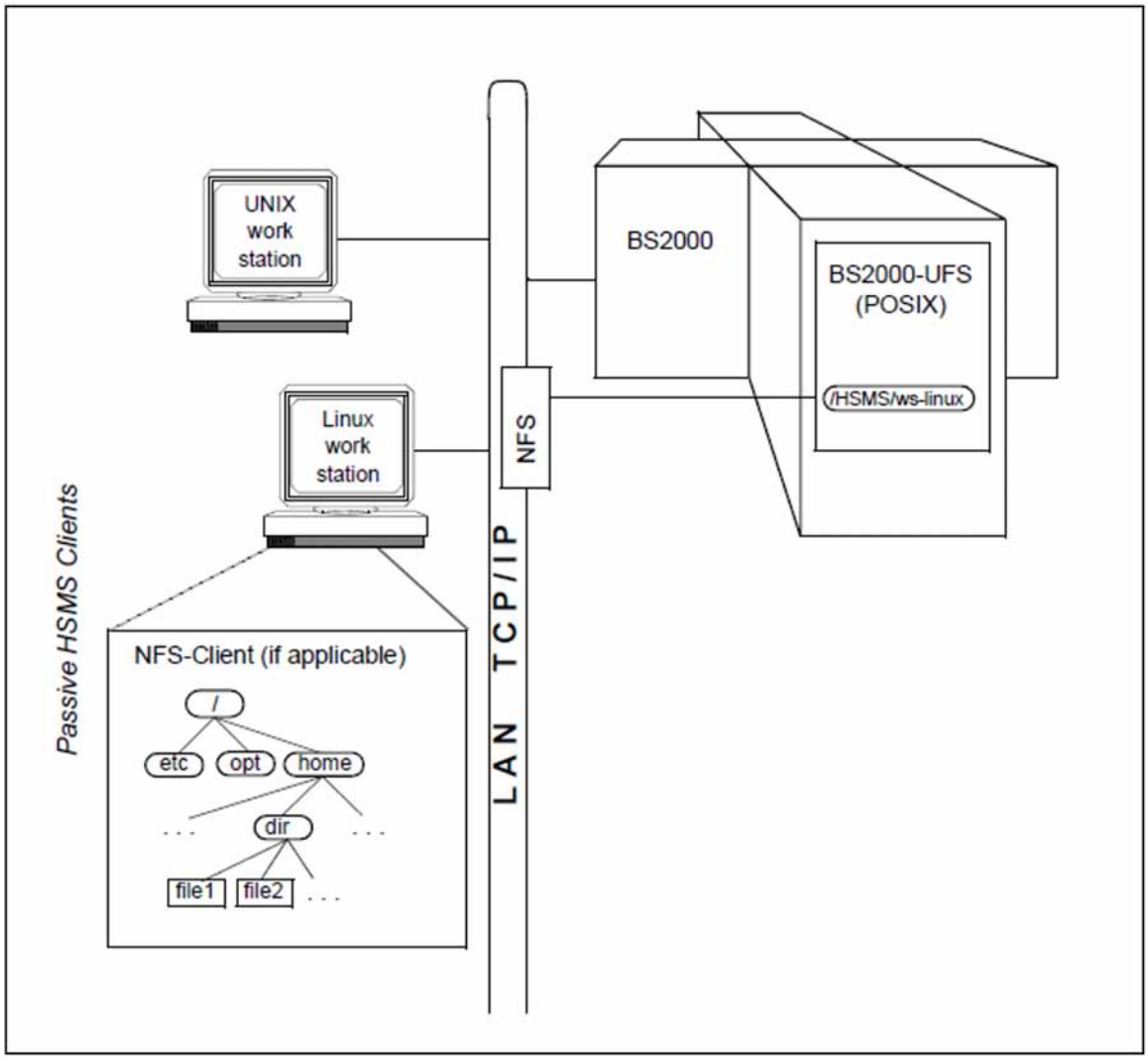


Figure 3: Backing up remote computers using POSIX mount

2.8.2 Storage hierarchy "node S0"

Since HSMS does not support the migration of node files via POSIX, there is only one storage level. Within this manual, this level is referred to as “node S0”.

The node S0 storage level comprises a number of node S0s. Each node S0 is named for the remote computer on which a file system is normally processed. Only the node S0 which corresponds to the local BS2000-UFS has no name.

Each node S0 contains one or more file systems, depending on the mount processes activated by the BS2000 system administrator.

Access to the node S0 and its files, referred to as node files, is made possible by a definition in the tree structure of the local BS2000-UFS:

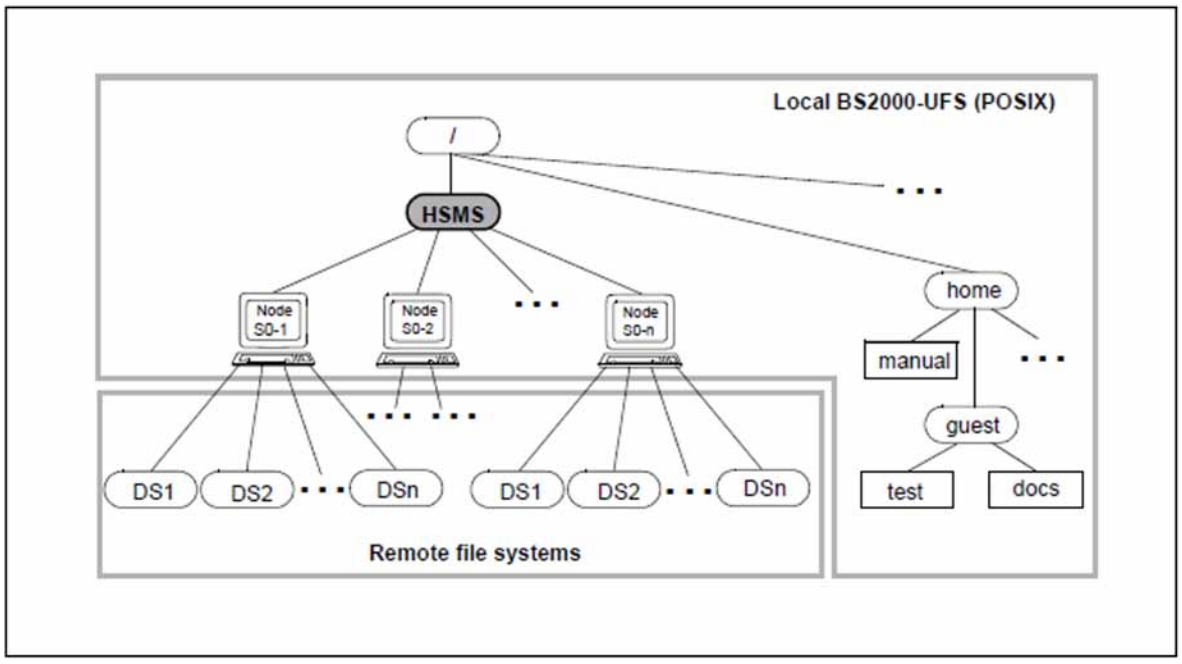


Figure 4: Passive node S0

Note

Within the context of HSMS, a “remote file system” is a file system with the following attributes:

- It has been mounted under the “HSMS” directory.
- It is subordinate to a specific node S0.
- It can be reached from the BS2000 root directory.
- It permits access by BS2000.
- It is mounted in the local BS2000-UFS.

2.8.3 Archives

The definition of archives has to distinguish between the following two cases:

- **Processing of files of the local BS2000-UFS**

  Users can define archives for processing their own files. The HSMS administrator has to define public archives which can be accessed via the symbolic names SYSNODEBACKUP and SYSNODEARCHIVE.

- **Processing of files on a remote (passive) node S0**

  The HSMS administrator must define archives either at system level (a single archive for all node S0s) or at the node S0 level (one archive per node S0). The HSMS administrator should choose one of the two options depending on how he or she wishes to organize backup and archival.

  For each archive that is used for the backup and archival of node files, the HSMS administrator can create a shadow archive.

2.8.4 User classes

HSMS distinguishes between three classes of users for the utilization and control of its functions. In this manual, the user classes to which the individual functions are available are indicated where appropriate.

HSMS administrator (“HSMS-ADMINISTRATION” privilege)

The HSMS administrator is authorized to process all files residing on the local BS2000-UFS or any remote file system. The HSMS administrator has to create public archives for the local BS2000-UFS as well as his or her own working archives for the remote file systems.

BS2000 system administrator (“TSOS” and “root” privileges)

The BS2000 system administrator has to mount each remote file system to be processed on the “HSMS” node of the local BS2000-UFS. For this action, the BS2000 system administrator must have the root privilege for the computer on which the remote file system is normally processed. The root privilege is granted in BS2000 by assigning user number 0; this user number is by default allocated to the user ID SYSROOT.

The BS2000 system administrator has to create the “HSMS” directory in the local BS2000-UFS at initialization.

Before mounting the remote file system of a new computer on the local BS2000-UFS, the BS2000 system administrator has to create the node S0 for the new computer in the “HSMS” directory (*mkdir* command).

2.8.5 Support for node path names

HSMS only supports POSIX path names for node files. The syntax of the POSIX path names are checked by HSMS. This also applies to path names taken from a list (operand PATH-NAMES=*SELECTED) or a file (operand PATH-NAMES=*FROM-FILE) and for requests from remote workstations.

You must pay careful attention to the use of uppercase and lowercase letters when entering POSIX file names and path names.

2.9 Entering statements via SDF

HSMS is operated via a statement interface. All inputs to HSMS are handled by the HSMS program.

Statements to HSMS are processed by the SDF command processor (System Dialog Facility). HSMS thus offers various forms of guided or unguided dialog, including the options of requesting statement-specific help menus and of conducting a correction dialog in the event of invalid statements. For further details see the “SDF Dialog Interface” manual [5].

3 HSMS archives

The archive is the basic HSMS management unit. HSMS stores and manages all saved, archived and migrated data (files and job variables) in archives.

Each archive consists of

- the archive definition, which is stored in the central control file and defines the archive's attributes
- the archive directory used for managing the files, job variables and volumes saved in this archive (implemented as an ARCHIVE directory file)
- the volumes and save files containing the saved data.

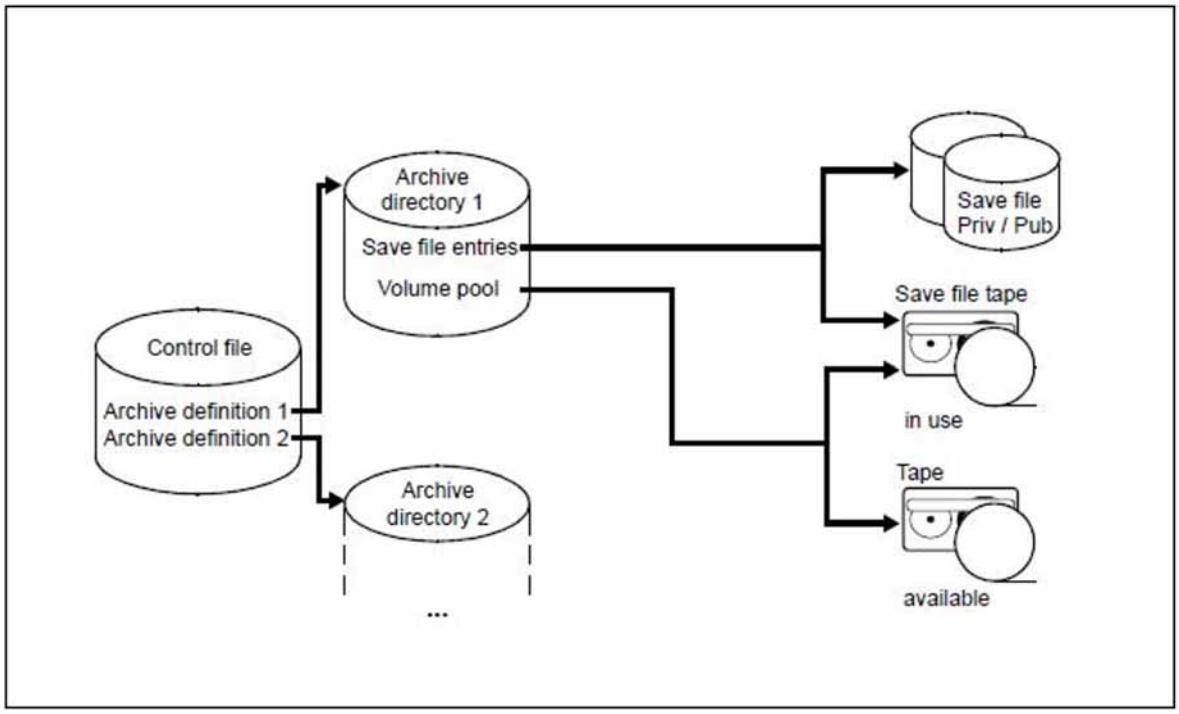


Figure 5: Structure of an HSMS archive

When processing *BS2000* files, HSMS distinguishes a different type of archive for each of the three basic HSMS functions - backup, archival and migration - plus the archive type "shadow archive". Shadow archives are used to administer the copies of save files which are automatically created on magnetic tape cartridges for backups and archivals.

When processing *node* files, HSMS distinguishes two further archive types, one for backup and one for archival.

None of these basic HSMS functions can be executed until an appropriate archive has been created. All HSMS statements to any of these basic functions refer to existing archives.

Nonprivileged users can only manage their own archives. They can access public archives according to the set access permissions (USER-ACCESS). For this, backup archives have to be set up under SYSHSMS.

When CREATE-ARCHIVE is carried out by a nonprivileged user, the user ID of this user becomes the owner of the archive.
When CREATE-ARCHIVE is carried out by an HSMS administrator, the user ID SYSHSMS is set up as the owner of the archive, unless a different user ID is specified in the archive name, and if this is the case then this user ID will become the owner.
The owner is responsible for the administration of the archive.

Shadow archives, migration and version backup archives must be installed by a HSMS administrator.

Nonprivileged users can obtain information on data managed in a public archive and belonging to their respective user IDs. If the archive owner allows other users to access his archive/directory (via SECOS), then each co-owner can use the archive as though it were his own. However, co-owners cannot manage the archive, i.e. they cannot share save files or change retention periods.

Data transfer

HSMS does not manage exported data in archives; after all, exported data is to be used at another location. However, it is possible to manage information about exported data in a directory whose structure is identical to that of an archive directory.

The exported data is written to save files which are treated like the save files of the other HSMS basic functions. In this respect, the sections on the save files and versions in this chapter also apply to data transfer.

3.1 Archive definition

Each archive is created by means of the HSMS statement CREATE-ARCHIVE. The archive definition, including all its attributes, is stored in the HSMS control file, which also includes the name of the archive directory.

The HSMS control file depends on the environment in which the archive is defined:

- For an SF pubset environment, it is the central HSMS control file.
- For an SM pubset environment, it is the control file of the SM pubset.

During creation the archive must be given a name. The archive name has the format of a file name but must not have more than 12 characters. In addition to the name, the archive is assigned an owner ID.

The HSMS administrator can specify an owner ID explicitly. If no owner ID is specified, the default owner ID is SYSHSMS, regardless of the user ID involved.

An archive can be addressed in HSMS statements by specifying its owner ID and archive name in the following form:

[$owner-id.]archivename

Archives in which shared SF pubsets are managed, should be set up in such a way that a separate archive is created for each shared pubset. The archive must have a directory file on the respective shared pubset and the archive definition must be identical on all servers. A request issued for a shared pubset imported in slave mode can only be processed if only pubsets assigned to the same master computer will be affected by this request.

In the case of shared SM pubsets, the directory files for backup, migration and archival are automatically stored in an SM environment on the SM pubset itself. Therefore, with SM pubsets HSMS automatically guarantees the coherence of the archive definitions for backup and migration.

The HSMS statement CREATE-ARCHIVE also serves to define the archive attributes. These can be modified after creation by means of the HSMS statement MODIFY-ARCHIVE-ATTRIBUTES.
The owner of an archive can request information about its attributes by means of the HSMS statement SHOW-ARCHIVE-ATTRIBUTES.

From HSMS V8.0 the archive attributes are also stored in the directory file in the case of CREATE-ARCHIVE or MODIFY-ARCHIVE-ATTRIBUTES. If the archive definition is lost the attributes can be restored from the directory file using CREATE-ARCHIVE (see “Restoring an inadvertently deleted archive definition” in chapter "Deleting archive definitions").

MODIFY-ARCHIVE-ATTRIBUTES can be used to change the catalog ID of the archive directory accordingly when the pubset has been renamed.

3.1.1 Archive utilization (archive type)

When an archive is created, its type must be defined with the ALLOWED-USAGE operand of the HSMS statement CREATE-ARCHIVE, i.e. it must be assigned one of the basic HSMS functions. HSMS distinguishes between

- backup archives for backup (ALLOWED-USAGE=*BACKUP)
- version backup archives for version related backup (ALLOWED-USAGE=*VERSIONBACKUP)
- long-term archives for archival (ALLOWED-USAGE=*ARCHIVAL)
- migration archives for migration (ALLOWED-USAGE=*MIGRATION)
- backup archives for the backup of node files of the POSIX file system or remote nodes S0 (ALLOWED-USAGE=*NODEBACKUP)
- long-term archives for the archival of node files of the POSIX file system or remote nodes S0 (ALLOWED-USAGE=*NODEARCHIVAL)
- shadow archives for the copies of save files. Shadow archive can only be assigned to backup or long-term archives.
  (ALLOWED-USAGE=*SHADOW).

Note that only the HSMS administrator may create migration archives and shadow archives. All other types of archive may be created by nonprivileged users also.

Each archive can only be used for the function for which it has been created. For instance, it is not possible to migrate files to a backup archive, but only to a migration archive.

Shadow archives can only ever be assigned to a single backup or long-term archive with SEVERAL SVID structure. As this assignment must already be made when the shadow archive is created, the long-term or backup archive to be assigned must already exist.

A shadow archive can be used for the following purposes:

- automatic or explicit copying of save files from a long-term or backup archive into the associated shadow archive
- explicit copying of save files from a shadow archive into the associated long-term or backup archive
- restoring data

The connection between an archive and the associated shadow archive is canceled by deleting the shadow archive. In addition to this total deletion of the shadow archive, the following functions are also offered:

- **Separating the shadow archive:** The connection between an archive and the associated shadow archive is canceled and the shadow archive is turned into a backup or long-term archive. The attributes of the archive originally assigned are taken over for the new archive.
- **Replacing the archive directory:** The definition of the shadow archive is deleted and the archive directory of the shadow archive is taken over for the assigned backup or long-term archive. The original archive directory is retained without any link to an archive.

3.1.2 Access rights

The owner of an archive has various possibilities of controlling access to the archive. By default an archive is not created as shareable, in other words only the owner of an archive may use it to back up data; it is a *private* archive.

All users have the right to create an archive that may be used by other users besides the owner for the basic function defined. This is considered to be a *public* archive:

```
//CREATE-ARCHIVE USER-ACCESS=*ALL-USERS
```

Public backup archives belonging to nonprivileged users can, however, only be used by other users within the context of co-ownership. A public backup archive is available to all users only if the archive's owner ID is SYSHSMS (see section "Default system archives"). However, long-term archives are not affected by this restriction.

Public archives from which users are to be able to restore the data belonging to their user ID must be set up with ACCESS=*READ. If users are to be able to back up their files in a public archive, the archive must be set up with ACCESS=*WRITE.

Co-ownership of an archive is defined via the co-ownership of the corresponding directory file. All co-owners can use this archive as their own, however, they cannot change the archive properties, share save files or change their RETPD. Co-ownership access is irrespective of the archive's specified access attributes.

Whether or not an archive is shareable within HSMS is independent of the file attributes of its directory and is determined exclusively by the entry in the HSMS control file – with the exception of co-ownership (see above). It is determined exclusively by the entry in the HSMS control file – with the exception of co-ownership (see above). Its ability to be shared cannot be changed by means of BS2000 commands.

The access authorization for a shadow archive includes only the explicit copying of save files from the long-term or backup archive into the associated shadow archive and vice versa, and the restoration of data from the shadow archive. The authorization for automatic duplication to a shadow archive depends entirely on the access authorization of the associated long-term or backup archive.

If a nonprivileged user performs an archival to a public archive to which a shadow archive is assigned, the data is automatically duplicated in the shadow archive, even if the shadow archive is private (ACCESS=*OWNER-ONLY).

The following overview clarifies the access to foreign archives, which comprise co-ownership (in conjunction with SECOS).
To enable access, it is required that each target archive is installed correspondingly (with USER-ACCESS=*ALL-USERS and ACCESS=*WRITE or the archive/directory's co-ownership). In order to backup own files into other archives, you have to be co-owner of the respective archive directory, unless the backup archive is created under the SYSHSMS ID.

Backup by user A:	**in archive of user ID A**	**in archive of user ID B**	**in SYSBACKUP**
Own file under user ID A	yes	yes	no
File under user ID B for which user A is a co-owner	yes	yes	no
Shared file under user ID B (no co-ownership)	no	no	no

yes: access permitted

no: access not permitted

Long-term archival by user A:	**in archive of user ID A**	**in archive of user ID B**	**in SYSARCHIVE**
Own file under user ID A	yes	yes	yes
File under user ID B for which user A is a co-owner	yes	yes	yes
Shared file under user ID B (no co-ownership)	no	no	no

yes: access permitted

no: access not permitted

Migration by user A:	**in archive of user ID A**	**in archive of user ID B**	**in SYSMIGRATE**
Own file under user ID A	no	no	yes
File under user ID B for which user A is a co-owner	no	no	yes
Shared file under user ID B (no co-ownership)	no	no	no

yes: access permitted

no: access not permitted

Version backup by user A:	**in archive of user ID A**	**in archive of user ID B**	**in SYSVERSION**
Own file under user ID A	no	no	yes
File under user ID B for which user A is a co-owner	no	no	yes
Shared file under user ID B (no co-ownership)	no	no	no

yes: access permitted

no: access not permitted

3.1.3 Setting monitoring for an archive

The MONITORING attribute controls monitoring of requests on the BS2000 backup monitor on the SE server on an archive-specific basis:

- MONITORING=*NO specifies that requests for this archive are not displayed on the BS2000 backup monitor. This is the default setting when an archive is created.
- MONITORING=*STD specifies, that the current settings of the global HSMS MONITORING parameter are evaluated for the monitoring.

For the effect of the global settings in conjunction with the archive-specific setting, see section “Central request monitoring on the SE server”).

3.1.4 Setting the backup server for an archive

The BACKUP-SERVER-USAGE attribute controls the system in which the backup requests of the local system may be executed on an archive-specific basis:

- BACKUP-SERVER-USAGE=*NO specifies that backup requests of the local system for this archive are executed on the system which is currently the master sharer of the shared pubset. This is the default setting when an archive is created.
- BACKUP-SERVER-USAGE=*STD specifies that the current setting in the HSMS parameter BACKUP-SERVER is to be evaluated when executing backup requests of the local system for this archive.

For further details on setting the backup server, see section “Working with shared pubsets”).

3.2 Archive directory

The archive directory contains the information about the save data managed in the archive, i.e.

- BS2000 files, job variables or node files
- save files
- save versions
- occupied and free save volumes.

The name of the archive directory is defined with DIRECTORY-NAME when creating the archive by means of the HSMS statement CREATE-ARCHIVE.

By default, HSMS generates a new archive directory when creating an archive; it can be given any name. As an alternative, an existing directory can be reassigned to a new archive, provided it is not (no longer) associated with any other archive. However, directories of a migration or long-term archive can no longer be assigned to a backup archive after the first write request has been issued.

The archive directory need not be assigned to the user ID of its owner. However, the right to create archive directories under a user ID other than one’s own is restricted to the HSMS administrator.

The user ID of the archive directory is the same user ID for which free tapes are reserved with MAREN.

Public archives can be protected against unauthorized access by assigning protection (password, BACL or GUARDS) to the archive directory. If one of these mechanisms is used, it is essential to have *write* access to the archive directory, even if USER-ACCESS=*ALL-USERS(ACCESS=*READ) has been defined for the archive.

Similarly, private archives can also be opened for a group of users, with authorized access through co-ownership (via SECOS). Then, it doesn’t matter which user access is specified for the archive.

An archive directory for BS2000 files and job variables cannot be used as archive directory for node files and vice versa.

Note for ARCHIVE users

> The archive directory is implemented by an ARCHIVE directory file. Existing ARCHIVE directory files can be placed under HSMS management (see section "Special aspects when switching from ARCHIVE to HSMS operation").

Notes on restricting the record length in directory files

Currently the data of a BS2000 file can be distributed to a maximum of approximately 300 volumes as the record length in directory files is limited. This value depends on the save options, e.g. with SAVE-PLAM-INFO = *YES it can decrease to around 294.

If more volumes are required when saving the file, not all the information can be written to the directory concerned. In this case the request is terminated as "COMPLETED WITH ERRORS" and message ARC0176 is issued.

3.3 Default system archives

HSMS provides default system archives for each of the basic functions. Default system archives are created as public archives by the HSMS administrator and earmarked for use by the basic functions backup, archival and migration. Default system archives can be used for these functions even if their names are unknown, since they can be addressed by the following symbolic names:

- SYSBACKUP for the system backup archive
- SYSVERSION for the system version backup archive
- SYSARCHIVE for the long-term system archive
- SYSMIGRATE for the system migration archive
- SYSNODEBACKUP for the system backup archive for node files of the POSIX file system or remote nodes S0
- SYSNODEARCHIVE for the system long-term archive for node files of the POSIX file system or remote nodes S0

In an **SF pubset** environment, default system archives are created as shareable archives and assigned to the entire system as follows:

```
//MODIFY-HSMS-PARAMETERS DEFAULT-HSMS-STORAGE=*PAR( -
//  SYSBACKUP=<archive name>, -
//  SYSARCHIVE=<archive name>, -
//  SYSMIGRATE=<archive name>, -
//  SYSNODEBACKUP=<archive name>, -
//  SYSNODEARCHIVE=<archive name>)
```

The assignment of the system backup archive is only valid in this case for all imported SF pubsets and node S0s. System archives for archival are also valid for all imported SF pubsets and node S0s but are not restricted to them (see relationship with SM pubsets).

The assignment of the system migration archive is valid only for those BS2000 S0 pubsets for which migration is permitted.

These system-wide assignments can be modified for specific pubsets/node S0s:

- Individual pubsets can be assigned different SYSBACKUP, SYSARCHIVE, SYSMIGRATE and SYSVERSION default system archives by means of the HSMS statement MODIFY-PUBSET-PARAMETERS.
- Individual node S0s can be assigned different SYSNODEBACKUP and SYSNODEARCHIVE default system archives by means of the HSMS statement MODIFY-NODE-PARAMETERS.

> **i** System version backup archives can only be assigned to individual pubsets by means of the HSMS statement MODIFY-PUBSET-PARAMETERS. Moreover, the same version backup archive can be assigned only to one pubset at the same time.

In an **SM pubset** environment, the default system archives are set up as shareable and assigned to the SM pubset with:

```
//MODIFY-SM-PUBSET-PARAMETERS -
//  SM-PUBSET-ID=<cat-id>, -
//  SYSBACKUP=<archive name>, -
//  SYSARCHIVE=<archive name>, -
//  SYSMIGRATE=<archive name>,-
//  SYSVERSION=<archive name>
```

The assignment of the system archives for backup, version backup, archival and migration is only valid within this SM pubset.

The archiving activity is not restricted to the SM pubset environment in which it is issued. The environment to which the archiving function relates can be the host environment. In this case the archive definition is stored in the central HSMS control file.

When a user addresses a default system archive by means of its symbolic name, this name is resolved. The name can thus be mapped onto the archive that the HSMS administrator has assigned to a predefined environment.

The environment from which the symbolic archive names are resolved is generally the catalog ID or node ID specified in the HSMS statement. All affected catalog or node IDs are searched for their assigned symbolic archive:

- With HSMS statements that work in an SF environment, these are catalog IDs which are specified in the S0-PUBSET-ID operand (with the HSMS statements REPAIR-CATALOG-BY-EXCHANGE and REPAIR-CATALOG-BY-RESTORE) or which are used in the FILE-NAMES operand of an HSMS statement.
- With HSMS statements that work in an SM environment, this is the catalog ID specified in the ENVIRONMENT operand.
- With HSMS statements that work on workstations, the node IDs are taken from the NODE-ID operand.

An exception is the HSMS statement SHOW-ARCHIVE in an SF environment if the value *SAVE-VERSIONS, *SAVE-FILES or *VOLUMES is used with the SELECT operand. With these operand values no file list, catalog ID or node ID can be specified:

- For a nonprivileged user, either the default catalog ID of the user (the default catalog ID is defined in the user catalog) or the BS2000-UFS is used in order to resolve the symbolic archive name.
- For a privileged user, HSMS searches all catalog and node IDs defined in HSMS (with the HSMS statements MODIFY-PUBSET-PARAMETERS or MODIFY-NODE-PARAMETERS) in order to resolve the symbolic archive name.

The mapping of the symbolic names must be unequivocal. Since it is based on specific pubsets/node S0s, the following restriction applies:

If the operand FILE-NAMES or JV-NAMES is to refer to files or job variables of more than one pubset/node S0, the same default system archive must have been assigned to all of the pubsets/node S0s involved so that the corresponding symbolic archive name, when specified, always refers to the same archive. If this is not the case, each save request should refer to the files of one single pubset/node S0 only.

Regarding system version backup archives, it is not allowed to assign one and the same archive to several pubsets. Therefore, operand FILE-NAMES must refer to files from one pubset only when dealing with system version backup archives.

Nevertheless, it is useful to create various archives for the pubsets/node S0s for performance reasons, as some archiving activities can only be performed serially. Creating several system archives for each basic function reduces the probability of bottlenecks occurring when archives are accessed.

3.4 Relationship between SF pubsets and archives

In a computer center, pubsets and archives are used flexibly:

- Pubsets are detached from one system and connected to another.
- User IDs are moved from one pubset to another.
- Migrated files are deleted or are brought back to the processing level S0. This means that the storage levels S1 and S2 require regular reorganization.
- The extent of the damage resulting from a system crash or operator error should be kept to a minimum (e.g. restricted to a single pubset).
- If a system crash or operator error leads to a loss of data, the data must be restored in such a way that work on undamaged pubsets can proceed unimpeded.
- Administration work carried out on a pubset (backup, migration, reorganization, ...) should not interfere with users working on other pubsets.

3.4.1 Possible forms of organization

Central and decentralized administration and the respective advantages and disadvantages are presented on the following pages. The type of administration chosen has an impact especially on backup and migration.

3.4.1.1 Central administration

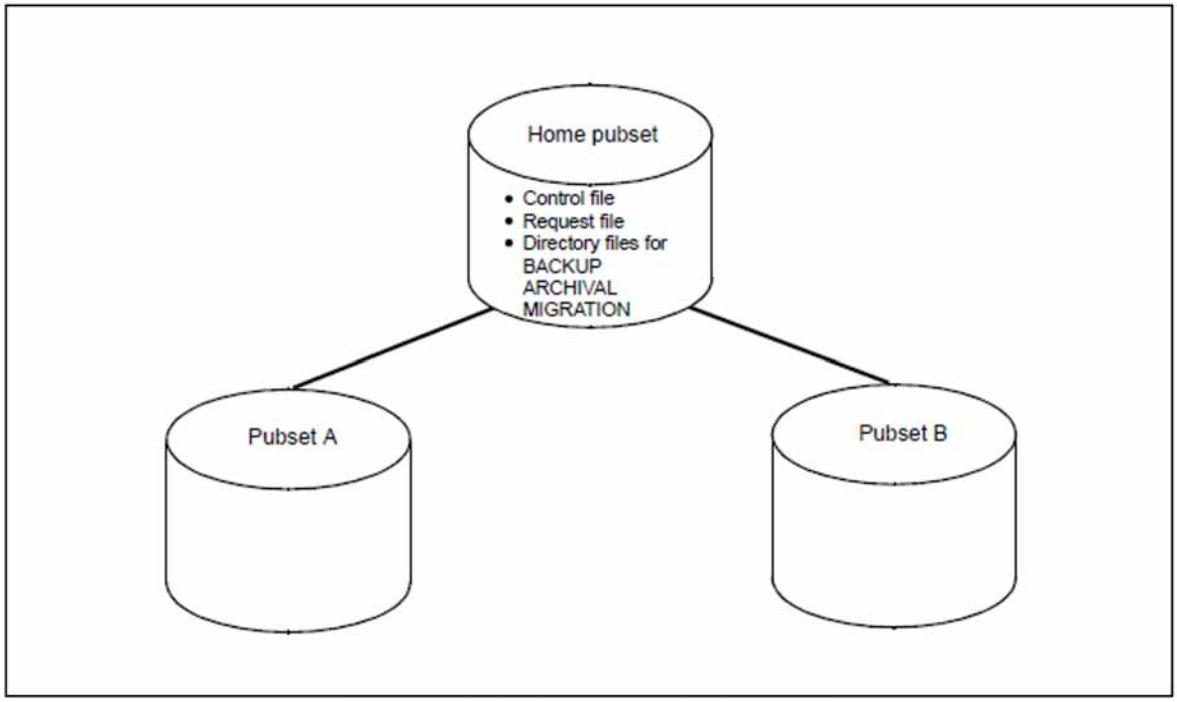


Figure 6: Central administration

3.4.1.2 Decentralized administration

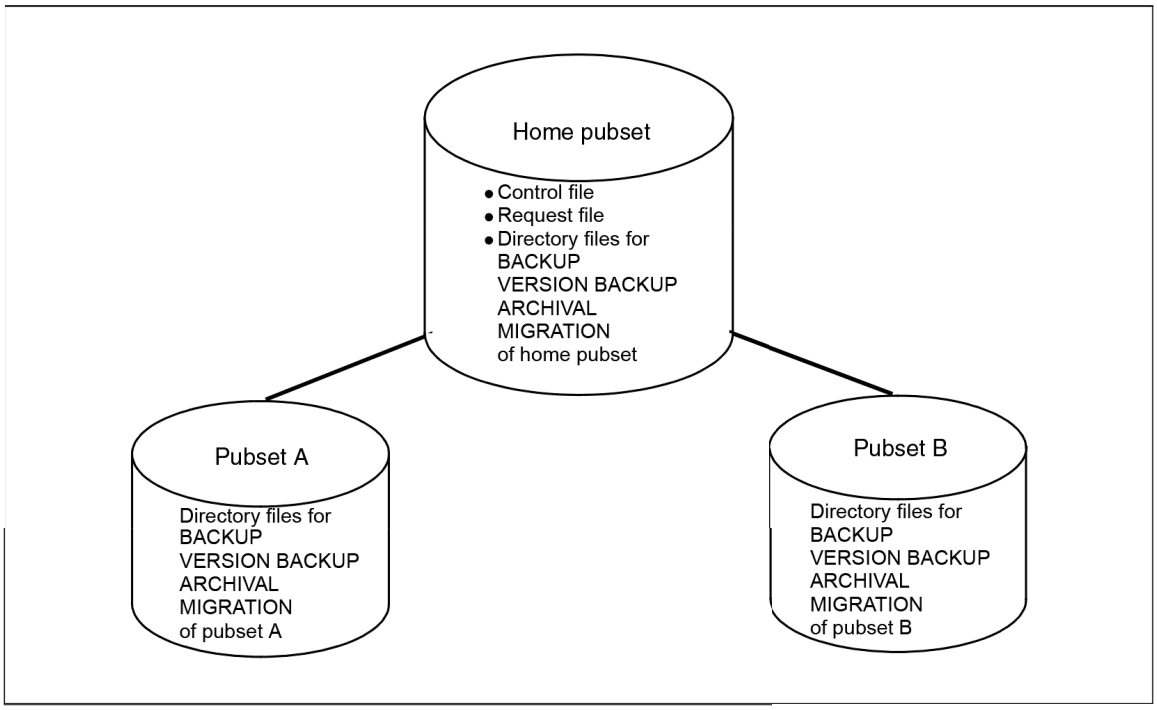


Figure 7: Decentralized administration

3.4.1.3 Advantages (+) and disadvantages (-) of each form of organization

Central administration	Decentralized administration
All pubsets are administered centrally: 1 directory file for migration 1 directory file for backup 1 directory file for archival	Each pubset is administered separately: 1 directory file for migration per pubset 1 directory file for backup per pubset 1 directory file for archival per pubset 1 directory file for version backup per pubset i Version backup is only possible within decentralized administration.
+ Backup and migration require only 1 HSMS statement for all pubsets.	- Backup, version backup and migration require 1 HSMS statement per pubset.
+ ARCHIVE automatically synchronizes the backup or migration of all pubsets.	- /SECURE RESOURCE-ALLOCATION is used on the output device to synchronize the tasks of different pubsets. If the pubsets are of different sizes, this may take additional time.
+ Optimum utilization of save volumes for small pubsets (1 volume per save run for multiple pubsets).	- Uneconomical utilization of save volumes for small pubsets (1 volume per save run per pubset). However, utilization of the volumes can be enhanced through continuation during the subsequent backups/migrations.
- Uses large directory files: Seriously negative impact on performance	+ Uses smaller directory files, improving runtime performance and reducing access conflicts.
- If a pubset is migrated to a different home pubset, the backup is no longer available without additional intervention (COPY-SAVE-FILE).	+ If a pubset is migrated to a different home pubset, the backup is still available. If the pubset does not contain the directory file, it must be copied.
- If the directory file is located on the home pubset and the pubset crashes, the directory file must be restored. Users cannot resume work until the HSMS environment has been completely restored.	+ If the home pubset crashes, processing can continue with a different pubset as soon as the control file has been restored. Processing can continue with another home pubset once the environment has been redefined; the directory files are retained.
- The serial processing of jobs in each archive has significant repercussions.	+ The serial processing of jobs has little effect.

- A request cannot be restarted after the home pubset has changed.
- The archive definitions are lost when the home pubset changes.

3.4.1.4 Recommended form of organization: decentralized administration

Decentralized administration permits the highest possible degree of flexibility in computer centers. However, tape processing under central administration is more cost-effective when using very small pubsets, since a smaller number of tape cartridges is required in this case.

In order to make it easier to restore a pubset after a system crash, the HSMS statements and commands defining the archives for backup, migration, etc., should be saved as a procedure file.

Notes regarding backups

1. A procedure file should be created for each pubset; the file should define an archive for migration, an archive for backup, and any other required archive types.

   Example

```
/BEGIN-PROCEDURE PARAMETERS=*YES(PROCEDURE-PARAMETERS=(&PVSID) -
/ ESCAPE-CHARACTER=C'&')
  /START-HSMS
  //CREATE-ARCHIVE ARCHIVE-NAME=MIGRATE.&PVSID, -
  // ALLOWED-USAGE=*MIGRATION,..., -
  // DIRECTORY-NAME=:&PVSID:$SYSHSMS.MIGRATION.DIR
  //CREATE-ARCHIVE ARCHIVE-NAME=BACKUP.&PVSID, -
  // ALLOWED-USAGE=*BACKUP(SAVE-FILE-STRUCTURE=*SEVERAL-SVID),..., -
  // DIRECTORY-NAME=:&PVSID:$SYSHSMS.BACKUP.DIR
  //CREATE-ARCHIVE ARCHIVE-NAME=ARCH.&PVSID, -
  // ALLOWED-USAGE=*ARCHIVAL,..., -
  // DIRECTORY-NAME=:&PVSID:$SYSHSMS.ARCHIVAL.DIR
  //CREATE-ARCHIVE ARCHIVE-NAME=VERS.&PVSID, -
  // ALLOWED-USAGE=*VERSIONBACKUP,..., -
  // DIRECTORY-NAME=:&PVSID:$SYSHSMS.VERSBACK.DIR
  //MODIFY-PUBSET-PARAMETERS PUBSET-ID=&PVSID, -
  // STORAGE-LEVEL=*S0(SYSMIGRATE=MIGRATE.&PVSID), -
  // SYSARCHIVE=ARCH.&PVSID, -
  // SYSBACKUP=BACKUP.&PVSID,-
  // SYSVERSION=VERS.&PVSID
  //END
/END-PROCEDURE
```

2. For system backup, the following HSMS statement must be specified for each pubset:

```
//BACKUP-FILES FILE-NAMES=:&PVSID:$*., -
// JV-NAMES=:&PVSID:$*.,...
```

3. To migrate all the files on the system, the following HSMS statement must be included for each pubset:

```
//MIGRATE-FILES FROM-STORAGE=*S0-STORAGE-LEVEL -
// (FILE-NAMES=:&PVSID:$*.,UNUSED=...,...)...
```

4. For version backup, the following HSMS statement must be specified for each pubset:

```
// BACKUP-FILE-VERSIONS FILE-NAMES=:&PVSID:$*.,...
```

5. File recall/restore must likewise be effected pubset by pubset.

Note on changing the home pubset

The archive definitions are lost whenever a pubset is exported and then imported. The directory files however, are still present on the pubset. Instead of the procedure shown above, as of HSMS V8.0 the archive attributes can be restored from the directory file with the CREATE-ARCHIVE statement (RECONTRUCT-ARCHIVE=*YES operand) (see also “Restoring an inadvertently deleted archive definition” in chapter "Deleting archive definitions"). If the migration or backup was performed on S1, the S1 pubset also has to be imported. The new system must also have access to storage level S2.

3.5 Relationship between SM pubsets and archives

For SM pubsets the archives for the backup, version backup, archival and migration functions must be administered decentrally.

In addition, all functions needed for the administration of the archives are stored in the SM pubset itself. This means that a control file that is specific to an SM pubset contains the archive definitions for that SM pubset. The archive directories assigned to the archives are also stored in the SM pubset.

Note

The standard system archive of an SM pubset for the long-term archiving of BS2000 files can also be defined in the global HSMS parameters. A long-term archive for BS2000 files which is defined in the SF environment can be used for both SF and SM pubsets.

3.6 Relationship between node S0s and archives

Passive node S0s offer the same flexibility as pubsets:

- Backup/archival of a node S0 can be managed from a central BS2000 system.
- The damage caused by unintentional destruction of an archive or directory file can be limited to a single node S0.
- Administrative work (backup, reorganization of the directory file etc.) performed on a node S0 should not impede other requests processed on different node S0s.
- Files and directory files can be moved from one node S0 to another.

3.6.1 Central and decentralized administration

Central and decentralized administration and the respective advantages and disadvantages are presented on the following pages.

Central administration	**Decentralized administration**
All node S0s are administered centrally: 1 directory file for backup 1 directory file for archival	Each node S0 is administered separately: 1 directory file for backup per node S0 1 directory file for archival per node S0
+ Only 1 HSMS statement is required for all node S0s during backup	- 1 HSMS statement per node S0 is required for backup
- Uses large directory files.	+ Uses smaller directory files, improving runtime performance and reducing access conflicts.
- If a node S0 is moved to another BS2000 server, the backup is no longer available without additional intervention (COPY-NODE-SAVE-FILE).	+ If a node S0 is moved to another BS2000 server, the backup is still available. The directory file must be copied.
+ ARCHIVE automatically synchronizes the backup or archival of all node S0s.	- /SECURE-RESOURCE-ALLOCATION synchronizes the tasks of different node S0s on the output device. Differences in node S0 sizes can cost time.
+ Optimum utilization of save volumes with small node S0s (1 volume per save run for multiple node S0s)	- Uneconomical utilization of save volumes with small node S0s (1 volume per save run per node S0) However, utilization of the volumes can be enhanced through continuation during the subsequent backups/migrations.

3.7 Save file

A save file is the “container” where data managed by the archive is stored. When HSMS saves data to cartridge, disk, or Net-Storage (regardless of the basic function for which it is intended), this data is entered in a save file. This means that HSMS does not create separate cataloged copies of the saved files.
In an HSMS save run, data is only ever written to a single save file, even if the save run involves a large number of files.

The save file itself consists of one or more catalogued BS2000 files on disk on one or more Net-Storage volumes or on cartridges. A save file on magnetic tape cartridges consists of a number of volumes having the same owner (i.e the same user ID is entered in the header of the magnetic tape cartridges) and the same retention period.
For the names of the save files created for HSMS by ARCHIVE, see the section "Save file names".

A save file on disk (storage level S1) is located under the user ID that contains the archive directory, connected with the archive where the save file is stored. To enhance portability, save files are created in the neutral NK4 format. This is done regardless of the format of the pubset (K/NK) or the setting of the BLKCTRL system parameter.

Each save file is uniquely identified by its save file ID (SFID). The save file ID is formed by the date and time of file creation as follows:

sfid = S.yymmdd.hhmmss

COPY-SAVE-FILE is a function which HSMS provides to copy save files (including excerpts of files) within the archive. This makes it possible, for example, to effectively reorganize older long-term archives. In BS2000/OSD-BC V9.0 and higher, the save file can also optionally be copied to an available Net-Storage. In HSMS V10.0 and higher, it can also be copied to shared disk.

To copy node save files of the POSIX file system or of remote nodes S0 within an archive, use the HSMS statement COPY-NODE-SAVE-FILE.

To copy save files created with the HSMS statement EXPORT-FILES or with the ARCHIVE statement EXPORT, use the HSMS statement COPY-EXPORT-SAVE-FILE.

A save file can contain either one or more save versions. This depends on the storage level of the save file and on the structure of the associated archive (SINGLE-SVID or SEVERAL-SVID).

Migration archives, long-term archives, version backup archives and archives for node files of the POSIX file system or of remote nodes S0 always have a SEVERAL-SVID structure. Backup archives can have either a SINGLE-SVID structure or a SEVERAL-SVID structure, depending on what is specified for the SAVE-FILE-STRUCTURE operand in the CREATE-ARCHIVE statement.

Save files for archives with a SINGLE-SVID structure and save files on storage level S1, on private disk, on pubset disk and on Net-Storage contain only a single save version.
Save files on storage level S2 for archives with a SEVERAL-SVID structure can contain more than one save version. When a save file containing multiple save versions is copied to storage level S1, private disk, shared disk, or Net-Storage, a separate save file is created for each save version of the original save file on the target medium.

3.7.1 Save file names

The names of the save files created by ARCHIVE for HSMS depend on the storage medium and, possibly, on the processing mode for save files (SAVE-FILE-PROCESSING):

- Magnetic tape or magnetic tape cartridge

  `ARCHIVE.SAVE.FILE(`*date-time-subsave#*`-0)`
- Private disk

  `ARCHIVE.SAVE.FILE.`*date*`.`*time*`.`*vsn*
- Public disk / S1

V9:	`ARCHIVE.SAVE.FILE.`*date*`.`*time*`.`*subsave#*	or
V10:	`ARCHIVE.SAVE.FILE.`*date*`.`*time*`.`*subsave#*`.`*seq#*	

- Net-Storage

V9:	`ARCHIVE.SAVE.FILE.`*date*`.`*time*`.`*vsn*	or
V10:	`ARCHIVE.SAVE.FILE.`*date*`.`*time*`.`*subsave#*`.0`	

Key:

date.time	Generation time in the format yymmdd.hhmmss
subsave#	Subsave number of the run which created this save file
seq#	Sequence number of the save file (0..F)
V9:	SAVE-FILE-PROCESSING=*HSMS-V9-COMPATIBLE
V10:	SAVE-FILE-PROCESSING=*HSMS-V10-COMPATIBLE

3.7.2 Retention period and expiration date

The backup and archival functions make saved data available as possible replacements for a certain period of time. The length of this period can be specified:
The retention period is the period of time during which a volume is protected against overwriting and thus the data stored on that volume is protected against deletion.

The operand RETENTION-PERIOD refers to the save files of an archive and serves to define the number of days after the completion of a save file during which the volume containing the save file is to be protected against overwriting.
In the migration and version backup functions, the retention period is determined only via the archive definition; in the other basic functions, it can also be specified in the corresponding HSMS statement.

The expiration date derives from the retention period. It is reached once the retention period has expired:

> **expiration date = creation date + retention period** (in days)
> (Expiration date = Creation date + Retention period)

In the case of a standard save file with NEW-STD-SAVE-FILE=*IN-PERIODS the continuation period must also be taken into account (see the section "Updating ("continuing") save files").

The expiration date is indicated on the volume, in the archive directory and, if applicable, in the MAREN catalog. The expiration date can be increased later in the HSMS archive directory; by doing so the expiration date of the volume in the MAREN catalogue is automatically increased as well. However, these changes cannot be carried out on the volume itself. Therefore, the volume is only protected after the original expiration date when MAREN is loaded and the volume is not moved to another system.

If the retention period of a save file on disk is increased, this protects the save file from deletion and the expiration date of its catalog entry is changed.

From HSMS V8.0A the retention period of a save file cannot just be increased but also reduced (SAVE-FILE-RETPD-UPDATE operand, also with a negative value). When the retention period of a save file is changed later, you must ensure that in backup archives the Cataloged-Not-Saved entries of files from incremental backups which have not been modified also require the reference to the last full backup, possibly in another save file, for restoration purposes. It does not make sense to release the save file with the full backups earlier, before the save file with the incremental backup.

In addition to the volume expiration date derived from the physical retention period, HSMS permits the definition of a file expiration date in connection with long-term archives. The file expiration date can be defined for a particular save version (and its files), independently of the retention period defined for the volume. A file expiration date may be useful, for instance, when reshuffling the save tapes: when reshuffling the cartridges, the archive owner may retain only those save versions whose logical expiration date has not yet been reached.

If, for instance, cartridge reshuffling takes place every 2 years and a retention period of 2 years has been defined, the cartridges can be released and used again for saving, without reinitialization, after copying the files whose logical expiration date has not yet been reached (for more information, refer to section "Copying within long-term archives").

The HSMS administrator can use the operand FILE-EXPIRATION-DATE to define for long-term archives whether the user-specified file expiration date may exceed the volume expiration date (FILE-EXPIRATION-DATE=*UNRESTRICTED).
Since a volume is no longer protected once the retention period has expired, the archive owner must take appropriate administrative measures to protect the volume, e.g. by using the SAVE-FILE-RETPD-UPDATE operand. This operand, which is recommended for archives with several save versions, allows the retention period of the volume to be automatically increased. The new retention period of the volume depends on the file expiration date of the newly created save version. This change also affects the MAREN catalog.

Note

Specifying the value 0 for a file's retention period can have undesirable consequences since this value indicates that the expiration date and creation date are the same. As a result, the save file has no retention period and can be deleted immediately after creation.

This status can lead to unexpected deletion and inconsistencies in the directory file: If the deletion of an archive whose retention period has the value 0 is initiated while an action statement is simultaneously issued for this archive (e.g. BACKUP-FILES), then the save file is deleted. However, since the action statement continues to run, a number of dependent records will be locked in the directory file and can therefore not be deleted. HSMS will be able to identify the inconsistencies later since the records remain in the directory file.

You should therefore be very careful when assigning a retention period of 0.

Retention period and expiration

When the retention period has elapsed (expiration date today or earlier) and all backup versions of a save file have reached their expiration date a save file is regarded as "obsolete" and can therefore be deleted using the MODIFY-ARCHIVE statement with a MAREN release of the volume in the event of tape backups.

As of HSMS V8.0A the option of automatically deleting obsolete save files is also provided implicitly for backups or copy operations in backup or long-term archives. This function is not available for migration and version backups. As of V11.0 MAREN also normally dispenses with a separate release run so that the release can take place immediately when the tape is deleted and it can be used again straight away for the subsequent backup or copy. Automatic deletion of obsolete save files is enabled with the archive attribute AUTOMATIC-DELETION = *OBSOLETE-SAVE-FILES.

When Several-SVID format and large cartridges are used it can be practical to update a backup file with a very large number of save versions. In this case at some point so many save versions starting at the start beginning of the save tape will be obsolete that it will be possible to release the first tape volume of the save file and reuse it. A finer function based on save versions is available for backup and long-term archives with the archive attribute AUTOMATIC-DELETION = *OBSOLETE-SAVE-VERSIONS: implicit deletion of obsolete save versions from the beginning of the save file in the archive directory in conjunction with the release of the tapes in the save file which are now no longer required.

With both variants implicit deletion takes place only when the save or copy operation is performed by the archive owner.

*Sketch of a save file on tape in Several-SVID format in the case of implicit deletion in accordance with the variant AUTOMATIC-DELETION = *OBSOLETE-SAVE-VERSIONS*

Assuming that SVIDs A to D each have an elapsed expiration date, SVIDs A to H are distributed to the backup tapes as follows:

Tape-1:	SVID-A	SVID-B	SVID-C	
Tape-2:	SVID-C remainder	SVID-D	SVID-E	SVID-F
Tape-3:	SVID-F remainder	SVID-G	SVID-H	Continue position

These obsolete save versions are deleted in the archive directory and Tape-1 is released. Tape-2 and Tape-3 are not released because these tapes do not yet contain elapsed SVIDs. Tape-3 is also needed for the next save run.

3.8 Save version

A save version is created each time a write request is issued for an archive. A save version contains all files and job variables saved by the request plus – in the event of incremental backup of files – references to those registered as “cataloged-not-saved” (CNS).
A save version is always part of a save file.

A save version is identified internally by its save version ID (SVID), which is formed by the date and time of its creation. HSMS statements can subsequently refer to a save version either via its creation date or via the name assigned to it in the action statement that resulted in the creation of the save version. The relevant operand in this case is SAVE-VERSION-NAME. Except for save versions created by means of version backups which can be referred only via their creation date. It is not possible to assign names to save versions of version backup archives; the specification of a value for the operand SAVE-VERSION-NAME in the MODIFY- ARCHIVE statement is rejected.

Note for ARCHIVE users

The distinction between save file and save version has been introduced in HSMS because in contrast to ARCHIVE, several backup versions can be contained in a single save file for backup, archival and migration.

When processing magnetic tape cartridges of this type using ARCHIVE, the save file ID (SFID) in the report is output as a save version ID (SVID). If the user wishes to import individual files, then they will need to specify the output SVID as the SFID when carrying out an IMPORT run.
The SVID can be located using the HSMS statements LIST-VOLUMES. If the SVIDs are listed in an ordered sequence on the magnetic tape cartridge, then the user can use the LIST statement of ARCHIVE to obtain the SVIDs of the various files by specifying the SFID.

3.8.1 Save type

The basis for file management in HSMS is the save version. Each time a save request is executed, the processed files are assigned a save type, which is output in response to the HSMS statements SHOW-ARCHIVE and SELECT-FILE-NAMES and in the reports. Note, however, that the reports do not differentiate between CNS and OPER and output CNS only.

Save type	Meaning
FULL	The complete file has been saved.
FULB	The complete file has been saved. Metadata of library elements has also been saved.
PART	Only the modified parts of the file have been saved.
PARB	Only the modified parts of the file have been saved. Metainformation from library members has also been saved.
MIGF	The file has migrated, the catalog entry has been saved.
FGGI	The index of a file generation group has been saved.
CATL	Only the catalog entry for the file was saved.
FULJ	A job variable was saved.
CNS	The file is cataloged but has not been saved.
OPER	Subset of CNS. The file has not been saved due to an error at file opening (listed under CNS in reports).
MDS	The node file was modified during backup. *Note* The modification cannot be identified if the file is opened before being processed by ARCHIVE and not closed until after the ARCHIVE run.
FNOD	FULL-NODE-FILE, SAM node files saved with SAVE-SAM-STRUCT=*NO (default, RAW mode). These SAM node files can only be restored on Net-Storage (FILE-TYPE=*NODE-FILE) (HSMS 11.0 A and BS2000 OSD/BC 11.0A or higher).

3.8.2 Compressing save versions

In order to better utilize the capacity of backup volumes, HSMS provides the option of compressing the data before it is written to a save file. The COMPRESS-FILES operand in the HSMS statements ARCHIVE-FILES, BACKUP-FILES, BACKUP-FILE-VERSIONS, EXPORT-FILES and MIGRATE-FILES tells HSMS to start software compression.

The archive owner can use the archive default setting to define the default value for the appropriate HSMS statements and also determine that save versions are compressed only when written to storage level S1 but not when written to S2 (COMPRESS=*S1-ONLY).

No entry need be made when reading the data, e.g. during restore. HSMS automatically decompresses compressed backups.

The degree of compression depends on the saved files, e.g. text files are more heavily compressed than are load modules. Although compression increases load on the CPU, it also reduces input/output operations and the number of backup volumes. Therefore, compression does not save time during backup, but rather reduces the number of volumes required for backup.

Modern MTC devices already perform compression using their hardware. In this case HSMS does not perform software-based compression and ignores instructions to do so in the COMPRESS operand.

3.8.3 Extending save versions

Extending save versions is only relevant for backup archives with a SINGLE-SVID structure and for version backup archives if the save file, containing a save version, is on a disk storage, i.e. storage level S1, on public disk or Net-Storage.

A save version can be extended by means of the operand SAVE-FILE=*CONTINUE

- of the BACKUP-FILES statement if the archive involved has a SINGLE-SVID structure.
- of the BACKUP-FILE-VERSIONS statement if the save file is on disk storage or Net-Storage.

However, save versions can only be extended by a disjunctive number of files.

In case of version backup archive, only the very last save version can be extended.

3.9 Standard save file

The standard save file is only significant in the context of SEVERAL-SVID archives.

Unless another save file is explicitly specified in the save statement, the save versions of an HSMS save run are written to the standard save file of an archive.

If the user does not specify otherwise in the save statement, all the data saved in a long-term archive or backup archive with a SEVERAL-SVID structure is written to the standard save file. The retention period for this standard save file is defined in the archive definition.

Migrated data is always written to the standard save file.

The name of the standard save file is entered in the archive definition of the relevant archive and is considered to be an archive attribute.

3.9.1 Updating ("continuing") save files

For requests which refer to storage level S2 and concern SEVERAL-SVID archives, HSMS offers the option of updating save files that already exist. In this case the current save version is written to cartridge behind the save versions already contained there. The new data is written to the volume behind the existing blocks. The save versions already contained in the save file are not changed in the process. The new data is written to the volume behind the existing blocks.

Whether and to what extent a standard save file is continued depends on the intervals at which a new standard save file is started for the archive. These are controlled by the archive definition (operand NEW-STD-SAVE-FILE):

- If *AT-EACH-REQUEST is specified for NEW-STD-SAVE-FILE, i.e. a new standard save file is started with each save request, the save file is not continued at all. Each save version is written to a separate save file.
- If *EACH-TAPE-SESSION is specified for NEW-STD-SAVE-FILE, i.e. a new standard save file is started at the beginning of each tape session, the save file is continued until the end of the tape session. This means that, during a tape session, all save versions are normally written to the same save file.
- A continuation period in days can be defined by using *IN-PERIODS and the operand CONTINUATION-PERIOD. The default save file is always switched at the beginning of each continuation period. It is continued until the end of the continuation period. All save versions are written to this save file during this period unless otherwise specified.

  The expiration date of a save file is calculated as follows if a continuation period is defined:

  Expiration date = creation date + retention period + continuation period
  (Expiration date = Creation date + Retention period + Continuation period)

  HSMS creates all save files with SEVERAL-SVID structure as continuable files, irrespective of the specification in the archive definition. This enables save files to be continued explicitly by the HSMS administrator using an HSMS statement. This can be done with various HSMS statements by specifying SAVE-FILE=*CONTINUE.
- *PUBLIC-DISK(...) enables the catalog ID of a pubset to be agreed on for a backup node archive. Because of this entry the BACKUP-NODE-FILES statement and the reference to the standard save file (specification SAVE-FILE=*STD) cause backup jobs to be executed for node as a backup on the defined pubset.
- Save files which were created on storage level S1, on shared disk, or on Net-Storage in HSMS version V9.0B and earlier can only be continued in HSMS V10.0A and higher if SAVE-FILE-PROCESSING=*HSMS-V9-COMPATIBLE (default) is set.
- Save files on storage level S1, on shared disk, or on Net-Storage which were created with *HSMS-V10-COMPATIBLE mode cannot be read or continued in HSMS V9.0B or earlier.
- Save files which were created on the extended storage level S1 (formed from all volume sets of the SM pubset under HSMS management) can only be continued with SAVE-FILE-PROCESSING=*HSMS-V10-COMPATIBLE. This also applies when the S1 level has been reduced to a specific volume set again. In this case the volume set last used is continued.
- Save files can only be continued if the same value has been specified for the SAVE-SAM-STRUCTURE operand as for the save file that is to be continued.
- In case the last SFID/SVID is inconsistent due to a DMS error, the current save file cannot be continued. A new save file has to be created.

3.10 Archive management

Archive management is essentially the management of the objects saved in the archives, above all the save files and volumes. In the case of migration and version backup, reorganizing is also required (see the sections "Reorganizing the migration archive" and “Reorganization of version backups”).

By using the HSMS statement SHOW-ARCHIVE, the *archive owner* can obtain information about the objects managed in the archive, i.e. optionally files, job variables, save versions and save files, and about volumes. *Nonprivileged users* can use this HSMS statement with public archives to obtain information about the objects belonging to their own user ID or objects of which they are co-owners.

In SHOW-ARCHIVE, a marking function can be used to request information on individual save files and volumes when the archive summaries are output.

3.10.1 Volume pool management

Note

If the software product MAREN is being used, the volumes are administered by MAREN and in particular assigned to the archive directories (see the “MAREN” manual [10], as well as the section "Restrictions due to new functions"). The volume management of HSMS is thus integrated in the computer-center-wide volume management. In this case you should ignore the volume management by HSMS described in this chapter.

A number of cartridges can be assigned to an archive, and these form the archive's volume pool.

`/MODIFY-ARCHIVE VOLUMES=*ADD` adds volumes of the class “TAPE” to an archive by specifying their VSNs and device types. They are then entered in the free volume pool. Unless otherwise specified, the save volumes for write requests referring to the archive are fetched from this free volume pool.

`//MODIFY-ARCHIVE VOLUMES=*REMOVE` removes volumes from the free volume pool.

The volumes added to the archive are listed with the remark POOL in the OWNER column of the output displayed in response to `//SHOW-ARCHIVE SELECT=*VOLUMES`.

Volumes that were not taken from the volume pool to execute an archive write request but were explicitly requested by the user via their VSN or assigned by the operator are only temporarily added to the archive directory. These volumes are listed with the remark OPERATOR in the OWNER column (see section "Allocating volumes").

The volumes in a volume pool may have any of the following states:

State	Meaning
AVAILABLE	Either the volume has not been used since being added to the archive or the save file has been deleted and the retention period has expired (free volume pool).
IN-USE	The volume is occupied by a save file whose retention period has not yet expired.
OBSOLETE	The VSN's retention period has expired. However, the save file has not been deleted yet.
FORCED-DELETE	The volume has been released although the retention period has not yet expired.
UNUSABLE	The volume cannot be used due to an error.

If a volume whose state is FORCED-DELETE is to be written to before the retention period has expired, it must be removed from the pool, reinitialized and returned to the pool.
A magnetic tape cartridge in the state UNUSABLE (or the VSNs of such tapes) cannot be used again until it has been removed from the pool and then added to it again.

3.10.2 Releasing save files

Save files whose retention period has expired can be released (deleted) from the archive by the archive owner using the SAVE-FILES operand of the MODIFY-ARCHIVE statement. A save file can only be released as a whole, even if it contains more than one save version.

Individual save files can be selected for deletion by specifying their save file ID or via specific attributes. However, the save files can also be selected by means of their attributes. Consequently all save files whose retention period has expired can be deleted using the following HSMS statement:

```
//MODIFY-ARCHIVE SAVE-FILES=*DELETE -
// (SAVE-FILE-ID=*BY-ATTR(SAVE-FILE-STATE=*OBSOLETE))
```

Released save files are removed from the archive. The volumes of level S2 save files are returned to the free volume pool. S1 save files or save files on private disk are deleted. However, a save file will not be removed:

- from version a backup archive if it contains at least one valid file version;
- from a migration archive if it contains data from currently migrated files.

A save file can be released even if its retention period has not yet expired or, in the case of migration or version backup archives, if it contains valid files or file versions respectively.

```
//MODIFY-ARCHIVE SAVE-FILES=*DELETE(...,FORCED-DELETE=*YES)
```

The save file is deleted without any further checks. The associated volumes in the archive's volume pool are assigned the state FORCED-DELETE until the originally defined retention period has expired. If you are working with an archive without a volume pool, the volumes are also released without any further check when the above-mentioned statement is issued.

3.10.3 Accessing archive objects without an archive directory

HSMS only supports full access (reading and writing) to magnetic tape cartridges if the associated archive directory can be accessed. If there is no longer an archive directory available, the volumes should not be accessed for writing. Although, read access is still possible:

- When migrated files are recalled, the archive directory is not accessed, even if one is available. All the information required for the recall is found in the catalog entry.
- Saved data can be recalled as follows:
  - with the HSMS statement IMPORT-FILES:

    ```
    //IMPORT-FILES ...,SAVE-FILE=*BY-VOLUME(SAVE-FILE-ID=svid)
    ```

    or

    ```
    //IMPORT-FILES ...,SAVE-FILE=*BY-PUBLIC-DISK(
                               SAVE-FILE-ID=svid,PUBSET-ID=cat_id)
    ```

    If backup took place to pubset or storage level S1
  - with the ARCHIVE statements FILES and RESTORE:

    ```
    FILES RESTORE ...,DIRECTORY=NONE,FROM=(vsn) or FROM=svid,(vsn)
    ```

3.10.4 Deleting archive definitions

The HSMS administrator or the archive owner can delete archive definitions using the HSMS statement DELETE-ARCHIVE. Note, however, that this only deletes the archive definition from the HSMS control file.
Archive definitions cannot be deleted until there are no more outstanding requests for the archive in the request file.

With DELETE-ARCHIVE the following functions are also available:

- **Separating the shadow archive:** The connection between an archive and the associated shadow archive is canceled and the shadow archive is turned into a backup or long-term archive. The attributes of the archive originally assigned are taken over for the new archive.
- **Replacing the archive directory:** The definition of the shadow archive is deleted and the archive directory of the shadow archive is taken over for the assigned backup or long-term archive. The original archive directory is retained without any link to an archive.

DELETE-ARCHIVE deletes neither the archive directory nor the files managed in this directory. The archive directory can be deleted using the BS2000 command DELETE-FILE.

Restoring an inadvertently deleted archive definition

The attributes of an archive are also stored in the archive directory. If the archive definition has been inadvertently deleted the archive can be restored from the archive directory using the CREATE-ARCHIVE statement:

```
//CREATE-ARCHIVE ARCHIVE-NAME=...,
   DIRECTORY-NAME=<directory>(NEW-DIRECTORY=*NO(RECONSTRUCT-ARCHIVE=*YES))
```

3.11 Management classes

A management class is a named collection of attributes defined by the HSMS administrator. It is used to control backup, version backup and migration processing in an SM pubset that is under HSMS control.

3.11.1 Assigning a management class

A whole series of attributes can be assigned to a single object (file, job variable or file generation group) by assigning it the name of a management class. The management class defines a number of backup and migration attributes for that object. A management class can only be assigned to objects located on SM pubsets. A management class cannot be assigned to objects on a private disk, a tape or an SF pubset.

A management class can be assigned to a *file* with the DMS commands CREATE-FILE and MODIFY-FILE-ATTRIBUTES. For a *job variable* with the DMS commands CREATE-JV and MODIFY-JV-ATTRIBUTES and for a *file generation group* with the DMS commands CREATE-FILE-GROUP and MODIFY-FILE-GROUP-ATTRIBUTES.

3.11.2 Attributes of a management class

Each management class contains all its backup, version backup and migration attributes. When the name of a management class is assigned to an object, the entire HSMS strategy is thus also assigned to it.

A distinction is made between the following attributes:

1. Attributes which define preconditions to be met before objects of an SM pubset are eligible for backup or migration (e.g. the attribute UNUSED-DAYS for migration).
2. Attributes which define the required result of a backup, a version backup or a migration (e.g. the attribute TO-STORAGE for migration).

All the attributes are described below. The possible values of an attribute can be found in "HSMS Vol. 2" [1] under the HSMS statement CREATE-MANAGEMENT-CLASS.

- **Attribute for backup**

  RETENTION-PERIOD (type 2)
  Minimum retention period for the save file that contains the object.

- **Attributes for migration from S0 level (not for job variables)**

  UNUSED-DAYS (type 1)
  Minimum number of days an object must remain unused to be eligible for migration to a background level.

  MINIMUM-SIZE (type 1)
  Minimum number of PAM pages an object must have to be eligible for migration to a background level.

  MAXIMUM-SIZE (type 1)
  Maximum number of PAM pages an object may have to be eligible for migration to a background level.

  MINIMUM-EXTENTS (type 1)
  Minimum number of extents a file must have to be eligible for migration to a background level.

  TO-STORAGE (type 2)
  Background level to which the object is to be migrated.

- **Attributes for migration from S1 level**

  MINIMUM-DAYS-ON-S1 (type 1)
  Minimum number of days an object must be on S1 to be eligible for further migration to S1 or S2.

  MAXIMUM-DAYS-ON-S1 (type 1)
  Maximum number of days an object may be on S1 to be eligible for further migration to S2.

  MINIMUM-SIZE (type 1)
  Minimum number of PAM pages an object must have to be eligible for migration to S1 or S2.

  TO-STORAGE (type 2)
  Background level to which the object is to be migrated.

- **Attributes for version backup**
  NUM-OF-BACKUP-VERS (type 2)
  The number of backup versions, which are kept within a version backup archive.

3.11.3 How the management classes are used

A management class can be defined with the HSMS statement CREATE-MANAGEMENT-CLASS. The attribute values can be displayed with the HSMS statement SHOW-MANAGEMENT-CLASS-ATTRIBUTES and changed with MODIFY-MANAGEMENT-CLASS-ATTRIBUTES.
Nonprivileged users are only allowed to use the HSMS statement SHOW-MANAGEMENT-CLASS-ATTRIBUTES.

With SM pubsets, management classes can only be used if HSMS is started on the master host.

To control a backup with management classes, the HSMS administrator must specify the name of a management class in the HSMS statement BACKUP-FILES; for a migration the administrator must specify the name of a management class in the HSMS statement MIGRATE-FILES; for a version backup the administrator must specify the name of management class in the HSMS statement BACKUP-FILE-VERSIONS. Only the objects that have been assigned the name of that management class are selected for processing. This procedure readily allows a subsystem to be defined. By subsystem, we mean a group of objects, all of which HSMS processes in the same way.

The operands of the HSMS statement that correspond to an attribute must be set to the value *STD. This ensures that the operand values are taken from the management class.
At the appropriate time, the HSMS administrator executes the requisite HSMS statements. The operand values of these HSMS statements are then replaced by the corresponding attributes of type 1 or 2 of the respective management class.

Note that management classes created in HSMS V11.0A and lower do not contain attributes for version backups. The files assigned to such management classes are processed by BACKUP-FILE-VERSIONS as if they have NUM-OF-BACKUP-VERS equal to 0, i. e. the files are skipped if they have never been saved within the version backup archive or, in case the file has been saved within the archive, 0 is written as NUM-OF-BACKUP-VERS into the directory. It is therefore recommended to provide management classes with version attributes if they are used in the context of version backup (see MODIFY-MANAGEMENT-CLASS).

Management classes created or modified in HSMS V12.0A still can be used in lower HSMS versions without any restriction for current behavior of the lower HSMS versions.

3.11.4 Management class protection

If the software product GUARDS is being used, management classes can be protected by guard profiles. This means that use can be restricted to specific management classes.

A guard profile must first be defined with GUARDS before the HSMS administrator can assign it to a management class. The same environment must be used for the guard profile as for the management class.

For more detailed information on GUARDS, see the “SECOS” manual [16].

3.12 Processing directory files with DIRCONV

The DIRCONV conversion program (task unprivileged) allows you to consistently modify the information contained in a directory file without a SAVE/RESTORE run. It also lets you merge several directory files.

3.12.1 DIRCONV functions

DIRCONV offers the following functions:

- Merging directory files
- Converting directory files
- Renaming catalog IDs
- Delete catalog IDs
- Outputting catalog IDs
- Updating the volume catalog
- Delete noncataloged files
- Converting repository files from the old to the new format
- Reorganizing directory files or repository files

DIRCONV runs under

- the systems support user ID TSOS.
- every other user ID.
  Only directory files, which do not contain any files or job variables registered by foreign users can be processed.
  If a directory file does contain files or job variables registered by foreign users, the run will be aborted with the following message:

```
RECORDS OF FOREIGN USER FOUND. RUN TERMINATED
```

In the case of password-protected directory files, the password must be entered (this also applies for the TSOS user ID).

3.12.1.1 Merging directory files

The DIRCONV statement MERGE-DIRECTORIES (see "MERGE-DIRECTORIES Merging directory files") can be used to merge multiple directory files into a single new directory file. This new directory file will contain all the information necessary to recreate, retrieve and recall files that were saved under one of the original directory files. With this new directory file you can continue to carry out backup, archival and migration. However, version backup directories cannot be merged.

During the merge process name conflicts may occur:

- If you enter `SILENT-RUN=*NO`, processing is interrupted and an error report is output to SYSOUT if a name conflict occurs.
- If you enter `SILENT-RUN=*YES`, DIRCONV checks the complete input directory for name conflicts and outputs these in a list.

The MAREN catalog is not updated automatically. The name of the directory file is not changed in the volume entries that belong to the original directory file. Therefore, you must use the UPDATE-VOLUME-CATALOG statement (see "UPDATE-VOLUME-CATALOG Updating the volume catalog") during a successful merge process in order to obtain consistency.

Note

For the opposite function, i.e. splitting a directory file into multiple files, is not possible.

3.12.1.2 Converting directory files

The DIRCONV statement SET-CATID (see "SET-CATID Converting directory files") can be used for converting directory files.

For ARCHIVE runs with a directory file this means:

- If the first save run was executed with `PARAM CATID=YES` (to support MPVS), all subsequent runs must also be executed with `PARAM CATID=YES`.
- If the first save run was executed with `PARAM CATID=*NO`, all subsequent runs must also be executed with `PARAM CATID=*NO`.

With DIRCONV you can convert directory files in `CATID=*NO` mode to `CATID=*YES` mode. A conversion in the opposite direction however, is not possible.

If the directory file is already in `CATID=*YES` mode, the run is terminated with the following message:

```
DIRECTORY IS ALREADY CONVERTED
```

This is also the case if the directory file was created with the ARCHIVE statement POOL and if no save runs have been executed with it yet. You can use this type of directory file immediately in `CATID=*YES` mode.

Due to the different CATID modes, the CONTINUE statement cannot be used to continue, with a converted directory file, save versions created before the conversion. In the report of an INQUIRE run with SV=... these save versions appear with the remark

```
CREATED BEFORE DIRECTORY CONVERSION
```

Otherwise, converted directory files behave in the same way as those created with `CATID=*YES`; every continuation run must be executed in `CATID=*YES` mode.

The beginning of the conversion run proper is indicated by displaying the user ID under which DIRCONV was started and the associated default catalog ID. DIRCONV also issues a message if the directory file contains neither files nor job variables. DIRCONV also issues a message if the directory file contains neither files nor job variables. In this case the directory file is nevertheless converted.

If a conversion run is performed under TSOS, all listed user IDs together with the catalog ID determined by DIRCONV are displayed in the order in which the user IDs occur in the directory file. In addition, every user ID to which the default catalog ID of TSOS was allocated because it was deleted at DIRCONV runtime is logged to SYSLST.

If a file name or JV name, including the new catalog and user IDs, is longer than 54 characters, the following message is issued:

```
***FILENAME TOO LONG: $userid.filename
```

or

```
***JVNAME TOO LONG: $userid.jvname
```

The file name or JV name is entered in the new directory with the dummy catalog ID S. The user thus has to process these entries later by means of `//RENAME-CATID`.

A message is also issued if the directory file does not does not contain a save version.

The end of conversion is reported by the message:

```
DIRECTORY CONVERSION FINISHED.
```

If a DIRCONV run is not performed under TSOS, a file with the name `S.tsn.DIRCONV.@.VNOCRID` is created, unless this file already happens to exist. You can delete this file if you wish.

DIRCONV method of operation when converting to CATID=*YES mode

DIRCONV inserts the default catalog identifier of the appropriate user ID into every FILES/JOBVAR record.

If the conversion is performed under TSOS, those user IDs not found in the user catalog will be output to SYSLST in addition.

Notes on systems support

- If at DIRCONV runtime a user ID is not entered in the user catalog (deleted), no default catalog ID can be determined. The FILES/JOBVAR records of the user ID are then supplemented by the TSOS default catalog ID.
- During a DIRCONV run the default catalog IDs for the users concerned should not be changed.
- Any existing SVID records are labeled “created before conversion”.
- The directory file is labeled “to be used in `CATID=YES` mode only”.
  When converting directory files from old BS2000 versions, file names may become too long if they are preceded by a catalog ID. In total the file name plus catalog and user IDs may not be longer than 54 characters.
  The following then applies to the length of the file names:
  - If the catalog ID is 1 character long:
    The file name must not exceed 41 characters without the catalog ID and user ID.
  - If the catalog ID is longer than 1 character (2 to 4 characters):
    The maximum possible length of 41 characters is reduced if in a multicharacter catalog ID the sum of the length of the catalog ID (including colons) and length of the user ID (including $ characters and period) is longer than13 characters (*:catid:$userid.*).

If DIRCONV finds a file name which is too long during conversion, “S” is used as the dummy catalog ID. If the file name in the directory file is no longer unambiguous because of this, DIRCONV aborts the run. The text `COPY ERROR, DMS051A` is then output to SYSOUT.

The directory file does not allow files to be read in if their names are too long.

3.12.1.3 Renaming catalog IDs

The DIRCONV statement RENAME-CATID (see "RENAME-CATID Renaming catalog IDs") can be used to rename a specific catalog ID or all catalog IDs in a directory file.
If you want to rename a specific catalog ID, you have the option of indicating whether only a specific user ID or all user IDs should be changed on this pubset.

This statement can be used in the following cases:

- An SM pubset is created which consists of various pubsets that have been saved earlier with a shared directory file. All catalog IDs in this directory file must be renamed as a single shared SM pubset ID.
- If the standard catalog ID of a user ID is changed, the associated files must be transferred to the new catalog ID, regardless of whether the files have been migrated or not. The information on these files, which is contained in the directory file, must be converted in order to take the change of the catalog ID into account.
- If a file name or JV name, including the new catalog and user IDs, is longer than 54 characters, the following message is issued:

  ```
  ***FILENAME TOO LONG: $userid.filename
  ```

  or

  ```
  ***JVNAME   TOO LONG: $userid.jvname
  ```

  The file name or JV name is entered in the new directory with the dummy catalog ID S. The user thus has to process these entries later by means of `//RENAME-CATID`.

3.12.1.4 Removing catalog IDs

The DIRCONV statement REMOVE-CATID (see "REMOVE-CATID Delete catalog IDs") can be used to remove information from a directory file about all files which reside on a specific pubset. Files resident on other pubsets are not affected and can still be restored.

This function allows you to prevent name conflicts which may occur when merging directory files or when renaming catalog IDs.

A PURGE statement (ARCHIVE run) or a MODIFY-ARCHIVE statement (HSMS run) is still possible and must be issued in order to release the respective data volumes. This is required even if all files that reside on the volume have been removed from the directory file.

In general, a catalog ID should not be removed until the saved files have been copied into another archive using the HSMS statement COPY-SAVE-FILE, which requires the use of another directory file.

3.12.1.5 Displaying catalog IDs

The DIRCONV statement SHOW-CATID (see "SHOW-CATID Outputting catalog IDs") can be used to output catalog IDs which are contained in an existing directory file. This information may be helpful in order to identify potential name conflicts prior to merging directory files.

3.12.1.6 Updating the volume catalog

The DIRCONV statement UPDATE-VOLUME-CATALOG (see "UPDATE-VOLUME-CATALOG Updating the volume catalog") can be used to eliminate inconsistencies between the MAREN catalog and an ARCHIVE directory file with regard to the directory name: all volumes that belong to a directory file receive the same directory name in their respective catalog entry.

For every volume listed in the directory file which was entered in the NEW-DIRECTORY-NAME operand, the following MAREN statement is issued in MARENADM

```
//MODIFY-VOLUME-ATTRIBUTES VOL=...,DIR-NAME=...
```

In addition, for each directory file entered in the DIRECTORY-NAMES operand by the MAREN administrator, the following MAREN statement is issued in MARENADM:

```
//MODIFY-VOLUME-ATTRIBUTES VOLUME=*ALL,SELECT=FREE(
             ARCHIVE-USAGE=old_file_name,NEW-ARCHIVE-USAGE=new_file_name)
```

This assigns the pool of prereserved free volumes to the new directory name.

This function is useful for eliminating discrepancies which may, for example, have resulted from merging multiple directory files into a new directory file.

Notes

- If the MAREN administrator has forgotten one or more old directory names, he/she must change the prereservation by entering the same MARENADM statement for the missing names.
- Directory names for volumes that have been entered in the new directory file do not need to be updated if the old directory names have been forgotten.

3.12.1.7 Delete noncataloged files

The DIRCONV statement REMOVE-UNCATALOGED-FILES (see "REMOVE-UNCATALOGED-FILES Delete noncataloged files") can be used to remove those files from a directory file which are no longer cataloged.
This prevents, for example, name conflicts if multiple directory files are merged. A message is issued for every file that was removed.

If any of the catalog IDs cannot be accessed, all files will be preserved in the directory file. Processing is then continued with the next catalog ID listed in the directory file.

3.12.1.8 Reorganizing directory and repository files

The DIRCONV statement REORGANIZE-DIRECTORY enables directory and repository files to be reorganized.

In this case empty data records which may have been created by deleting save versions are removed (with //MODIFY-ARCHIVE ... SAVE-FILE=*DELETE). This is necessary to prevent the maximum number of sets for one file and/or job variables (JV) from being exceeded. When the maximum number (256) for a file or JV is reached, no further information can be written to the directory for this file and/or job variable, which is displayed by the message ARC0176.

The maximum number can in particular be reached when the same files are repeatedly saved to the same archive. If old save versions are regularly deleted in this case, the reorganization of the directory file can significantly reduce the number of data records and thus prevent the directory file from overflowing.

It is consequently recommended that reorganization be performed regularly using the DIRCONV statement REORGANIZE-DIRECTORY.

When reorganization takes place, a copy of the original directory file is created. Following successful reorganization, this copy can be used in place of the original directory file.

3.12.1.9 Converting repository files

The DIRCONV statement REORGANIZE-DIRECTORY enables repository files to be converted from the old format (HSMS < V7.0) to the new format (HSMS V7.0 or higher). The new format offers significantly better performance for backup operation with long path names of node files (Unix). All save files remain unchanged here. In the associated node archive only the existing directory needs to be replaced by the new directory from the DIRCONV conversion.

3.12.2 Input/output and calling DIRCONV

Output from DIRCONV is directed to SYSOUT.

DIRCONV can only process directory files. Other file types are rejected. The directory files for input and output must not be identical.

DIRCONV is called with:

```
/START-DIRCONV
```

or

```
/START-EXECUTABLE-PROGRAM $DIRCONV
```

DIRCONV then checks the specified input file to ascertain whether it is a directory file. If it is not, the run is terminated and the following message is issued:

```
FILE EMPTY OR NOT IN CORRECT FORMAT
```

Any errors occurring during a system function call or data access will cause DIRCONV to terminate abnormally. An appropriate message to be output.

Large directory files should always be reorganized after a DIRCONV run.

3.12.3 DIRCONV statements

The following section describes the DIRCONV statements in alphabetical order.

3.12.3.1 MERGE-DIRECTORIES Merging directory files

This statement can be used to merge multiple directory files into a single file.

MERGE-DIRECTORIES
DIRECTORY-NAMES = list-poss(10): <filename 1..54> ,**NEW-DIRECTORY-NAME =** <filename 1..54> ,**SILENT-RUN** = ***NO** / ***YES**

DIRECTORY-NAMES = list-poss(10): <filename 1..54>
Fully qualified path names of the directory files to be merged.

NEW-DIRECTORY-NAME = <filename 1..54>
Fully qualified path name of the new directory file.

SILENT-RUN = *NO / *YES
If the value is *NO, a message is output if an error occurs. The run is terminated.

If the value is *YES, the run is only simulated: the input directory is read in full but no output file is created. Every error (e.g. name conflict) that is identified during execution is logged.

3.12.3.2 REMOVE-CATID Delete catalog IDs

This statement can be used to remove from a directory file information on all or selected files that resided on a specified pubset at any given save time.

REMOVE-CATID
DIRECTORY-NAME = <filename 1..54>
,**CATID** = <cat-id>
,**USER** = ***ALL** / <name 1..8>
,**FILE-NAME** = ***ALL** / ***NONE** / <filename 1..80 with-wild>
,**JV-NAME** = ***ALL** / ***NONE** / <filename 1..80 with-wild>
,**NEW-DIRECTORY-NAME** = <filename 1..54>

DIRECTORY-NAME = <filename 1..54>
Fully qualified path name of the directory file from which a catalog ID is to be removed.

CATID = <cat-id>
Catalog ID of the files to be removed.

USER =
User whose files are to be removed from the directory file.

USER = *ALL
All users whose files reside on the specified catalog ID are to be removed.

USER = <name 1..8>
ID of the user whose files are to be removed.

FILE-NAME =
Name of the files which are to be removed from the directory file.

FILE-NAME = *ALL
All files of the specified users are to be removed.

FILE-NAME = *NONE
No files are to be removed.

FILE-NAME = <filename 1..80 with-wild>
The name of the files which are to be removed is entered directly. It must be fully or partially qualified without a catalog ID and user ID. This name is extended by the catalog ID and user ID which are specified in the other operands.

JV-NAME =
Name of the job variables to be removed from the directory file.

JV-NAME = *ALL
All job variables of the specified users are to be removed.

JV-NAME = *NONE
No job variables are to be removed.

JV-NAME = <filename 1..80 with-wild>
The name of the job variables which are to be removed is entered directly. It must be fully or partially qualified without a catalog ID and user ID. This name is extended by the catalog ID and user ID which are specified in the other operands.

NEW-DIRECTORY-NAME = <filename 1..54>
Fully qualified path name of the new directory file.

3.12.3.3 REMOVE-UNCATALOGED-FILES Delete noncataloged files

This statement can be used to remove those files from a directory file which are no longer cataloged.

REMOVE-UNCATALOGED-FILES
DIRECTORY-NAME = <filename 1..54> ,**NEW-DIRECTORY-NAME** = <filename 1..54> ,**SILENT-RUN** = ***NO** / ***YES**

DIRECTORY-NAME = <filename 1..54>
Fully qualified path name of the directory file from which the uncataloged files are to be removed.

NEW-DIRECTORY-NAME = <filename 1..54>
Fully qualified path name of the output directory file.

SILENT-RUN = *NO / *YES
If the value is *NO, a message is output if an error occurs. The run is terminated.

If the value is *YES, the run is only simulated: the input directory is read in full but no output file is created. Every error (e.g. name conflict) that is identified during execution is logged.

3.12.3.4 RENAME-CATID Renaming catalog IDs

This statement can be used to change the catalog ID in a file or job variable name. You can restrict the set of names to be changed by specifying a catalog ID and user ID

```
RENAME-CATID

 DIRECTORY-NAME = <filename 1..54>

 ,OLD-CATID = *ALL / *BY-SPECIFICATION(...)

      *BY-SPECIFICATION(...)

           |     CATID = <cat-id>

           |     ,USER = *ALL / <name 1..8>

 ,NEW-CATID = <cat-id>

 ,NEW-DIRECTORY-NAME = <filename 1..54>

 ,SILENT-RUN = *NO / *YES
```

DIRECTORY-NAME = <filename 1..54>
Fully qualified path name of the directory file within which a catalog ID is to be renamed.

OLD-CATID =
Catalog ID to be renamed to the file and job variable name of the specified directory file.

OLD-CATID = *ALL
All catalog IDs are to be renamed as a single shared pubset ID.

OLD-CATID = *BY-SPECIFICATION(...)
Only the catalog ID specified below is to be renamed.

CATID = <cat-id>
Catalog ID to be renamed.

USER = *ALL
The catalog ID is to be renamed for all users.

USER = <name 1..8>
Catalog ID is to be renamed only for file and job variable names of this ID.

NEW-CATID = <cat-id>
New catalog ID specified in OLD-CATID is to be replaced.

NEW-DIRECTORY-NAME = <filename 1..54>
Fully qualified path name of the new directory file.

SILENT-RUN = *NO / *YES

If the value is *NO, a message is output if an error occurs. The run is terminated.

If the value is *YES, the run is only simulated: the input directory is read in full but no output file is created. Every error (e.g. name conflict) that is identified during execution is logged.

3.12.3.5 REORGANIZE-DIRECTORY Reorganizing repository files

This statement enables you to reorganize directory and repository files. Prior to a reorganization, obsolete save versions/files should be removed (ARCHIVE: PURGE resp. HSMS: MODIFY-ARCHIVE SAVE-FILE=*DELETE).

This statement can be used to convert a repository file in the old format (earlier than HSMS V7.0) to the new format (HSMS V7.0 or higher). The new format offers significantly better performance for backup operation with long path names of node files.

REORGANIZE-DIRECTORY
DIRECTORY-NAME = <filename 1..54> ,**NEW-DIRECTORY-NAME** = <filename 1..54>

DIRECTORY-NAMES = <filename 1..54>
Fully qualified path name of of the directory or repository file which is to be reorganized.

NEW-DIRECTORY-NAME = <filename 1..54>
Fully qualified path name of the new directory or repository file. The name must be different from the one specified in the DIRECTORY-NAME operand.

The directory specified should not already exist or should at least be empty.

After a directory file has been successfully reorganized, the following message is issued:

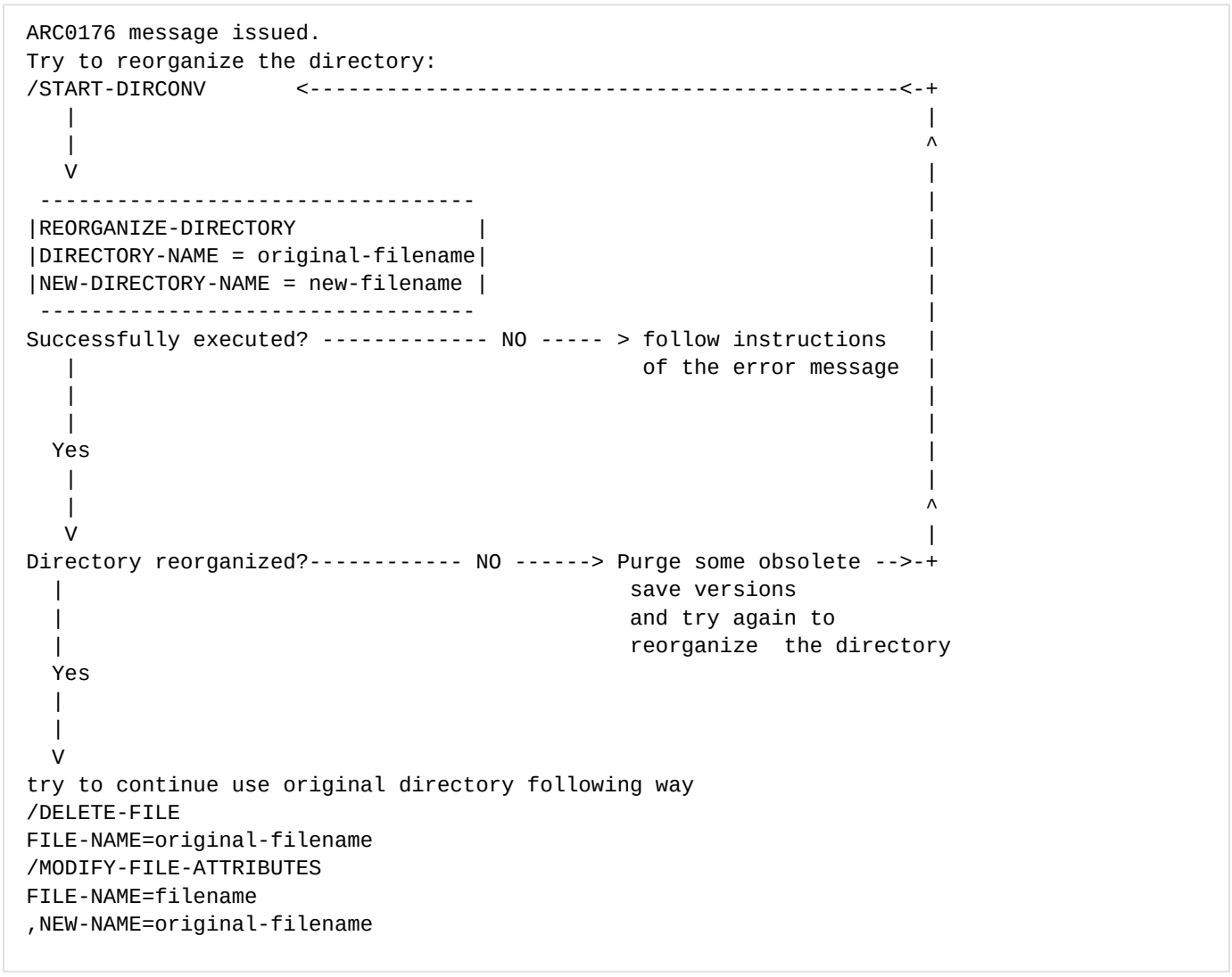
```
THE DIRECTORY HAS BEEN REORGANIZED
```

If it was not possible to perform reorganization, e.g. because the directory file had already been reorganized or no empty data records were available, the following message is issued:

```
THE DIRECTORY IS ALREADY REORGANIZED
```

No messages are issued when repository files are reorganized.

Example.


ARC0176 message issued.
Try to reorganize the directory:
/START-DIRCONV
/REORGANIZE-DIRECTORY
/DIRECTORY-NAME = original-filename
/NEW-DIRECTORY-NAME = new-filename
Successfully executed?
NO
follow instructions
of the error message
Yes
Directory reorganized?
NO
Purge some obsolete
save versions
and try again to
reorganize the directory
Yes
try to continue use original directory following way
/DELETE-FILE
FILE-NAME=original-filename
/MODIFY-FILE-ATTRIBUTES
FILE-NAME=filename
,NEW-NAME=original-filename

3.12.3.6 SET-CATID Converting directory files

This statement can be used to convert a directory file which was created under CATID=*NO mode into CATID=*YES mode.

SET-CATID
DIRECTORY-NAME = <filename 1..54> ,**NEW-DIRECTORY-NAME** = <filename 1..54>

DIRECTORY-NAME = <filename 1..54>
Fully qualified path name of the directory file in which a catalog ID must be added for all file names.

NEW-DIRECTORY-NAME = <filename 1..54>
Fully qualified name of the new directory file.

3.12.3.7 SHOW-CATID Outputting catalog IDs

This statement can be used to output all catalog IDs included in the specified directory file.

SHOW-CATID
DIRECTORY-NAME = <filename 1..54>

DIRECTORY-NAME = <filename 1..54>
Fully qualified path name of the directory file.

3.12.3.8 UPDATE-VOLUME-CATALOG Updating the volume catalog

This statement can be used to create consistency between a MAREN catalog and an ARCHIVE directory file. The MAREN administrator can change the prereservation of a free volume pool with this statement.

UPDATE-VOLUME-CATALOG
DIRECTORY-NAMES = ***NONE** / list-poss(10): <filename 1..54> ,**NEW-DIRECTORY-NAME** = <filename 1..54>

DIRECTORY-NAMES
Indicates if a pool with prereserved free volumes of an old directory name must be reallocated to the new directory name.

DIRECTORY-NAMES = *NONE
No free volume pool needs to be reallocated.

DIRECTORY-NAMES = list-poss(10): <filename 1..54>
Fully qualified path names of the original directory files which are to be used to create a new directory file and to which free volume pools have been allocated. Only the MAREN administrator is permitted to enter a path name here.

NEW-DIRECTORY-NAME = <filename 1..54>
Fully qualified path name of the new directory file.

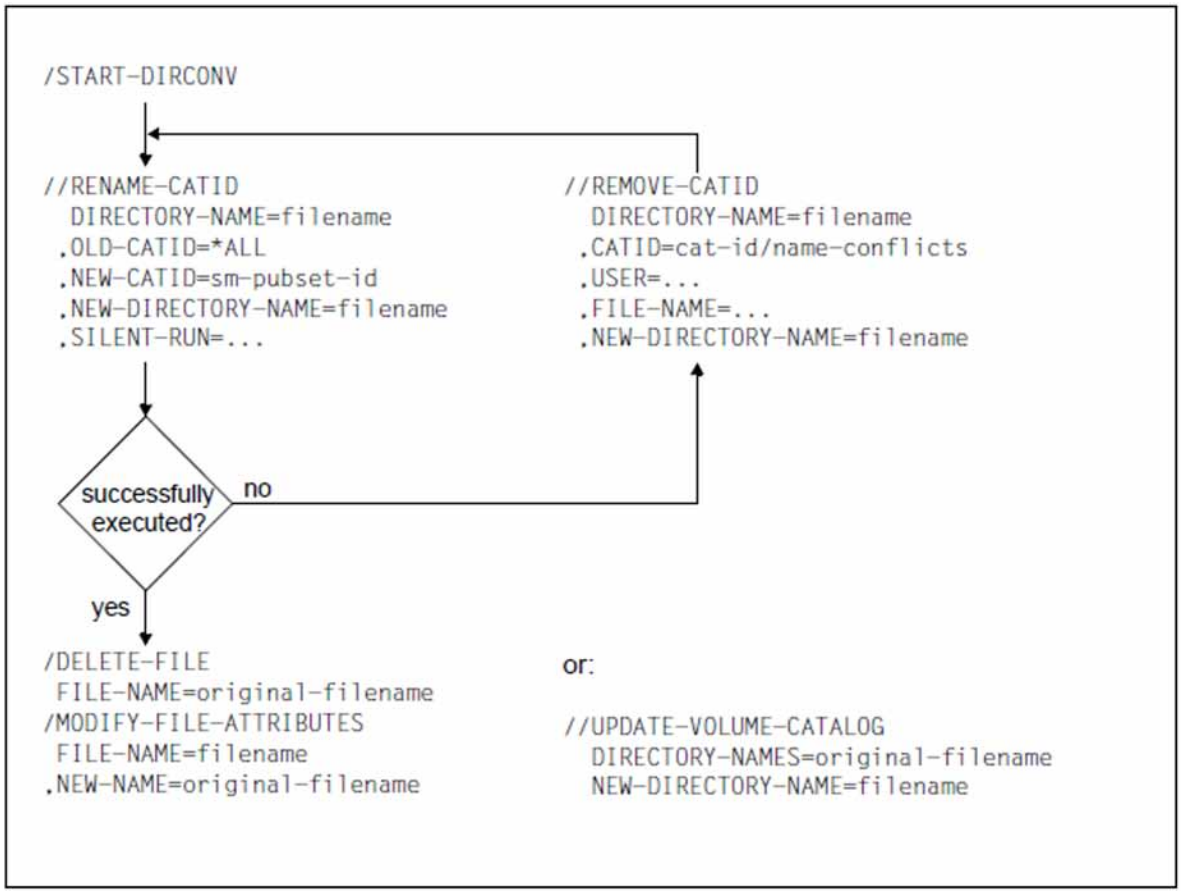
3.12.4 Usage models

This section describes three essential usage models.

3.12.4.1 Creating an SM pubset consisting of pubsets that have already been saved with a shared directory file

All catalog IDs included in the directory file must be converted into a single SM pubset ID.

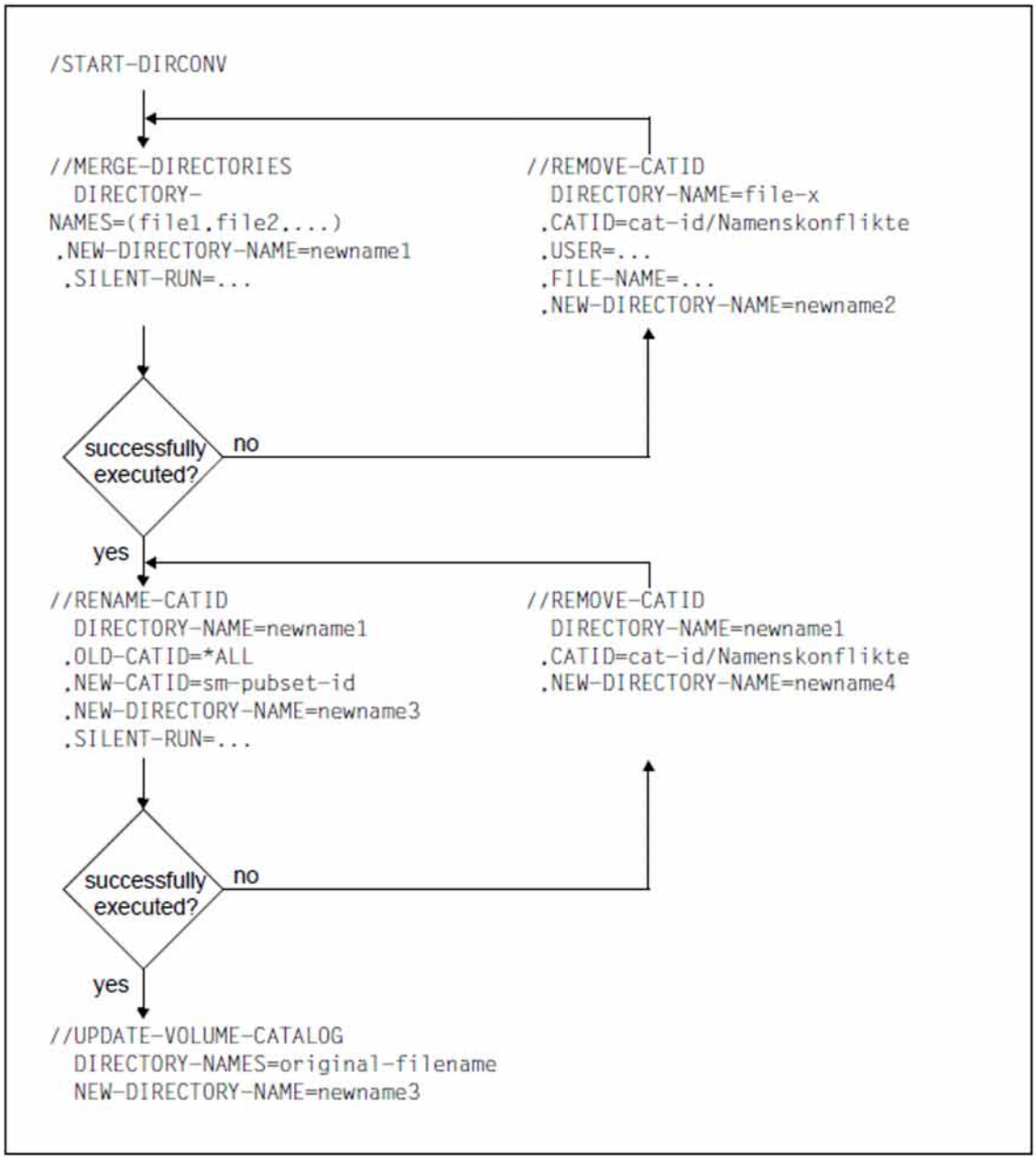
Activities to be executed:



3.12.4.2 Creating an SM pubset consisting of pubsets that have already been saved with their own directory file

The directory files must be merged into a single new directory file and all catalog IDs must be converted into a single SM pubset ID.

Activities to be executed:



3.12.4.3 Transfer of a user ID to another pubset

If a user ID is to be transferred to another pubset, all files and job variables of the user must be transferred to the target pubset. The migrated files of the user do not need to be recalled specially.

Activities to be executed:

```
/START-DIRCONV
//RENAME-CATID DIRECTORY-NAME=backup.dir (Name des SYSBACKUP-Directory)
              ,OLD-CATID=*BY-SPECIFICATION(CATID=cat-id1,USER=user-id)
              ,NEW-CATID=cat-id2
              ,NEW-DIRECTORY-NAME=new.backup.dir
//RENAME-CATID DIRECTORY-NAME=migrate.dir
              ,OLD-CATID=*BY-SPECIFICATION(CATID=cat-id1,USER=user-id)
              ,NEW-CATID=cat-id2
              ,NEW-DIRECTORY-NAME=new.migrate.dir
//END
```

Now the original directory files for SYSBACKUP (backup.dir) and SYSMIGRATE (migrate.dir) must be deleted. The directory file *new.backup.dir* must be renamed to *backup.dir*, and *new.migrate.dir* to *migrate.dir*, since these names are used by the HSMS archives SYSBACKUP and SYSMIGRATE.

The user ID files must then be reconstructed under the new catalog ID:

```
/START-HSMS
//RESTORE-FILES FILE-NAMES=:cat-id2:$user-id.
               ,MIGRATED-FILES=*CATALOG-ONLY
               ,JV-NAMES=:cat-id2:$userid.
              [,ARCHIVE-NAME=*SYSBACKUP]
//END
```

Note

The localization information in the catalog entries of the migrated files corresponds to the information contained in the SYSMIGRATE directory file. The HSMS statement REPAIR-CATALOG-BY-RESTORE is therefore not necessary.

4 HSMS functions

The basic functions (applications) of HSMS are described in the first part of this chapter:

Basic function	DMS files	Node files
Backup	X	X
Archival	X	X
Migration	X	–
Data transfer	X	–
Version backup	X	–

X The basic function is supported.

– The basic function is not supported.

The options for copying save files are also described here.

All basic functions (with the exception of migration) are described in full, i.e. both the application and management of these functions are described. Control and management of the migration function are described in chapter "Management of HSMS", since they are much more extensive for migration than for the other basic functions.

The second part of the chapter contains a description of the supporting functions relating to the basic functions. These include:

- Selection of BS2000 files, job variables and node files
- Action statements and their processing (section "Request handling")
- Allocating volumes and controlling tape access (section "Volume processing")
- Support of various disk formats and conversion during load operations (section "Handling BS2000 disks")
- HSMS screen masks and reports (section "HSMS output")
- Support of aliases (section "Aliases for BS2000 files and job variables")

In the following, UNIX files are referred to as “node files”.

The node files may be located either on the local BS2000-UFS or on a remote file system. A remote UNIX file system can be backed up directly with NFS. For the NFS backup a remote UNIX file system must be mounted on the special “HSMS” directory of the local BS2000-UFS. Wherever differences exist between the basic functions for BS2000 files/job variables on the one hand and UNIX files on the other, they are dealt with in separate sections.

HSMS also processes environments with SM pubsets. The backup, version backup and migration functions are specific to an environment, i.e. they only process objects located in that environment. By contrast, the archiving function is not restricted to a particular environment: it can process objects from different environments (SF and SM pubsets) in one run. However, if default system long-term archive is to be used for archival function, no default long-term archive must be defined in SM-environment for the SM pubsets concerned in order mixture of files from different environments can be possible.

4.1 Backup

Backup is the precautionary creation and updating of copies of the data inventory in order to permit the restoration of data lost due to operator errors such as unintentional deletion or due to hardware failures.

In HSMS V12.0A a new type of backup processing is introduced: version backup.

According to the necessity it is now possible to use any of the two types of backup processing, which have different benefits and specifics, either in parallel or alternatively:

- **backup**
  Provides backup of files from different storages and job variables; supports full, incremental or partial backup of files; automatic duplication of save files and automatic deletion of obsolete save files; statements BACKUP-FILES, BACKUP-NODE-FILES.
- **version backup**
  Ensures that a number of file versions are kept; provides only incremental backups of files; reorganization mechanism is introduced to copy still necessary file versions and to be able to remove obsolete file versions. Statements BACKUP-FILE-VERSIONS, REORGANIZE-VERSION-BACKUP, CHECK-CATALOGED-FILES.

4.1.1 General notes on saving data for both types of backup processing

BS2000 files are files which are managed and edited in BS2000. Unless an explicit distinction is made, in the case of files on Net-Storage the term "BS2000 file" refers to both the file type BS2000 and the file type node file which is also supported in BS2000 OSD/BC V10.0 and higher.

Within the context of this section, the term “files” always refers to BS2000 files.

HSMS saves data logically.

Large files (files > 32 GB) can also be saved. No particular provisions are required to do this. However, when restoring large files, bear in mind their special characteristics (see section "Restoring large files (> 32 GB)").

Encrypted files are saved in encrypted format and restored again in the original encrypted format. The associated crypto password need not be specified for saving or restoring. Other accesses to the restored file require the crypto password to be specified again, and the technical prerequisites for file encryption must be provided in the system.

When the HSMS statements BACKUP-FILES or BACKUP-FILE-VERSIONS are used in an environment with SF pubsets, they can only process SF pubsets; SM pubsets cannot be processed.
When the HSMS statements BACKUP-FILES or BACKUP-FILE-VERSIONS are used in an environment with SM pubsets, they are restricted to the environment in which they are called, i.e. they relate to precisely one SM pubset.

By default it is defined (see //SHOW-HSMS-PARAMETERS, MIGRATION-CONTROL, BACKUP-MANDATORY= YES) that, for security reasons, a file cannot be migrated until the current version of the file has been written to a save file at least once.
In the event of backup, it is possible to save either just the catalog entries of migrated files (save type MIGF) or both the catalog entries and the data contained in the files. With version backup, both the catalog entries and the data are saved. For further details, see section "Data backup and migrated files".

How a save file can be extended depends on its structure and its storage location:

- In the case of backup archives with single SVID structure or save files on disk, specifying the operand SAVE-FILE=*CONTINUE permits a disjunct set of files to be added to a save file written to an archive.
- In the case of archives with several SVID structure, the save file can be updated with a new save version (irrespective of whether the same files are saved). The file must be distinguished by their save version.

4.1.2 Backup with BACKUP-FILES

Within this section the specific features of the first type of backup are described. In other words, this section describes how files and job variables are backed up using the HSMS statement BACKUP-FILES.

4.1.2.1 Saving BS2000 files and job variables

HSMS provides for system backups and application backups and in both cases offers the options of full backup, incremental backup and partial backup. HSMS can additionally be used to reorganize the data residing on disk storages.

This type of backup uses backup archives either user archives or system backup archives.

System backups are the prerogative of the HSMS administrator.

Backups may be performed by any user.

A nonprivileged user can back up files to any other user archive if:

- the archive is public (i.e. was installed with ACCESS=*WRITE)
- the files belong to the archive owner and the job submitter is the co-owner of the files.
- the foreign archive and its directories have co-ownership. These can be used as though they were the user's own. Only archive management, i.e. changing archive properties or deleting obsolete save files, is excluded.

Nonprivileged users can restore files and job variables belonging to their own user ID as well as files and job variables of which they are co-owners:

- from his own backup archives or from foreign archives/directories, which he has access to due to co-ownership.
- from a public backup archive

The measures to be taken by the HSMS administrator in connection with data backup are described in section "Data backup and migrated files".

The archive owner and the HSMS administrator can use the SAVE-FILE operand of the BACKUP-FILES statement to define whether

- the files to be saved are to be written to the archive's standard save file (*STD), if permitted by the archive definition
- a new save file is to be created (*NEW)
- an existing save file is to be updated (*CONTINUE).

Files and job variables are backed up using the HSMS statement BACKUP-FILES.

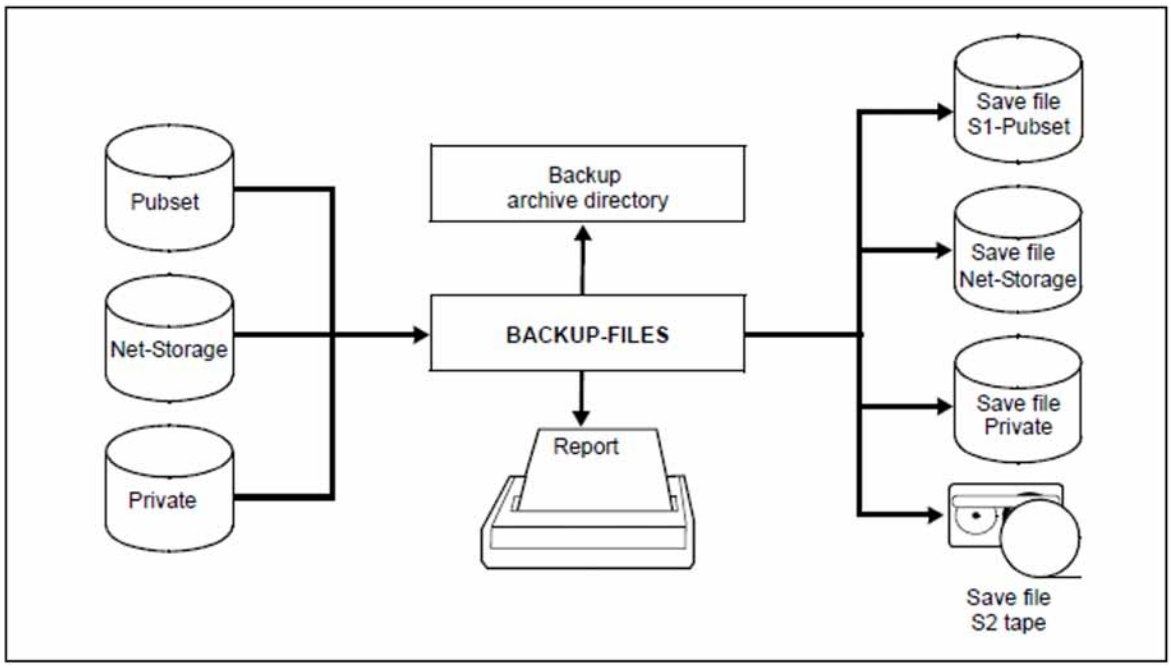


Figure 8: Backup using BACKUP-FILES

The files which are to be stored, which must reside on pubsets, Net-Storage or private disk, can be selected according to

- the volume on which they reside (SUPPORT)
- the file type for Net-Storage files (FILE-TYPE=*BS2000 / *NODE-FILE)
- the backup class assigned to them
  (Operand MAXIMUM-BACKUP-CLASS).
- a list of file names (see section "Selection of BS2000 files, job variables and node files").

System backup archive

Backed up data is managed in backup archives. The HSMS administrator creates system backup archives for system backups of SF pubsets and SM pubsets. They are normally addressed via the symbolic name SYSBACKUP. It is the default backup archive to which the HSMS statement RESTORE-FILES refers unless another archive is specified.

The system backup archive must be created as a public archive with read access (operand USER-ACCESS=*ALL-USERS/ACCESS=*READ) in order to enable nonprivileged users to restore data from this archive.

Full and incremental backup

A full backup saves all selected files completely (except for migrated files), regardless of whether or not they have been changed since the last backup.

Backups performed by the HSMS administrator include data of online migrated files if the appropriate backup method has been selected. The backup includes the catalog entries from the catalog and the data from the migration level. It is also possible to limit the backup of migrated files to their catalog entries.

For files on Net-Storage you can also choose whether the entire file (default) or only the catalog entry is to be backed up (SAVE-OPTIONS= *PAR(SAVE-NET-STOR-DATA=*YES/NO) operand).

An incremental backup does not save all specified files. If HSMS detects that the appropriate archive already contains the current contents of a file, the file is not saved again. Instead it is registered in the directory as Cataloged-Not-Saved (CNS). As a result, only those files whose contents were modified or newly created since the last backup are saved. Incremental backups can therefore result in a considerable reduction of the time and storage space required for a system backup. They are carried out automatically by HSMS unless otherwise specified. Node files on Net-Storage are always fully backed up – also in the case of incremental backups.

If a file is entered as Cataloged-Not-Saved (CNS) in the archive directory, the retention period of the last full backup of this file is checked. If the expiration date of the last full backup is older than the expiration date of the current incremental backup, the retention period of the last full backup is updated. Consequently, the last full backup is assigned the same expiration date as the current incremental backup, i.e. the date of the full backup does not expire before that of the associated incremental backup. As a result, the full backup cannot be deleted before all the associated incremental backups unless the user forces deletion by specifying FORCE-DELETE in the MODIFY-ARCHIVE statement.

If only the catalogue entries of the migrated files have been saved with the last full backup (SAVE-DATA=*STD /*S0)), the data of the migrated files also have to be saved (SAVE-DATA=*S1-S0/*S2-S1-S0), so that in case of a crash at migration level, the data can be recovered. Details on S1/S2 crash troubleshooting can be found here: section "Using RESTORE on migrated files".

Full backup offline

A full backup may be initiated either for storage level S0 (FROM=*S0), for previous backups stored in an archive (FROM=*ONLY-LATEST-BACKUPS) or for both (FROM=*LATEST-BACKUPS-OR-S0).

In the latter case (FROM=*LATEST-BACKUPS-OR-S0), the backup is produced from the copies in the save file of all files currently stored in the archive rather than from the contents of the files at storage level S0. As a result, time requirements for a full backup at storage level S0 are similar to those for an incremental backup, since the procedure is as follows: first all modified files of storage level S0 are saved. Once this step has been completed, storage level S0 is again available. The backup is then continued, with files being copied from tape to tape.

In a full backup with FROM=*ONLY-LATEST-BACKUPS, the complete backup is performed offline. No preparatory online steps are necessary at storage level S0 because the last incremental backup is taken as a reference. All files which are entered in this incremental backup (fully or as CNS), are backed up from their last save and combined into a single new full backup. But if this last incremental backup contains files which were only partially saved (partial backup), these files are skipped because they cannot be restored offline.

A full backup of migrated files with FROM=*ONLY-LATEST-BACKUPS is not permitted. Migrated files which were marked as MIGF in the last backup remain as MIGFs in a pure full backup offline.

To be able to guarantee a complete offline backup which includes storage levels S0, S1 and S2, the previous last backup run must have been started with the specification SAVE-DATA=*S2-S1-S0 and must have been performed without partial backup.

Special aspects

A full backup with the operand LATEST-BACKUPS-OR-S0 proceeds in exactly the same way as any other full backup, except for the handling of the metadata (catalog entry) of unmodified files. Files that have not been changed are not taken from the S0 storage level, instead they are taken from the last full or incremental backup (including their metadata).

S0 catalogued files are the basis of the file volume to be saved, therefore, no data on S0 should be deleted; this also applies to offline backups.

A full backup of S0 or of the most recent backup in the archive always has to start a new save file; it is not possible to continue an existing save file.

Up to 255 partial backups of a file can be performed between any two full backups of this file. The 256th backup is automatically executed as a full backup for this file even if otherwise specified in the HSMS statement.

The type of backup is controlled via the SELECT-FILES operand of the BACKUP-FILES statement. Specifying SELECT-FILES=*MODIFIED-FILES results in an incremental backup, while SELECT-FILES=*ALL-FILES causes a full backup to be carried out.

The type of full backup can be specified via the FROM operand. The default value *S0 causes the files of storage level S0 to be saved. If the value *LATEST-BACKUPS-OR-S0 is specified, the backup of CNS files contained in the archive is copied.

Making a full backup by means of incremental backups works like this:

- The system catalog is checked to determine the status of the files.
- All modified files are backed up from storage level S0.
- All unmodified files are backed up from tape.

Partial backup

The scope of an incremental backup can be further reduced by what is called a partial backup.

Partial backup is not possible for UPAM files with BLOCK-CONTROL-INFO=*NO. Except for PLAM libraries. These libraries can be partially backed up because the PLAM access method replaces the PAM key with its own mechanism for marking modified sections in libraries.

As in the case of incremental backup, files which have not been modified since the last backup are not saved. If files have been modified, HSMS checks which parts of a file (2-Kbyte blocks, PAM pages) have been changed since the last full backup (not partial backup). Only these parts are saved.
As a result, the last partial backup and the last full backup are required to restore a partially backed up file.

If a file is entered as “Partially Saved” in the archive directory, the retention period of the last full backup for this file is checked. If the expiration date of the last full backup is older than the expiration date of the current partial backup, the retention period of the last full backup is updated. Consequently, the last full backup is assigned the same expiration date as the current partial backup, i.e. the date of the full backup does not expire before that of the associated partial backups. As a result, the full backup cannot be deleted before all the associated partial backups unless the user forces deletion by specifying FORCE-DELETE in the MODIFY-ARCHIVE statement.

Partial backups are requested by means of the statement

```
//BACKUP-FILES SELECT-FILES=*MODIFIED-FILES -
// (PARTIAL-FILE-SAVE=*LARGE-FILES/*YES)
```

A partial backup partially saves either all files with LARGE=*YES or SAVED-PAGES=*MODIFIED-PAGES in their catalog entry or files of a specific size irrespective of the catalog entry. In the latter case, the size is specified by means of *YES(MINIMUM-SIZE=<integer>).

Note

File generation groups are always backed up completely to one save file with multiple backup versions. If a file generation or a file generation index is specified, a full backup of the index and of all file generations is performed.

4.1.3 Version backup

In HSMS V12.0A a new type of backup processing is introduced: version backup.

This provides the following features:

1. Backup and keep multiple file versions in the version backup archive for files cataloged in the respective pubset. The number of versions to keep is determined by the user in the file properties.
2. A mechanism to ensure that a deleted file (possibly accidentally deleted) will be kept in the archive long enough in the previously backed up versions so that the user can take action to reconstruct the data.

Number of Backup Versions

The new file attribute NUM-OF-BACK-VERS (0..32) for the number of backup versions is introduced within BS2000 OSD V11.0B. This defines how many copies of a file are to be kept in the version backup archive at the same time.

NUM-OF-BACKUP-VERS = 0 means that the file will not participate in version backup. In the SM environment, the number of backup versions can also be controlled within management classes. The value in the management class predominates a deviating value in the catalog entry.

A new type of archive – the **version backup archive** – is introduced for the purpose of version backups.

There is one version backup archive per pubset where the desired number of file versions are saved - to always be sure that a consistent number of file versions are kept.

Thus, there are no user version backup archives. Version backups can be done only using system version backup archives. In both the SF and SM environments, they are defined Pubset-specifically and addressed by the symbolic name SYSVERSION. However, there is no global SYSVERSION archive for the SF environment.

The system version backup archive must be created as a public archive with write access (operand USER-ACCESS=*ALL-USERS/ACCESS=*WRITE) in order to enable nonprivileged users to save and restore data within this archive. Alternatively with ACCESS=*READ only the HSMS administrator may perform the save runs but the nonprivileged users are able to restore data from the archive.

The HSMS statement BACKUP-FILE-VERSIONS is used to save files in a version backup archive.

A user can specify the storage media from and to which the data is to be written.

Version Backup is not possible for:

- files on private disks
- FGGs
- temporary files
- catalog entries of tape files
- JVs

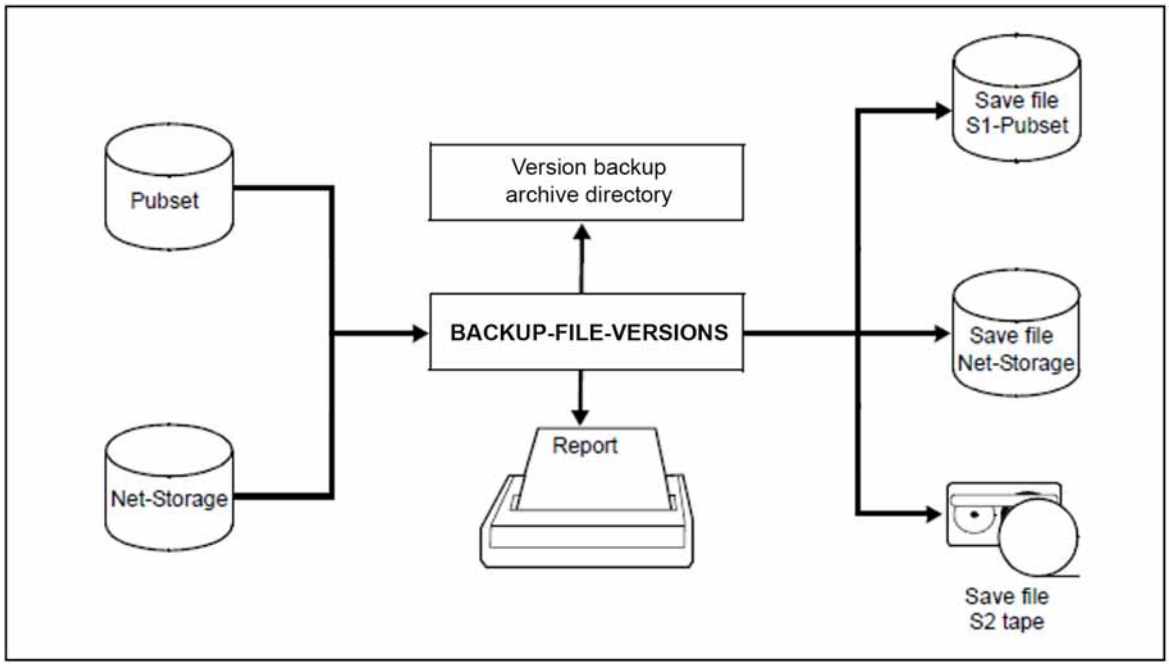


Figure: Version backup using BACKUP-FILE-VERSIONS

Version backup functionality available only as of BS2000 OSD/BC V11.0B, when a new file attribute NUMBER-OF-BACKUP-VERSIONS has been added, and in *HSMS-V10-COMPATIBLE mode. In an inhomogeneous BS2000/HSMS environment command will not be transferred to a master (backup server) running BS2000 < 11.0B or HSMS < 12.0A. The command request will be aborted on the host, which starts the request with an appropriate message.

4.1.3.1 Incremental version backups

Only incremental backups are possible within version backup: only modified files are saved, not modified files are skipped, it means no CNS entries are provided.

During the save run the file attribute NUM-OF-BACKUP-VERS is stored in the respective file record of the archive directory. From this value HSMS can decide how many file versions need to be kept in the archive or the other way round which old file versions can be removed during a reorganization run. (Reorganization is described in the next chapter).

However, in case not modified files have changed the number of backup version attribute or the value in the management class, to which the files are assigned, has been modified, the corresponding value in the version backup directory is updated correspondingly. This update is done during a save run for the files specified in the file names operand, but can be done also with the statement CHECK-CATALOGED-FILES.

For migrated files both the catalog entries and the data are always saved.

The backup class may be used as a criterion for deciding whether a file is to be backed up.

4.1.3.2 Reorganization of version backups

Reorganization is used to save space on the respective backup storage level (S1 or S2). There are obsolete file versions (file versions which should not be kept any longer according to file attribute number of backup versions (NUM-OF-BACKUP-VERS), and there are files that have been deleted for a longer time from the pubsets file catalog and are now ready to be removed from the version backup archive. The **reorganization** of version backup archives is provided via the statement REORGANIZE-VERSION-BACKUP. The steps on reorganizing a version backup archive are the following:

1. Perform the HSMS statement CHECK-CATALOGED-FILES:
   CHECK-CATALOGED-FILES has two purposes: it updates the number of backup versions (NUM-OF-BACKUP-VERS) in the files records of the archive directory from the catalog entries of the TSOSCAT. In addition, CHECK-CATALOGED-FILES checks if the files residing in the archive directory still exist on the corresponding pubset. If a file was deleted from the pubset in the meantime, the file is marked as deleted and the current date is inserted as a deletion-date in the file record in the directory. It is recommended that the HSMS administrator create reports during the check runs to be able to inform the users about deleted files. Additionally, the user can check this situation by using HSMS statement SHOW-ARCHIVE. With the SHOW-ARCHIVE statement the user can see which files will be lost during the next reorganization: SHOW-ARCHIVE provides information on which files are deleted from S0, the deletion date and which are additionally marked for deletion from the archive. If the file was deleted unintentionally from the pubset, the user may restore the file.
   If a CHECK-CATALOGED-FILES finds that a file record has a deletion date but exists again in the TSOSCAT of the corresponding pubset, the deletion date is erased from the file record.
2. MODIFY-ARCHIVE FILE = *MARK-FOR-DELETION(…)
   In order for a file to be deleted from the archive during reorganization, it must first be marked for deletion. This flag is set with MODIFY-ARCHIVE FILE = *MARK-FOR-DELETION(...) and is only possible after a certain period of time (SECURE-PERIOD) has elapsed since the file was deleted from the pubset. The SECURE-PERIOD is a property of the version backup archive. It is 180 days by default. The files are now “marked for deletion”, but the deletion is still not performed. In case the HSMS administrator wants to remove files from the archive although they still exist on the pubset, he must use the force option to mark the files for deletion: MODIFY-ARCHIVE FILE = *FORCE-DELETION(...)
3. After that, a reorganization of the archive needs to be performed by REORGANIZE-VERSION-BACKUP statement. Now obsolete files and files that are marked for deletion are removed from the archive.

Hint: In case CHECK-CATALOGED-FILES had been done previously and/or some files had been marked for deletion with MODIFY-ARCHIVE FILE=*MARK-FOR-DELETION(…)/*FORCE-DELETION(...) and no reorganization followed, during BACKUP-FILE-VERSIONS or CHECK-CATALOGED-FILES the deletion date and the mark for deletion of specified existing (again) files are reset. Corresponding hints are given within the report.

> The user can change the value of the number of backup versions in the file catalog at any time. However, the change will not be transferred to the archive directory until the next backup or the next call of the CHECK-CATALOGED-FILES statement and only if the file is specified in the FILE-NAMES operand. This means that the updating of information functions or reorganization runs takes effect from this moment on.

4.1.4 Backup using the function "Concurrent Copy"

The Concurrent Copy (CCOPY) function offers a backup option which permits a consistent backup (incl. version backup) of a specified set of files to be produced while allowing any editing of the files to be continued. The computer center may benefit from this by saving a considerable amount of time compared with the normal backup methods.

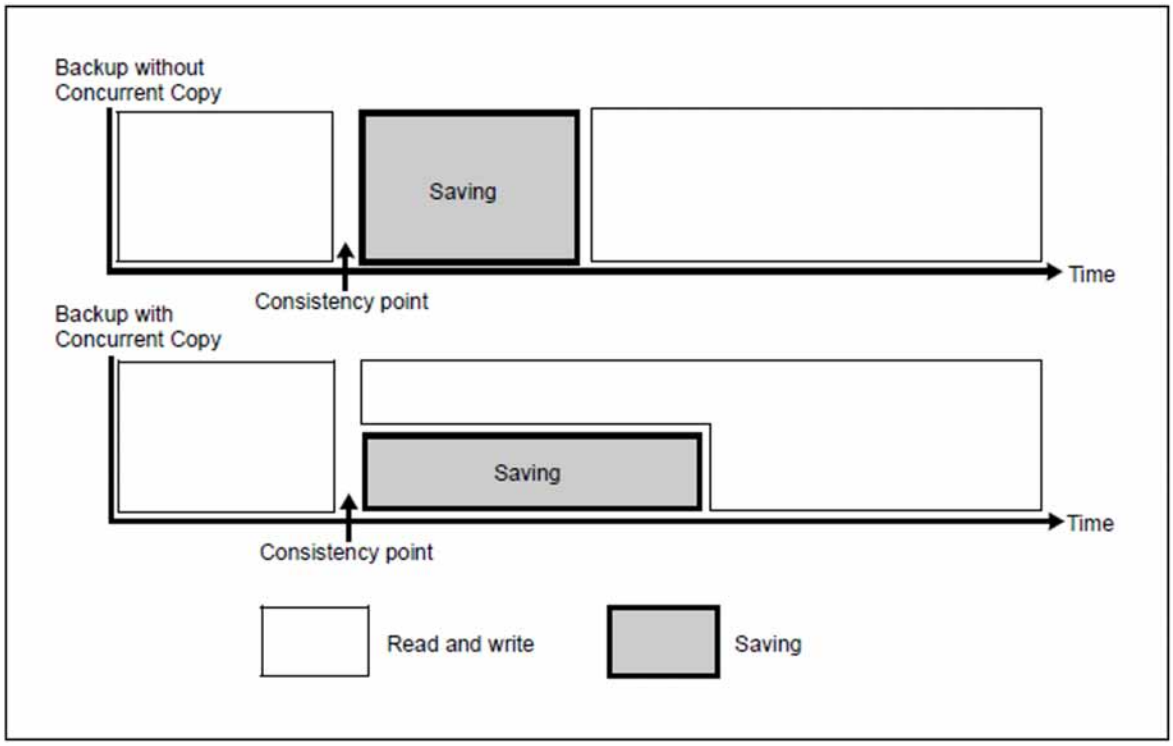


Figure 9: Backup processes with and without Concurrent Copy

The CCOPY function is intended in particular for applications which have to remain open for extended periods but during whose execution it is nevertheless necessary to back up files. However, it requires more resources than a normal backup. The CCOPY function is activated by means of the CONCURRENT-COPY operand in the BACKUP-FILES or BACKUP-FILE-VERSIONS statements.

In order to request a backup with the CCOPY function it is necessary, when using a work file (WORK-FILE-NAME=*STD or <filename>), to define a set of files which are to be backed up in this way. This set of files may contain, for example, all the files required by the application. When the backup job is created, these files must be open at least for read access to ensure that they have a consistent status (exceptions here are online backup with SHC-OSD on "Backups with SHC-OSD" and the transfer of a Snapset backup on "Transfer of files and job variables from Snapsets to a backup archive"). The backup job must be created using //BACKUP-FILES ..., CONCURRENT-COPY=*YES; or in case of version backup, by //BACKUP-FILE-VERSIONS ..., CONCURRENT-COPY=*YES.

Once initialization is complete, the HSMS/system administrator receives messages indicating the results of this first step. If the job variable service is active in the system, the administrator can use a job variable to query whether a problem has occurred. If the backup cannot be initialized for one or more files, HSMS informs the administrator of the files which could not be processed.

The cause may be that a file was locked. The administrator can then remedy the error and repeat the HSMS statement BACKUP-FILES or BACKUP-FILE-VERSIONS in case of version backup.

If an error occurs during backup with CONCURRENT-COPY=*YES or if any files cannot be saved, the backup run is still continued for the remaining files, saving as many of the specified files as possible.
Files which could not be saved can be included in a later backup. Any files that could not be saved can be included in a subsequent backup with or without the CCOPY function. However, consistency with the files saved in the initial backup is not guaranteed in this case.

Saving with CONCURRENT-COPY=*YES stores the file. Otherwise the files will be saved one after the other.

If one of the required files cannot be found or is open, the backup is continued as if it had been started without CONCURRENT-COPY=*YES.

When saving with CONCURRENT-COPY=*YES you cannot carry out a write buffer with DAB.

Irrespective of any backup server definition, backups with Workfile are always sent to and processed on the master in the SPVS network. Backups of split-off mirrors (*BY-ADDITIONAL-UNIT) are always processed locally.

Back up with work file

The basic mechanism used as default for determining a consistency point when backing up data is to write so-called before-image files: for the duration of a CCOPY backup, whenever a block in a file forming part of the backup set is modified for the first time, then, before this happens, a copy of the as yet unmodified block is written to a work file. The CCOPY backup uses internal tables, which are updated whenever a before-image file is read, whether a block is to be read from the original file or the work file.

The work file for the before-image files can either be created by default by HSMS or be specified explicitly in the HSMS statement BACKUP-FILES or BACKUP-FILE-VERSIONS:

```
//BACKUP-FILES ..., CONCURRENT-COPY=*YES(WORK-FILE-NAME= -
             <filename 1..54 without-gen-vers>)
```

or

```
//BACKUP-FILE-VERSIONS ..., CONCURRENT-COPY=*YES(WORK-FILE-NAME= -
             <filename 1..54 without-gen-vers>)
```

If file sets are backed up on shared pubsets, then it is necessary to ensure that the work file is accessible on the master host or backup server of the shared pubset if this work file is explicitly specified. It is essential that the backup (but not the creation of the backup job to HSMS) is performed on the master host or backup server.

Performance

The writing of before-image files to the work file, which requires one read and one write operation per (large) block, together with the internal administration of the before-image files, places an increased demand on operating resources: I/O, CPU time and memory space for the work file. The increased CPU load and the additional I/O operations have a negative effect on performance. They are dependent on the location and scope of the changes that are performed to the data in question in parallel to the backup. If the system load is low, the effects are correspondingly small.

Backups with SHC-OSD

With SHC-OSD the mirroring functions of the disk storage systems are available in BS2000 as a further basic mechanism for CCOPY backups. When these functions are used, disks of a pubset within a disk storage system are mirrored on additional disks (additional units or clone units, depending on the mirroring function). If necessary, these additional mirror disks can be split from the original disks and processed as a pubset copy independently of the original pubset. The splitting and rejoining of the disk pairs is controlled by SHC-OSD. This function is particularly useful for backups. It is integrated in HSMS via the CCOPY subsystem. For details on the BS2000 host component for storage systems, refer to the “SHC-OSD” manual [23].
If remote mirroring in a second disk storage system is used, the splittable disks can also be used in the remote disk storage system.
For HSMS backups which use the mirroring functions of a disk storage system with SHC-OSD, mirroring must be performed on the composite pubset, i.e. for all the volumes of a pubset. The mirror disk pairs are then split on the initialization of the backup job for all the volumes of the pubsets involved. Pubset I/Os are stopped for the duration of splitting. This ensures that the following conditions are fulfilled:

- The data of the file set that is to be backed up is consistent. Files opened in write mode do not result in the backup being aborted or can be saved as in a backup without CCOPY using the SAVE-ONLINE-FILES=*YES option.
- The pubset metadata on the split-off mirrors (additional units or clone units) is consistent in that they permit the reconstruction of the pubset at the time of splitting.

This ensures that the consistent data status as at the start of the backup is available during the entire backup operation. Once the backup has been successfully concluded, the mirroring is automatically recommenced. This can be prevented by specifying DISCARD-COPY=*NO in the BACKUP-FILES or BACKUP-FILE-VERSIONS statements. Error handling is discussed below.

Backing up open files

When pubset copies are backed up under CCOPY, files that are still open can also be backed up. This function was designed with databases in mind (e.g. SESAM, UDS, ORACLE):

If databases are to remain permanently accessible, they cannot be run down and closed just to perform a backup. As a result, databases are usually backed up while they are still open. More detailed information on backup can be found in the manuals on the various database management systems.

The variation of online backup means that a database backup can be carried out much quicker than before, because the length of time that the database files are locked, is now much shorter, and this results in part in smaller LOG files subsequently required to reconstruct a consistent state.

Renaming mirror disks

When the disks of the pubset copy are split off, they are renamed in accordance with the following rules:

- In the case of volumes which contain a period, the period is replaced by a colon, e.g. ABC.05 –> ABC:05
- Volumes with names starting with the prefix PUB receive the prefix P:B, e.g.

This renaming process is necessary so that, following splitting, the volumes on the original disks and the mirror disks do not have the same names. Only in this way is it possible to address the disks of the pubset copy via a unique name.

Creating a pubset copy

When you initiate the backup, you can specify CONCURRENT-COPY=*YES(WORK-FILE-NAME=*BY-ADDITIONAL-UNIT) in the HSMS statements BACKUP-FILES or BACKUP-FILE-VERSIONS to indicate that you want to use a pubset copy. However, it should be noted that when you perform this type of backup job, the mirror disks of the pubset copy are not available for other backups. For this reason, it is necessary to coordinate backup jobs which use the same mirror disks.

Only a pubset copy saved with //BACKUP-FILES can be used to reconstruct the complete SF or SM pubset from the mirrored data volumes of the original pubset and the backed up files.

If you also wish to save open files, you must also specify SAVE-ONLINE-FILES=*YES. However, this only backs up the files that are open, for which online backup has been specified using the OPNBACK operand of the CATAL macro call (see the “DMS macros” manual [22]).

In addition, you can specify WRITE-CHECKPOINTS=*YES in order to ensure that you can restart or reset mirror disks (see below). This is recommended since it requires almost no extra effort. At the same time, the availability of the data stock in its status at the start of the backup is guaranteed for the entire duration of the backup.

Authorization for backup using SHC-OSD with assigned customer privileges

Backups based on SHC-OSD mirroring functions to a large degree demand operating resources which are not available without limit or infinitely extensible but, instead, whose use has to be precisely planned. Consequently, this type of job is reserved for systems support staff (TSOS privilege) and HSMS administrators (HSMSADM privilege). If necessary, it is also possible to assign a customer privilege in order to allow application administrators who do not possess either of these privileges to perform a backup using SHC-OSD mirroring functions. However, control of customer privileges requires the use of the SECOS software product (see the “SECOS” manual [16]). The following steps are necessary:

1. You define one or more customer privileges which provide authorization for this backup. These customer privileges are reported to the CCOPY subsystem in a parameter file on subsystem start.
   By default, the CCOPY information file is used for this purpose:
   `SYSSSI.CCOPY.`*<subsys-vers>*
   Alternatively, you can use START SUBSYSTEM command with
   PARAMETER=‘<filename>‘ to specify a parameter file with a name of your choice; this specification has priority.

   The parameter file is a SAM file with variable record length and contains precisely one record. This record specifies the customer privileges for backups with SHC-OSD mirroring functions in the form of the string CU[STOMER]-PRIV[ILIGES]=(0,1,2,...,8). If only a single customer privilege is specified then the parentheses can be omitted. The value 0 means that no customer privilege is to be checked. This value is set by default in the CCOPY information file.
2. You use the corresponding SECOS function to assign one or more user IDs to the customer privileges for which CCOPY backups with SHC-OSD mirroring functions have been enabled as specified under point 1 (see the “SECOS” manual [16]).

Performance

CCOPY backups with SHC-OSD mirroring functions barely differ from backups without CCOPY in terms of performance: the data to be backed up is read from the split-off mirror disks, while the files that continue to be processed during the backup are located on the original disks. There is therefore no interference between the backup and the application.

Because the catalog accesses are not executed from the disks of the pubset that are running but locally from the split-off disks, it is not practical in shared-pubset mode to send the job to the pubset master to enhance performance. CCOPY backups with SHC-OSD mirroring functions are therefore executed locally on the system on which the backup call was issued.

Error handling

In the case of CCOPY backups with SHC-OSD mirroring functions, the data that is to be backed up and all the metadata for the pubsets involved is available on the pubset copies at the consistency point status at the moment the backup was started. In this way, extensive recovery measures can be performed if, as is recommended, the backup has been defined as restartable. This can be done in the HSMS statements BACKUP-FILES and BACKUP-FILE-VERSIONS or with WRITE-CHECKPOINTS=*YES during the definition of the archive.

When SHC-OSD mirroring functions are used, this specification also makes it possible to perform application-specific recovery measures. In contrast, if you specify WRITE-CHECKPOINTS=*NO then the procedure is the same as for CCOPY backups with before-image files: no restart mechanisms are provided for by HSMS/CCOPY.

Below, we present certain error scenarios and the associated restart possibilities if WRITE-CHECKPOINTS=*YES is specified:

1. Error during backup

   Here the same applies as for backups without CCOPY: if the HSMS job has the status INTERRUPTED then it can be continued using the HSMS statement RESTART-REQUESTS. For more detailed information, see section "Restarting requests".

2. System failure

   - Backup in progress and restartable application
     Once the system is available again, a warm start of the application is performed. The backup can be continued using the HSMS statement RESTART-REQUESTS. For more detailed information, see section "Restarting requests".
   - Backup in progress and non-restartable application
     Once the system is available again, the affected pubset must be reset to its consistent status at the start of the backup in order to make it possible to repeat the interrupted operation. To do this the following steps are necessary:
     - The pubset's original disks must be synchronized with the mirror disks. This is done using the following command. Here the pubset may not be imported:

       ```
       /RESUME-MULTI-MIRRORING UNIT=*BY-PUBSET(PUBSET=<catid 1..4>), -
       /                       RESTORE=*TO-ORIGINAL,UNLOCK-ADD-MIRROR=*YES
       ```

       or

       ```
       /RESTART-CLONE-SESSION UNIT=*BY-PUBSET(PUBSET=<catid 1..4>), -
       /                      RESTORED-SESSION=*ACCEPT
       ```

       (see the "SHC-OSD" manual [21]). This resets the data to its status at the last consistency point.
     - Next, the pubset volumes which possess the archive numbers of the mirror disks must be renamed to their original names using the PVSREN utility so that the archive numbers are again compatible with the pubset syntax. This is done using the PVSREN statement:

       ```
       //RESTORE-LABELS-OF-PUBSET CATID=<catid 1..4>
       ```

       (see the "Utility Routines" manual [3])
     - The reset pubset must be imported.
     - The interrupted operation must be repeated. It may also first be necessary to reissue the backup job.

Restrictions

The following restrictions apply to backups with CCOPY:

- The WRITE-CHECKPOINTS and SAVE-ONLINE-FILES functions are only available for backups of pubset copies.
- Private disks are not supported.
- A CCOPY backup on a shared pubset with work file (WORK-FILE-NAME=*STD or <filename>) is always processed on the master. A CCOPY backup of pubset copies is executed locally on the side where the backup call was issued.
- File migration is not supported during a CCOPY session.
- The memory space occupied by a file which belongs to the file set that is to be backed up may not be reduced using the following BS2000 command during the entire duration of the backup:

```
/MODIFY-FILE-ATTRIBUTES ...,SUPPORT=*PUBLIC-DISK(SPACE=*RELEASE(...))
```

- A file may, at any one time, form part of only one CCOPY backup. If a job for further CCOPY backup involving this file is requested then this job is rejected.
- CCOPY optimizes the catalog accesses when initiating the backup of a pubset copy.

Transfer of files and job variables from Snapsets to a backup archive

In BS2000 pubset copies can be generated on the basis of Snap disks (so-called Snapsets) from which files or job variables can then be restored directly in the original pubset with the DMS command. Since the total number of these pubsets is limited for a pubset by the disk subsystem, the generation and deletion of Snapsets must take place in the same rythmn. Before a Snapset is deleted the files and job variables saved on it can be transferred to a backup archive using the HSMS statement BACKUP-FILES. The catalog ID of the pubset and the identifier of the Snapset to be saved are specified here:

```
//BACKUP-FILES ..., CONCURRENT-COPY=*YES(WORK-FILE-NAME= -
             *FROM-SNAPSET(PUBSET-ID=<catid>,SNAPSET-ID=<snap-id>)
```

In contrast to the backup of pubset copies, when saving Snapsets you must bear in mind that the files and job variables do not match the current state at the time BACKUP-FILES is processed but reflect a backup status at the time the Snapset was generated. Consequently a save version generated in this way is assigned the date of Snapset generation in the backup archives.

If newer save versions already exist in the backup archive a new save file must be created in order to retain the uniqueness of the save versions within a save file. An update of the save file is rejected. The transfer of files or job variables from Snapsets should not be combined with direct backups in the same backup archive.

Data from Snapsets cannot be transferred to a version backup archive.

4.1.5 Save options

The SAVE-OPTIONS operand can be used to set options for the current backup run.

Saving UDS databases online (SAVE-ONLINE-FILES)

In HSMS, SAVE-ONLINE-FILES=*YES can be specified to save files which are open during the backup operation. However, not all files are saved, but rather only opened files for which online backup has been expressly specified, using the OPNBACK operand in the CATAL macro (see the “DMS macros” manual [22]).

This function is intended only for database files, since this type of backup can result in file inconsistency if used with normal files.
Online backup of UDS databases can be used for UDS-SQL V1.2 and higher. This requires that an after-image file be used (for additional conditions. Please refer to the manuals for UDS-SQL).

Saving the archive directory (SAVE-DIRECTORY)

The archive directory which is updated during the save operation can also be written to the save file on backup volume or pubset at the end of the save request. In addition to the saved data, the save file then also contains a directory of the saved data.

Saving of the archive directory is initiated by specifying SAVE-DIRECTORY=*YES. It is not saved by default.

The HSMS administrator is permitted to save the archive directory, as is also the archive owner and the co-owner of the archive directory.

The following applies to saving the archive directory:

- If no data was saved with the request, the directory is not saved either.
- Only a full backup of the archive directory is carried out, even if incremental backups are implemented.
- The archive directory is not distributed to different volumes. Saving of the archive directory and the volume used for this are displayed in the report.
- When MAREN V12.0 or higher is used, the backup volume (tape or MTC) which contains the archive directory is assigned the "Volume Contains Directory" attribute in the MAREN catalog. When the directory is restored using IMPORT-FILES, the corresponding volume can be ascertained again.

When the archive directory has been saved, it can be restored from save file on tape, Net-Storage, or private or public disk in the event of an emergency (if it no longer exists or is not readable). To do this you use the IMPORT-FILES statement with *DIRECTORY and SAVE-FILE specified:

- If the directory was saved to tape, Net-Storage (with SAVE-FILE-PROCESSING=*HSMS-V9-COMPATIBLE) or private disk, SAVE-FILE=*BY-VOLUME must be specified.
- If the directory was saved to public disk or within SAVE-FILE-PROCESSING=*HSMS-V10-COMPATIBLE mode to Net-Storage, SAVE-FILE=*BY-PUBLIC-DISK must be specified.

Save files on tape have a several SVID structure. In this case you need to specify an SVID, and in any other case the SFID. The SFID, SVID and volume or pubset are noted in the report on the backup. When MAREN V12.0 or higher is used, the volume (tape or MTC) can also be ascertained from the MAREN catalog.

The attributes of an archive are also stored in the archive directory. If the archive definition has been lost the archive can be restored from the archive directory using the CREATE-ARCHIVE statement (RECONSTRUCT-ARCHIVE=*YES operand).

During reorganization of version backup archive it is not possible to save the archive directory.

Backing up information from a PLAM library (SAVE-PLAM-INFO)

The SAVE-PLAM-INFO operand of the HSMS statements BACKUP-FILES or BACKUP-FILE-VERSIONS causes metadata of a PLAM library to be saved. The default setting *STD causes the archive attribute SAVE-PLAM-INFO of the same name to be evaluated. In the case of the HSMS statement ARCHIVE-FILES the metainformation is saved only if the archive attribute is set accordingly.

HSMS uses the backed-up information of a PLAM library if one or more library elements are to be restored using the HSMS statement RESTORE-LIBRARY-ELEMENTS. You can use RESTORE-LIBRARY-ELEMENTS to

- restore an old version of a library element to a library without having to restore the complete library.
- restore the last backup status of a library element without having to restore the complete library.

The HSMS statement RESTORE-LIBRARY-ELEMENTS can also be used to copy selected elements to another library (target library).

The library elements addressable with RESTORE-LIBARY-ELEMENTS cannot be displayed in a dialog. SHOW-ARCHIVE does not allow any elements to be displayed either; only all PLAM libraries saved with PLAM-INFO=*YES are displayed:

```
//SHOW-ARCHIVE <archiv-name>,SELECT=*FILES(..., -
//              FILE-SAVE-STATE=*SAVED(TYPE=*WITH-PLAM-INFO), ...), ...
```

To restore library elements users must either know the element names at the time of the backup or they have the elements listed beforehand with the ELEMENTS=*LIST-ALL-TO-REPORT operand in the RESTORE-LIBRARY-ELEMENTS statement.

The additional information that is saved requires a certain amount of space on the backup volume and lengthens the backup operation. The space and time required depend on the size of the library and the number of elements present in the backed-up PLAM library.

To obtain optimum performance when performing backups, you should use SAVE-PLAM-INFO

- only if you use the HSMS statement RESTORE-LIBRARY-ELEMENTS frequently, or
- prefer it for large libraries containing a small number of elements if you encounter storage space problems.

Saving the data or catalog entries of BS2000 Net-Storage files

- This section refers only to backups via BACKUP-FILES statement. Version backup does not provide such an option. Within version backup Net-Storage files are always saved with data and catalog entry.
- This section refers only to Net-Storage files of the type BS2000 file. Net-Storage files of the type node file are always backed up fully – with data and the catalog entry.

The default SAVE-NET-STOR-DATA=*YES in the HSMS statement BACKUP-FILES saves the catalog entries and data of Net-Storage files. With SAVE-NET-STOR-DATA=*NO, only the catalog entries of BS2000 Net-Storage files are saved.

//SHOW-ARCHIVE causes files on Net-Storage for which only the catalog entries have been saved to be displayed with the save type CATL (meaning catalog only). The CATL mark is also displayed in the reports in the SAVE TYPE column.

When only the catalog entry has been backed up for a Net-Storage file, the save version of the file is not incremented.

When backup takes place with SELECT-FILE=*MODIFIED-FILES (incremental backup), Net-Storage files which have not been changed since the last backup are saved according to the following rule:

Last saved with	**Current incremental backup with SAVE-NET-STOR-DATA=**	
	***YES**	***NO**
FULL only	CNS	CATL
CATL only	FULL	CATL
Both FULL and CATL	CNS	CATL
No FULL, no CATL	FULL	CATL

FULL, CATL and CNS specify the save type.

With //RESTORE-FILES, privileged users can use the NET-STORAGE-FILES operand to choose whether data and the catalog entry or only the catalog entry are restored for Net-Storage files:

- *CATALOG-ONLY restores only the catalog entry.
- *CATALOG-ONLY restores only the catalog entry.

Save versions containing FULL saves of a Net-Storage file can also be used for restoring catalog entries of the file on Net-Storage.

Save versions containing FULL saves of a Net-Storage file can also be used for restoring catalog entries of the file on Net-Storage.

Special aspects of SAM node files

Files on Net-Storage with the file type NODE-FILE can be processed by BS2000 as well as open systems. Open system files are stored without structure.

NODE-FILE PAM files are processed as unstructured data by the BS2000. No further special aspects have to be taken into account for backup and recovery.

For SAM node files, the BS2000 access method writes the traditional SAM data structure into Net-Storage. On their way into the UFS, Net-Client removes the SAM specific structure information will be removed from the data blocks, which are to be migrated, and the code is converted into an ASCII code. This means that a file is saved on the remote Net-Storage, which could be considered as a text file by the open world.

There are two possible ways to access a SAM node file from BS2000:

- one is to maintain the SAM structures in the BS2000 application by adding them with Net-Client during the write process when the files are migrated from the Net-Server or
- without conversion, i.e. without adding SAM structures to the data blocks and without code conversion to EBCDIC (this is called RAW mode).

Accessing the SAM node files in RAW mode is more efficient and is the best way to achieve fast backups. However, SAM node files backed up this way can only be recovered as NODE-FILE type files on Net-Storage.

RAW mode is used if the SAVE-SAM-STRUCTURE=*NO operand is specified for the backup in the SAVE-OPTIONS; this is the default value, due to the better performance.

If SAVE-SAM-STRUCTURE=*YES is specified, the conversion of the file is used for the backup of the SAM node file. This process creates data blocks in SAM format and the data is converted into EBCDIC characters according to the coded character set and net coded character set in the catalogue entry of the file. This means that the “true” SAM file will be written to the save file. This backup type is less efficient, however, has the benefit of enabling a restore of the saved files on both Net-Storage (BS2000 or NODE-FILE type) and Public-Space (NEW-SUPPORT). So SAVE-SAM-STRUCTURE=*YES enables highly flexible restoring.

> ¡ Within version backup SAM node files are always saved with their SAM structure.

4.1.6 Automatic duplication of save files

In a backup to magnetic tape cartridge, the save file is automatically duplicated if the following requirements are met:

- A shadow archive was assigned to the backup archive concerned.
- Automatic duplication has not been deactivated. In other words: in the OPERATION-CONTROL operand of the HSMS statement BACKUP-FILES, the value *ALLOWED must be specified for SHADOW-COPY.

The save file is copied into the assigned shadow archive at the end of the backup run. The copy in the shadow archive is given the same save version ID and save file ID as the original, i.e. it receives the same time stamp.

If the directory file is to be saved at the end of the backup run (SAVE-DIRECTORY=*YES), the directory file of the assigned shadow archive is also saved at the end of the automatic duplication.

The directory file of the shadow archive, however, can differ from that of the associated backup archive because not all backups are copied to the shadow archive (backups to disk, backups for which SHADOW-COPY=*INHIBITED is specified, a new shadow archive can be linked to a non-empty backup archive).

If a save file in a backup archive is to be extended, the automatic duplication mechanism also attempts to extend the save file with the same SFID in the associated shadow archive. If this save file does not exist, duplication (and possible deletion of the files) is not carried out.

When copying explicitly to the shadow archive a new save file is generated there or an existing save file is updated in the same way as when saving in the main archive.
In the context of a disaster prevention strategy it may make sense to store the tape copy immediately in a specially protected room after every copy operation. However, the save files in the shadow archive cannot then be updated as in the main archive, but a new save file must be created after each copy operation. This mode is set using the archive attribute SHADOW-COPY=*ALLOWED-AND-NEW-SFID.

The automatic duplication mechanism enables the precautionary measures against data loss to be adapted to the requirements and to the storage capacity of a computer center:

- Maximum data security, with very high costs with regard to tape consumption and high costs with regard to backup times:
  - Each system backup archive is assigned a shadow archive.
  - All backups are performed with the specification SHADOW-COPY=*ALLOWED and the archive attribute SHADOW-COPY=*ALLOWED-AND-NEW-SFID.

  Result: Every backup on S2 is automatically copied to the shadow archive and the tape copy is then immediately placed in a secure storage location.

- Maximum data security, with high costs with regard to tape consumption and backup times:
  - Each system backup archive is assigned a shadow archive.
  - All backups are performed with the specification SHADOW-COPY=*ALLOWED.

  Result: Every backup on S2 is automatically copied into the shadow archive.

- Medium data security, with average costs for tape consumption and backup times:
  - Each system backup archive is assigned a shadow archive.
  - All weekly full backups are performed with the specification SHADOW-COPY= *ALLOWED.
  - The incremental backups in between are performed with the specification SHADOW-COPY=*INHIBITED.

  Result: Only the weekly full backups on S2 are copied to the shadow archive.

- Low data security, with lower costs for tape consumption and backup times:
  - No shadow archives are assigned to the system backup archives.

  Result: The automatic duplication mechanism is not used.

4.1.7 Backing up to disk or Net-Storage

The data volumes requiring backup in computer centers continue to increase. Since it can be difficult to increase the number of available devices and the time available for backup, backups to disk can be of great help. While tape devices often can only be used during restricted periods, disk availability is always guaranteed. For this reason, the data is first backed up to disk or Net-Storage. Later they can then be moved to tape. Possible criteria for the move time are, for example, the availability of MTC devices or the reaching of a particular disk utilization.

Depending on the archive type, save files are moved with the following statements:

- Backup archive: MOVE-SAVE-FILE
- Version backup archive: REORGANIZE-VERSION-BACKUP

4.1.7.1 Move save files and node save files via MOVE-SAVE-FILES statement

The MOVE-SAVE-FILES statement enables both save files and node save files to be moved conveniently from disk or Net-Storage to tape. Moreover, from HSMS V12.0A save files can also be moved from tape to public disk or Net-Storage. The statement combines the steps copying and deletion of a number of save files or node save files from original storage. It is thus also used to reorganize a disk or free tapes (only HSMS as of V12.0A).

In a move request several save files are transferred from one storage to another:

- Save files can be moved from disk, Net-Storage or tape to public disk, Net-Storage or another tape
- Node save files can be transferred from disk to tape.

At the same time, MOVE-SAVE-FILES deletes the save files no longer required from the original storage. As no shadow archives are supported with backups to disk or Net-Storage, the save files are implicitly copied into the shadow archive when they are moved to tape (as with a direct save to tape). An option is available to suppress copying to the shadow archive.

After a save file or node save file has been moved, the same status is achieved as for an original backup directly to the new storage.
A save file or node save file is always copied in its entirety (a reduction as with explicit copying is not possible). It thus also contains all Cataloged-Not-Saved (CNS) entries of the original save file. A restore from the backup archive provides the same result after the move as before.

Note

> A direct backup to tape (for this archive) may only be performed after the move request has been completed. Otherwise, problems occur during the move because the continuation on tape would no longer correspond to the sequence of the save versions.

Selecting files for moving

The following criteria are available for selecting the save files to be moved:

- Either save files on disk storages (private, public disks and Net-Storage) or save files on tapes are to be moved:

  `FROM-STORAGE = *DISK(...)`
  `FROM-STORAGE = *S2-STORAGE-LEVEL`
- Minimum retention time of the save or node save files on disk:
  All save or node save files which have been residing on disk for at least the specified number of days or longer are moved.

  `FROM-STORAGE=*DISK(MINIMUM-DAYS-ON-S1=`*`tage`*`, ...)`
- Size of the disk storage space to be released by the move:
  Only as many save or node save files are moved from disk to another storage as are required to release the specified number of 2-Kbyte blocks by the move.

  `FROM-STORAGE=*DISK(... ,RELEASE-PAGES=`*`pam-seiten`*`)`

Creating or continuing moved save files

- In the case of backup archives with single SVID structure, the save files are created anew when they are moved (default). A new save file is created on tape for each moved save file.
- A number of save files on disk or Net-Storage are combined into one save file. In the case of backup archives with several SVID structure, the last save file created on S2 can be continued. This option is not available if a backup to tape took place after the backup to disk.
- After the move the new save versions have the date of the relevant old ones incremented by 1. This ensures that the sequence of all save versions is guaranteed. The save files are stored in the archive directory with the date of the MOVE-SAVE-FILES time.

For the process from **tape to disk**, each save version is moved to a separate save file on disk (the save file id (SFID) of the new save file will be the previous save version id (SVID) with its timestamp plus one second).

For the process from **S2-storage level to S2-storage level**, the result depends on the type of the archive:

- For several-svid archives a new save file is created. All save versions from S2-storage-level are moved to it. Each copy of the save versions receives a SVID equal to the original SVID plus one second.
- For single-svid archives each save version is copied to a separate save file. The copy of save version has a SVID equal to the original SVID plus one second. Each new save file has a SFID with current time stamp.

4.1.7.2 Move save files when reorganizing version backup

Moving save files with MOVE-SAVE-FILE is not possible for version backups. However, function of reorganizing version backups provides possibility of changing storage of the save files being reorganized. By means of the statement REORGANIZE-VERSION-BACKUP it is possible to specify storage to which save files will be copied.

```
TO-STORAGE     = *S2-STORAGE-LEVEL(...)
```

Save files to be reorganized can be selected by their creation date:

```
SAVE-FILE-ID         = *BY-ATTRIBUTES(CREATED-BEFORE=
```

A save version created by a reorganization run receives the current timestamp as a SVID. The functionality of the REORGANIZE-VERSION-BACKUP statement combines that of several statements:

- copying of several save files with selecting only valid file versions to a given storage (COPY-SAVE-FILE)
- subsequent purge of original save files from the archive.

In case of reorganization from disk to S2-level, each save file on disk after reorganization will be presented as save version with current date as a svid in a new save file on the tape. When reorganizing to disc storages all the save versions/save files are copied to separate save files on the disc storage. New save files are created: no continuation of save files is provided during the reorganization.

The retention period of save files created by reorganization is taken from the version backup archive attributes.

4.1.8 Restoring backed up BS2000 files and job variables

Restoring is the copying of saved data from the backup archive to the processing level. It is initiated by means of the HSMS statement RESTORE-FILES.

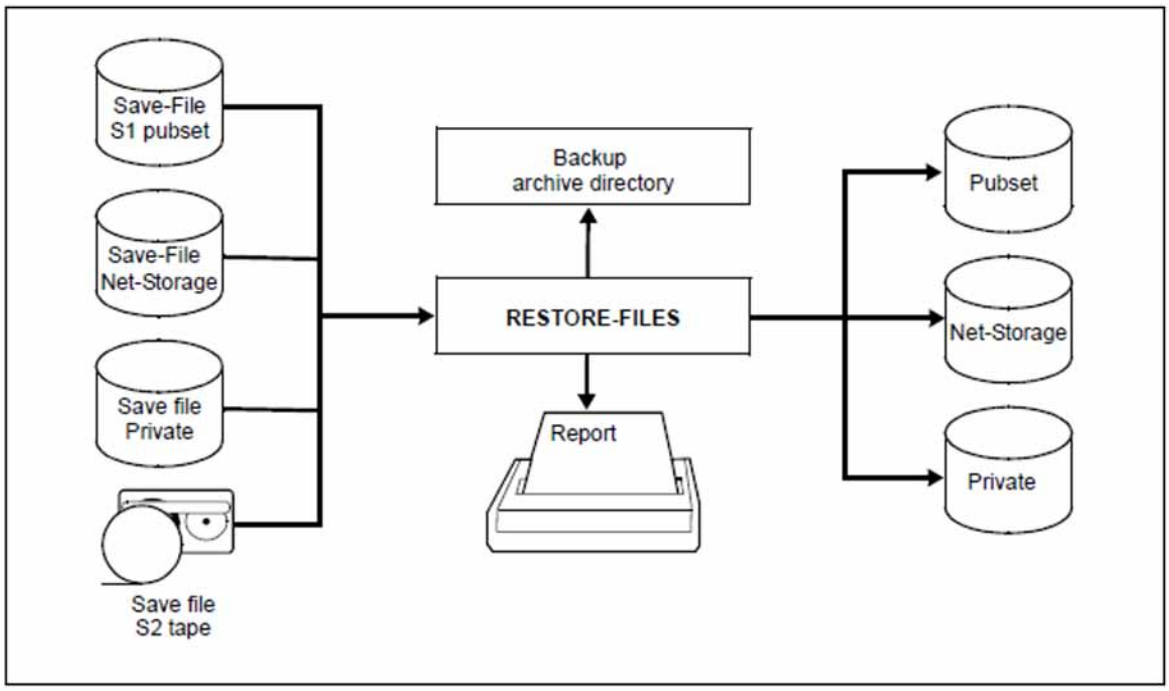


Figure 10 a: Restore from backup archives using RESTORE-FILES

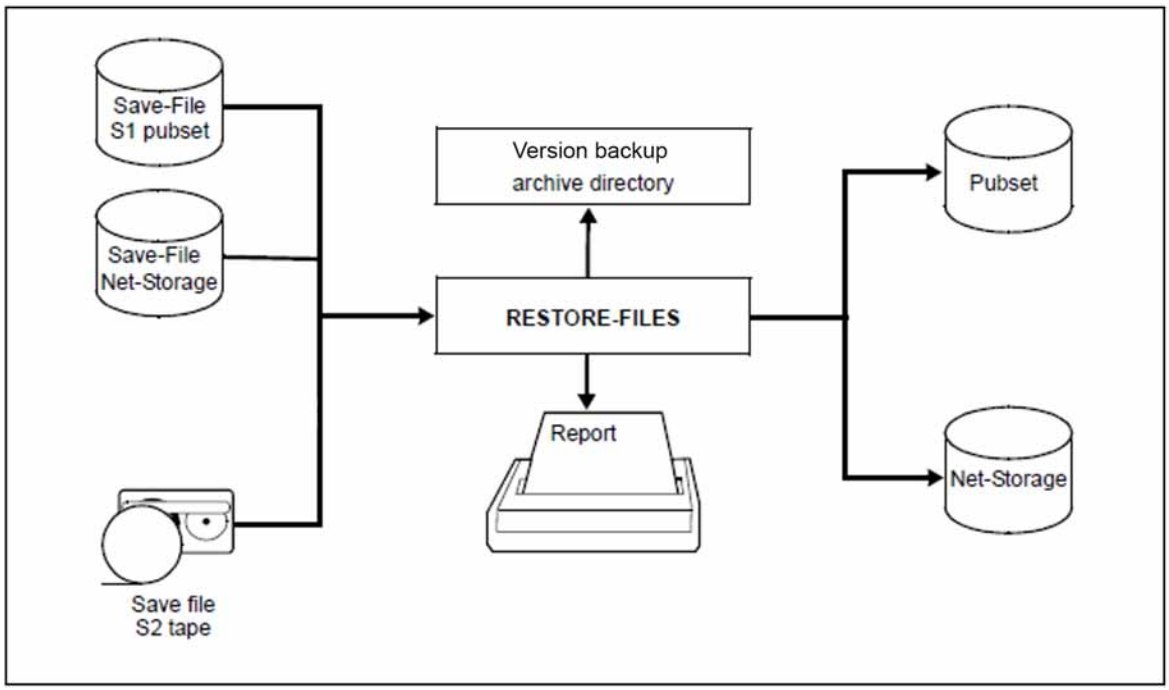


Figure 10 b: Restore from version backup archives using RESTORE-FILES

A RESTORE-FILES statement that refers to BACKUP statements is restricted to the environment in which it was issued. If a RESTORE-FILES statement refers to SF pubsets, it cannot include any SM pubsets in the processing; if it refers to an SM pubset, it can only process one SM pubset at a time.

Nonprivileged users can restore files and job variables that either belong to them or for which they possess co-owner rights (see the “SECOS” manual [16]). These objects are restored:

- either from the standard system backup or system version backup archives or
- from backup archives which are defined in the user’s environment (SF pubsets or SM pubsets) or
- from backup archives which are defined in the environment of other users, provided that the user is authorized to access these archives

In the case of restoring from a backup archive, the files are created with the same catalog attributes that applied for saving with FULL or MIGF (unless the catalog ID or user ID has been changed).

When restoring migrated backup files (i.e. those stored as “migrated”), the HSMS administrator can choose whether just the catalog entries are put on SO storage, or the data from the migrated files as well. Furthermore, he or she can also restore the data of migrated files to a migration level.

Restoring from version backup archives is only possible on BS2000 OSD V11.0B or higher.

4.1.8.1 Restoring large files (> 32 GB)

Large files (>= 32 GB) can only be restored on pubsets with the attributes LARGE-OBJECTS=*YES and LARGE-FILES-ALLOWED=*YES.

If a file >= 32 GB is read into a pubset with the attribute LARGE-FILES-ALLOWED=*NO, this will always cause a error. This also applies, even if the file has been selected implicitly, for example, when a number of files have been selected using a selection method such as partial selection or a wildcard selection.

Backups from older BS2000 versions can also be restored in higher BS2000 versions without restrictions, this applies to standard pubsets and pubsets for which the attribute has been set to LARGE-OBJECTS=*YES.

When carrying out a partial backup (backup) the following must be applied:
If the partial backup version to be restored, or one of the associated full backup files is a file that is more than 32 GB in size, then the restoration process can only be carried out on a pubset which has the attributes LARGE-OBJECTS=*YES and LARGE-FILES-ALLOWED=*YES.

4.1.8.2 Selecting save versions

The data to be restored can typically be fetched from any save version managed in the archive.

The criteria for selection of save versions from which data are to be restored are different for restoring from backup archives and version backup archives. The difference is caused by the concept of the backups.

Selecting save versions from a backup archive

The files and job variables to be restored can typically be fetched from any save version managed in the archive. This is useful because RESTORE-FILES normally refers to specific files or job variables identified via FILE-NAMES, PATH-NAMES or JV-NAMES. All save versions are searched for the specified files/job variables. By default they are taken from the most recent save version that contains them.

As an alternative, the search can be restricted to the most recently created save version by specifying SELECT-SAVE-VERSIONS=*LATEST(...). In this case the files and job variables which were marked as “cataloged-not-saved” (CNS) can be restored as follows:

- either from the latest save version created, by specifying SELECT-SAVE-VERSIONS=*LATEST(CREATED-AFTER=*EARLIEST-DATE) or
- from the latest save version created after a given date, by specifying SELECT-SAVE-VERSIONS=*LATEST (CREATED-AFTER=<date>).

> The following should be taken into account when selecting save versions for restoring Net-Storage files: When a save version which is noted as Cataloged-Only (CATL) and is more recent than the last save version with a full backup (FULL) is selected, no Net-Storage files are restored. In this case a range of save versions should be selected which does not contain the CATL version and the restore operation should be repeated.

Restoring catalog entries of BS2000 Net-Storage files:

> This section refers only to Net-Storage files of the type BS2000 file. It is not possible only to restore the catalog entries of Net-Storage files of the type node file.

The options for selecting save versions also apply for restoring the catalog entry of a BS2000 Net-Storage file. The most recent save version containing the catalog entry only (save type CATL) is selected first. If a save version with CATL exists or if the save version with CATL does not contain the most recent version of the file, the catalog entry of the Net-Storage file can also be restored from the most recent save version which also contains data (save type FULL or PART).

Whether a save version which only contains the catalog entry or the catalog entry and data is used for restoring the catalog entry of a Net-Storage file is determined according to the following rule:

- The selection of the save versions contains both save types CATL and FULL.
  - The selection of the save versions contains both save types CATL and FULL
  - If the version and CFID of the file are the same, the catalog entry is restored from the save version with CATL.
  - If the version and CFID of the file are different and the most recent version is of the type FULL, the catalog entry is restored from this save version with FULL.
  - If the selection contains save versions with CNS, the catalog entry from the most recent save version which contains the most recent version of the file (either CATL or FULL) is restored.
- If the selection contains no save version with CATL but does contain one with FULL, the catalog entry is restored from this save version with FULL.

In some cases, several save versions can be created on the same day. The term “latest save version” is therefore extended to mean a number of save versions which were all created on the same day (specification: SELECT-SAVE-VERSIONS=*LATEST(DAY-INTERVAL=*YES). But this operand value is only available for saved files.

Additionally save versions with a specific save version name or save date and time can be selected for restore.

```
SELECT-SAVE-VERSIONS=*BY-ATTRIBUTES(...)
```

It is also possible to specify an interval from which to select a save version.

Example of how save versions are restored

- The files have been saved as follows by various backup runs:

Save version name	**BACKUP01**	**BACKUP02**	**BACKUP03**	**BACKUP04**
FILE.1	FULL[1]	FULL	FULL	CNS
FILE.2	FULL	CNS	FULL	
FILE.3	FULL			FULL
FILE.4	FULL			

[1] = Save type

- The result of the various restore runs looks as follows:

//RESTORE-FILES FILE-NAMES=FILE.,- SELECT-SAVE-VERSIONS=	**Restored files**	**From save version**
*LATEST	FILE.1	BACKUP03
	FILE.3	BACKUP04
*ALL	FILE.1	BACKUP03
	FILE.2	BACKUP03
	FILE.3	BACKUP04
	FILE.4	BACKUP01
*BY-ATTRIBUTES(SAVE-VERSION-NAME=BACKUP02)	FILE.1	BACKUP02
	FILE.2	BACKUP01

Selecting save versions from version backup archives

The selection criterion SELECT-SAVE-VERSION for a version backup archive has its own meaning in comparison to other types of archive.

The selection of file versions from version backup archives is performed according to the original SVID only. Thus only the option *BY-ORIGINAL-DATE can be specified when restoring from version backup archives. All other options are rejected. For version backup archives the selection criteria is “file-version”-oriented (not “save-version”-oriented).

To restore the latest version of the file the SELECT-SAVE-VERSIONS operand should have the value *STD which is equal to SELECT-SAVE-VERSIONS=*BY-ATTRIBUTES(SAVE-VERSION-NAME=*ANY,SAVE-VERSION-DATE=*BY-ORIGINAL-DATE(CREATED-BEFORE=*LATEST-DATE,CREATED-AFTER=*EARLIEST-DATE)).

The order of saved file versions is not guaranteed in a version backup archive. After a reorganization, the newest file version may be not in the newest save version (SVID).

If the restore of files with the latest original SVID is needed, the specification SAVE-VERSION-DATE = *BY-ORIGINAL-DATE (CREATED-BEFORE = *LATEST-DATE, CREATED-AFTER = *SAME-AS-BEFORE) should be chosen. For this selection criteria HSMS determines the highest original SVID within the whole version backup archive and then choose only those files which have been originally saved within the save version.

All possible specifications of selecting the save versions in version backup archives with examples are described in the “HSMS Vol. 2” manual [1].

4.1.8.3 Reorganizing disk storages

Pubsets can be reorganized by performing a backup followed by restoration. Such a reorganization will be required whenever the distribution of the disk files over an increasing number of extents reaches the point where the files cannot be extended because the maximum number of extents has been reached. With HSMS all files can be written back contiguously, thus reorganizing the storage space:

```
//RESTORE-FILES REPLACE-FILES-AND-JV=*YES(REORGANIZE-SPACE=*YES)
```

By additionally specifying the operand RELEASE-UNUSED-SPACE=*YES, storage space requirements can be reduced even further: any pages behind the file's last-page pointer, i.e. pages allocated but not occupied, are released.

Example of restoring BS2000 Net-Storage files

- The files have been saved as follows by various backup runs:

Save version name	BACKUP01	BACKUP02	BACKUP03	BACKUP04
NET.FILE.1	FULL (1)	CNS (1)	CATL (1)	CNS (1)
NET.FILE.2	CATL (1)	CATL (2)	CNS (2)	-
NET.FILE.3	CATL (1)	FULL (1)	CATL (2)	CNS (2)
NET.FILE.4	FULL (1)	-	CNS (1)	-

Here FULL, CATL, CNS are save types. The different numbers in brackets show different versions (or CFIDs) of the files.

- The result of the various restore runs looks as follows:

//RESTORE-FILES FILE-NAMES=NET.FILE.,- NET-STORAGE-FILES=*DATA-AND-CATALOG,- SELECT-SAVE-VERSIONS=	**Restored files**	**From save version**
*ALL	NET.FILE.1	BACKUP01
	NET.FILE.2	ARC0081
	NET.FILE.3	ARC0507
	NET.FILE.4	BACKUP01
*LATEST	NET.FILE.1	BACKUP01
	NET.FILE.2	-
	NET.FILE.3	ARC0507
	NET.FILE.4	-
BACKUP03	NET.FILE.1	BACKUP01
	NET.FILE.2	ARC0081
	NET.FILE.3	ARC0507
	NET.FILE.4	BACKUP01
*INTERVAL(AFTER=BACKUP02,BEFORE=BACKUP03)	NET.FILE.1	BACKUP01
	NET.FILE.2	ARC0081
	NET.FILE.3	ARC0507
	NET.FILE.4	BACKUP01

Example of restoring the catalog entries of BS2000 Net-Storage files

- The files have been saved as follows by various backup runs:

Save version name	BACKUP01	BACKUP02	BACKUP03	BACKUP04
NET.FILE.1	FULL (1)	CNS (1)	CATL (1)	CNS (1)
NET.FILE.2	CATL (1)	CATL (2)	CNS (2)	-
NET.FILE.3	CATL (1)	FULL (1)	CATL (2)	CNS (2)
NET.FILE.4	FULL (1)	-	CNS (1)	-

Here FULL, CATL, CNS are save types. The different numbers in brackets show different versions (or CFIDs) of the files.

- The result of the various restore runs looks as follows:

//RESTORE-FILES FILE-NAMES=NET.FILE.,- NET-STORAGE-FILES=*CATALOG-ONLY,- SELECT-SAVE-VERSIONS=	Restored files	From save version
*ALL	NET.FILE.1	BACKUP03
	NET.FILE.2	BACKUP02
	NET.FILE.3	BACKUP03
	NET.FILE.4	BACKUP01
*LATEST	NET.FILE.1	BACKUP03
	NET.FILE.2	-
	NET.FILE.3	BACKUP03
	NET.FILE.4	-
BACKUP02	NET.FILE.1	BACKUP01
BACKUP02	NET.FILE.2	BACKUP02
BACKUP02	NET.FILE.3	BACKUP01
BACKUP02	NET.FILE.4	-
*INTERVAL(AFTER=BACKUP01,BEFORE=BACKUP02)	NET.FILE.1	BACKUP01
	NET.FILE.2	BACKUP02
	NET.FILE.3	BACKUP02
	NET.FILE.4	BACKUP01

//RESTORE-FILES FILE-NAMES=NET.FILE.,- NET-STORAGE-FILES=*CATALOG-ONLY,- SELECT-SAVE-VERSIONS=	**Restored files**	**From save version**
*ALL	NET.FILE.1	BACKUP03
	NET.FILE.2	BACKUP02
	NET.FILE.3	BACKUP03
	NET.FILE.4	BACKUP01
*LATEST	NET.FILE.1	BACKUP03
	NET.FILE.2	-
	NET.FILE.3	BACKUP03
	NET.FILE.4	-
BACKUP02	NET.FILE.1	BACKUP01
	NET.FILE.2	BACKUP02
	NET.FILE.3	BACKUP01
	NET.FILE.4	-
*INTERVAL(AFTER=BACKUP01,BEFORE=BACKUP02)	NET.FILE.1	BACKUP01
	NET.FILE.2	BACKUP02
	NET.FILE.3	BACKUP02
	NET.FILE.4	BACKUP01

4.1.8.4 Renaming files and job variables

When restoring files, the NEW-FILE-NAMES operand can be used to rename the files. However, the BS2000 naming conventions must be observed. Renaming is not possible for individual files but only for all files restored by the same request.

The following options exist for renaming:

- The files can be given a prefix or suffix.
- The files can be moved to another catalog and/or user ID provided the user is permitted to access this catalog ID or user ID.
- The HSMS administrator has the additional right to rename the files under another user ID.

> When restoring the catalog entries of BS2000 Net-Storage files (//RESTORE-FILES with the NET-STORAGE-FILES=*CATALOG-ONLY operand), it is only possible to move the files to another catalog. Neither moving to another user ID nor renaming the file is permitted.

In the same way, job variables can be renamed using the NEW-JV-NAMES operand during their restoration from backup archives.

4.1.8.5 Restoring from a shadow archive

If a backup archive was assigned a shadow archive, the shadow archive can be used for restoring the files. This is helpful if files have been destroyed in the associated archive or the relevant volume cannot be accessed.

4.1.8.6 Restoring elements from a PLAM library

You can restore elements located in the backup archive or version backup archive of a PLAM library with the HSMS statement RESTORE-LIBRARY-ELEMENTS. The preceding backups or version backups of the library must contain the metadata about the library elements (SAVE-PLAM-INFO=*YES in the BACKUP-FILES or BACKUP-FILE-VERSIONS statements or in the backup or long-term archive's save options).

If a PLAM library is to be saved with the metadata about the library elements, the library must be located at S0 level. When PLAM libraries are migrated this metadata is not saved.

The library must be accessible on S0, in other words may not be migrated. The elements can be overwritten and renamed as required.

The backup version of the library containing the specified elements must be known.

The library elements can also be restored to a target library which is not the source library.

The element names of a PLAM library saved with PLAM-INFO=*YES are only contained in the PLAM information in the save file but not in the archive directory. The element names of a PLAM library saved in this way can be listed using the ELEMENTS=*LIST-ALL-TO-REPORT operand in the RESTORE-LIBRARY-ELEMENTS statement.

4.1.8.7 Restoring Net-Storage files

Restoration can be restricted to files on Net-Storage (STORAGE-TYPE operand).

Net-Storage files of the BS2000 or node file type, whose SAM structure has been saved, can be restored to different volume types. During this process, their file type can be changed. In contrast, net storage files of the file type node file, whose SAM structure has not been saved, can only be restored on a Net-Storage volume and with their original file type node file.

4.1.9 Backing up node files of the local BS2000-UFS and nodes S0

For node files, HSMS supports the backup types full backup and incremental backup, but not partial backups.

All users are authorized to perform these backups. System backups, however, are the prerogative of the HSMS administrator.

Nonprivileged users are restricted to saving and restoring their own node files residing on the local BS2000-UFS. The HSMS administrator is authorized to process the node files of all users that reside on the local BS2000-UFS as well as those located on a remote file system.

Node files are saved using the HSMS statement BACKUP-NODE-FILES. Individual files can be selected via their names (using wildcards) and via the SELECTION-BOUNDARY operand. The backup may optionally be restricted to the explicitly specified files or, if saving a directory, include all its subdirectories and subordinated files in the file system or file tree (see section “Selection of node files of the BS2000-UFS”).

When node files are saved, the directory file cannot be saved at the same time in the same save run because the directory file is a BS2000 file and therefore has to be saved in a BS2000 save run.

4.1.9.1 Scope of a backup

Depending on the file type of the node file to be saved, HSMS saves either only its inode or both inode and data:

Node file to be saved	UNIX
Regular file	yes
Directory	yes
Symbolic link	yes
Hard ink	yes
Semaphore	no
FIFO file	no
Character special file	no
Block special file	no
Shared data	no
File system /proc	no

With a symbolic link, HSMS processes only the inode and the path of the node file it references.

A hard link is processed as follows: for the first occurrence of a hard link to a file, HSMS processes both inode and data as with a normal node file. For all subsequent hard links to the same file, HSMS writes the path name and a pointer to the first hard link to the save file.

For more detailed information about symbolic links and hard links please refer to the section on the *ln* command in the “POSIX Commands” manual [13].

4.1.9.2 System backup archive for node files

The HSMS administrator must create the system backup archive as a public archive with read access (USER-ACCESS=*ALL-USERS/ACCESS=*READ). All users can obtain read access to this archive via its symbolic name SYSNODEBACKUP.
Write access to the system backup archive is always restricted to the HSMS administrator, any specification of ACCESS=*WRITE being ignored.

4.1.9.3 Full and incremental backup

The same rules apply to node files as for BS2000 files.
With incremental backups, the entries in the directory file marked “CNS” (cataloged-not-saved) are interpreted as “still present in node file system but not modified since the last backup”.

A full backup of S0 or of the most recent backup in the archive is performed like any full backup. HSMS always saves the most recent version of the metadata (ownership and access rights, time stamp and other file attributes).

The type of backup is controlled via the SELECT-FILES operand of the HSMS statement BACKUP-NODE-FILES. Specifying SELECT-FILES=*MODIFIED-FILES or *BY-CATALOG-MODIFICATION results in an incremental backup, while SELECT-FILES=*ALL-FILES produces a full backup.

4.1.9.4 Automatic duplication of node save files

In a node backup to magnetic tape cartridge, the save file is automatically duplicated if the node backup archive concerned was assigned a shadow archive. The automatic duplication mechanism must be activated by default (see the OPERATION-CONTROL operand of the HSMS statement BACKUP-NODE-FILES).

The save file is copied into the assigned shadow archive at the end of the backup run. The copy in the shadow archive is given the same save version ID and save file ID as the original, i.e. it receives the same time stamp.

The directory file of the shadow archive can differ from that of the associated node backup archive because not all backups are copied to the shadow archive (backups to disk, backups for which SHADOW-COPY=*INHIBITED is specified, a new shadow archive can be linked to a non-empty backup archive).

If a save file in a node backup archive is to be extended, the automatic duplication mechanism also attempts to extend the save file with the same SFID in the assigned shadow archive. If this save file does not exist, duplication is not carried out.

The automatic duplication mechanism enables the precautionary measures against data loss to be adapted to the requirements and to the storage capacity of a computer center:

- Maximum data security, with very high costs with regard to tape consumption and high costs with regard to backup times:
  - Each system backup archive is assigned a shadow archive.
  - All backups are performed with the specification SHADOW-COPY=*ALLOWED and the archive attribute SHADOW-COPY=*ALLOWED-AND-NEW-SFID.

  Result: Every backup on S2 is automatically copied to the shadow archive and the tape copy is then immediately placed in a secure storage location.

- Maximum data security, with high costs with regard to tape consumption and backup times:
  - Each system backup archive is assigned a shadow archive.
  - All backups are performed with the specification SHADOW-COPY=*ALLOWED.

  Result: Every backup on S2 is automatically copied into the shadow archive.

- Medium data security, with average costs for tape consumption and backup times:
  - Each system backup archive is assigned a shadow archive.
  - All weekly full backups are performed with the specification SHADOW-COPY= *ALLOWED.
  - The incremental backups in between are performed with the specification SHADOW-COPY=*INHIBITED.

  Result: Only the weekly full backups on S2 are copied to the shadow archive.

- Low data security, with lower costs for tape consumption and backup times:
  - No shadow archives are assigned to the system backup archives.

  Result: The automatic duplication mechanism is not used.

4.1.9.5 Backup of LATEST-BACKUPS-OR-S0

In a node environment save files can be created in multiplex operation. If one of the last backups which was used as input during offline copying was created in multiplex operation, the newly created save file must also be created entirely in multiplex operation. This means that all input save versions which were not created in multiplex operation but were used during offline copying must also be multiplexed in the new output save file so that the complete structure of the new save file remains consistent. Thus if a full backup is performed with FROM=*LATEST-BACKUPS-OR-S0 and at least one save version is multiplexed in the HSMS archive in question, the backup of S0 (in other words the modified files) is also created in multiplex operation, even if the user did not request this. The multiplex factor in this case is 1.

4.1.10 Restoring backed up node files

Restoring backed up node files of the BS2000-UFS or remote nodes S0 is initiated by means of the HSMS statement RESTORE-NODE-FILES.

Nonprivileged users can restore their own node files residing on the local BS2000-UFS. They can restore their node files from any backup archive to which they have access (USER-ACCESS=*ALL-USERS). A restricted form of the HSMS statement RESTORE-NODE-FILES is available to them for this purpose.

The HSMS administrator has access to all backup archives in order to restore the node files of all users that reside on the local BS2000-UFS or any remote file system.

When restoring the hard links of a node file, the corresponding inodes and data are restored first. Subsequently, the link between the path name and the inode is restored. If the node file to be restored existed on the node S0 before restoration, the hard links existing at restore time are retained.

4.1.10.1 Selecting save versions

The same selection rules apply to node files and to BS2000 files.

4.1.10.2 Renaming node files

The NEW-PATH-NAMES operand can be used to rename node files during the restore. Note, however, that this cannot be done for individual files but only for all node files restored by the same request. The following options are supported:

- The file names can be given a prefix, a suffix or a new path name.
- The node files can be moved to another node S0.
- Part of the file name can be modified.

If the original file names were used with symbolic links or shortcuts, renaming can result in inconsistent statuses: The content of symbolic links or shortcuts is not modified when renaming takes place.

4.1.10.3 Restoring from a shadow archive

If a backup archive was assigned a shadow archive, the shadow archive can be used in the RESTORE-NODE-FILES statement. This is helpful if backed up files of the original archive have been destroyed or the associated volumes cannot be accessed.

4.1.11 Examples of backups

This section provides examples dealing with the following subjects:

- Automatic full and partial backup using repeat jobs
- Backup to S1 with and without compression
- Partial backup after catalog modification
- Version backup in SM environment

with corresponding notes on restoration.

An example of backing up the archive directory and restoring an archive together with archive directory is provided in the section "Example of long-term archival".

Automatic full and partial backup using repeat jobs

Weekly full backups and daily partial backups are implemented via automatically initiated repeat jobs.

```
/.FULLSAVE SET-LOGON-PARAMETERS TSOS  ————————————————————————————  (1)
/ START-HSMS
//MODIFY-ARCHIVE ARCHIVE-NAME=HSMS.BACKUP -  ————————————————————————  (2)
//  SAVE-FILES=*DELETE(SAVE-FILE-ID=*BY-ATTR -
//    (SAVE-FILE-STATE=*OBSOLETE))
//BACKUP-FILES FILE-NAMES=*ALL, -  ——————————————————————————————————  (3)
//  SELECT-FILES=*ALL-FILES, -
//  ARCHIVE-NAME=*SYSBACKUP, -
//  SAVE-FILE=*NEW(RETENTION-PERIOD=14, -
//    USER-ACCESS=*ALL-USERS), -
//  TO-STORAGE=*S2-STORAGE-LEVEL, -
//  OPERATION-CONTROL=*PAR(REPORT=*FULL, -
//    OUTPUT=MANUAL.REPORT.FULL, -
//    EXPRESS-REQUEST=*YES)
//END
/ SET-JOB-STEP
/ EXIT-JOB SYSTEM-OUTPUT=*DELETE
/.PARTSAVE SET-LOGON-PARAMETERS TSOS ——————————————————————————————  (4)
/ START-HSMS
//BACKUP-FILES FILE-NAMES=*ALL, -  ——————————————————————————————————  (5)
//  SELECT-FILES=*MODIFIED-FILES(PARTIAL-FILE-SAVE=*YES -
//  (MINIMUM-SIZE=200)), -
//  ARCHIVE-NAME=*SYSBACKUP, -
//  SAVE-FILE=*NEW(RETENTION-PERIOD=14, -
//    USER-ACCESS=*ALL-USERS), -
//  TO-STORAGE=*S1-STORAGE-LEVEL, -
//  OPERATION-CONTROL=*PAR(REPORT=*FULL, -
//    OUTPUT=MANUAL.REPORT.PART, -
//    EXPRESS-REQUEST=*YES)
//END
/ SET-JOB-STEP
/ EXIT-JOB SYSTEM-OUTPUT=*DELETE
/enter-job hsms.e.full-save,proc-admis=(user-id=tsos, -   ————————————  (6)
/ account=adminstr),start=at(date=*today,time=21:00),repeat-job=*weekly
% JMS0066 JOB 'FULLSAVE' ACCEPTED ON 16-08-12 AT 14:27, TSN = 0ABW
/enter-job hsms.e.part-save,proc-admis=(user-id=tsos, -   ————————————  (7)
/  account=adminstr),start=at(time=23:30),repeat-job=*daily
% JMS0066 JOB 'PARTSAVE' ACCEPTED ON 16-08-12 AT 14:27, TSN = 0ABY
```

(1) ENTER file HSMS.E.FULL-SAVE

(2) Before backup, MODIFY-ARCHIVE is used to release the obsolete save files so that magnetic tape cartridges from the volume pool are available for the subsequent backup run. The archive cannot be specified in symbolic form here.

(3) The BACKUP-FILES statement specifying a weekly full backup is issued: all files on all pubsets are saved to the system backup archive in their entirety; the saved files cannot be deleted for the next two weeks.

(4) ENTER file HSMS.E.PART-SAVE

(5) The BACKUP-FILES statement specifying daily incremental and partial backup is issued: those pages which have been modified since the last full save in all files larger than 200 pages on all pubsets are saved; all files smaller than 200 pages are saved in their entirety if they have been modified since the last incremental backup.

(6) The batch task for a full save is started; it is repeated once a week.

(7) The batch task for partial backup is started; it is repeated once a day. On the day on which both tasks are started the full save is executed somewhat earlier. The result is that the subsequent partial backup will be (virtually) empty.

Users can obtain the most recent status of a file (a file which has been inadvertently deleted, for example) from this system backup by issuing the following HSMS statement:

```
//RESTORE-FILES FILE-NAMES=<file name>
```

The file is restored, and, depending on the global HSMS parameter OUTPUT, a report is output on the printer or sent as an email attachment. From this archive, users can also restore any status of a file from the last 14 days.

Earlier versions of the file can be selected via the SELECT-SAVE-VERSIONS operand.

Backup to S1 with and without compression

Files are saved to a backup archive on storage level S1, once with and once without compression of the save file. The save files are compared.
Files are restored from the compressed save file.

```
//BACKUP-FILES FILE-NAMES=$MANUAL.FILE.*, -  ————————————————————————  (1)
//  SELECT-FILES=*ALL-FILES, -
//  TO-STORAGE=*S1-STORAGE-LEVEL, -
//  ARCHIVE-NAME=*SYSBACKUP, -
//  SAVE-FILE=*NEW, -
//  OPERATION-CONTROL=*PAR(REPORT=*FULL, -
//    OUTPUT=HSMS.MAN.R.BCF.1, -
//    WAIT-FOR-COMPLETION=*YES)
% HSM0003 HSMS STATEMENT COMPLETED

//END
% HSM0014 HSMS PROGRAM TERMINATED
Report HSMS.MAN.R.BCF.1 (extract):  ————————————————————————————  (2)
***   BACKUP - FILES             HSMS V12.0        FULL         REPORT  *** 2016-08-12  15:
57:53     PAGE   2
REQUEST-NAME=BCF#0AAK REQUEST-DATE=2016-08-12 15:57:37 USER-ID=SYSHSMS  REQUEST-
STATE=COMPLETED WITHOUT ERROR
 %  ARC0002 STATEMENT ACCEPTED. ARCHIVE SEQUENCE NUMBER 'A.160812.155740', VERSION '11.0'
%  ARC0033 ARCHIVE
SUBTASK TSN '0AAL' GENERATED
         SAVE FILE IDENTIFIER - S.160812.155738    SAVE-VERSION-DATE=16-08-12  SAVE-VERSION-
TIME=15:57:38
SUBSAVE              NUMBER       VSNS                0        0:2BC
```

```
/START-HSMS
//BACKUP-FILES FILE-NAMES=$MANUAL.FILE.*, -   ————————————————————————)  (3)
//  SELECT-FILES=*ALL-FILES, -
//  TO-STORAGE=*S1-STORAGE-LEVEL, -
//  ARCHIVE-NAME=*SYSBACKUP, -
//  SAVE-FILE=*NEW, -
//  COMPRESS-FILES=*YES, -   ————————————————————————————————————————————  (4)
//  OPERATION-CONTROL=*PAR(REPORT=*FULL, -
//    OUTPUT=HSMS.MAN.R.BCF.2, -
//    WAIT-FOR-COMPLETION=*YES)
% HSM0003 HSMS STATEMENT COMPLETED

//END
% HSM0014 HSMS PROGRAM TERMINATED
Report HSMS.MAN.R.BCF.2 (extract):   ————————————————————————————————  (5)
***   BACKUP - FILES                HSMS V12.0        FULL        REPORT  *** 2016-08-12  15:
58:12    PAGE    3
REQUEST-NAME=BCF#0AAK REQUEST-DATE=2016-08-12 15:57:53 USER-ID=SYSHSMS  REQUEST-
STATE=COMPLETED WITHOUT ERROR
 %  ARC0002 STATEMENT ACCEPTED. ARCHIVE SEQUENCE NUMBER 'A.160812.155755', VERSION '11.0'
%  ARC0033 ARCHIVE
SUBTASK TSN '0AAM' GENERATED
         SAVE FILE IDENTIFIER - S.160812.155757    SAVE-VERSION-DATE=16-08-12  SAVE-VERSION-
TIME=15:57:57
SUBSAVE              NUMBER       VSNS                 0        0:2BC
```

```
/SHOW-FILE-ATT FILE-NAME=$MANUAL.FILE.*,INFORMATION=SPACE-SUMMARY ————  (6)
:2BY:  PUBLIC:    21  FILES RES=       270  FRE=       45 REL=       27 PAGES
/SHOW-FILE-ATT FILE=:2BC:$*.ARCHIVE.SAVE.FILE.010812.155738., -
/  INFORMATION=SPACE-SUMMARY
:2BC:  PUBLIC:     1  FILE  RES=       255  FRE=        2 REL=        0 PAGES
/SHOW-FILE-ATT FILE=:2BC:$*.ARCHIVE.SAVE.FILE.010812.155757., -
/  INFORMATION=SPACE-SUMMARY
:2BC:  PUBLIC:     1  FILE  RES=        30  FRE=        2 REL=        0 PAGES
//RESTORE-FILES FILE-NAMES=$MANUAL.FILE.*, -   ————————————————————————  (7)
//  REPLACE-FILES-AND-JV=*YES, -
//  ARCHIVE-NAME=*SYSBACKUP, -
//  SELECT-SAVE-VERSIONS=*LATEST, -
//  OPERATION-CONTROL=*PAR(REPORT=*FULL, -
//    OUTPUT=HSMS.MAN.R.RSF.1, -
//    WAIT-FOR-COMPLETION=*YES)
% HSM0003 HSMS STATEMENT COMPLETED

//END
% HSM0014 HSMS PROGRAM TERMINATED
```

(1) The files named FILE. of user ID MANUAL are saved to the system backup archive on S1.

(2) Extract from the HSMS-generated report of the save run with output of the SFID.

(3) The same files are saved once again, but this time ...

(4) ... the data is compressed during writing.

(5) Extract from the HSMS-generated report of the save run with output of the SFID.

(6) Comparison of the saved files with the save files to which they were written shows the reduced space requirements of the compressed save file.

(7) The files are restored from the compressed save file. Decompression need not be explicitly specified, the data is decompressed automatically.

Partial backup after catalog modification

This backup resembles an incremental backup. All files are backed up which were modified after the specified time.

In the example below 5 files (FILE1, FILE2, ..., FILE5) are backed up according to this procedure:

- The files are modified or deleted and occasionally saved after a full backup.
- After the 3rd. day the files are restored again.

1st day: Date: 14.04.2016

```
/START-HSMS
//CREATE-ARCHIVE ARCHIVE-NAME=archive, -
//               ALLOWED-USAGE=*BACKUP(SAVE-FILE-STRUCTURE=*SEVERAL-SVID), -
//               DIRECTORY-NAME=archive.dir, -
//               RETENTION-PERIOD=100
//BACKUP-FILES FILE-NAMES=file*, -
//               SELECT-FILES=*ALL-FILES, -
//               ARCHIVE-NAME=archive, -
//               OPERATION-CONTROL=*PAR(REPORT=*FULL,OUTPUT=l.archive)
//END
```

After the full backup FILE2 is modified and FILE3 deleted.

2nd day: Date: 15.04.2016

```
/START-HSMS
//BACKUP-FILES FILE-NAMES=file*, -
//               SELECT-FILES=*BY-CATALOG-MODIFICATION -
//               (CHANGED-AFTER=*LATEST-SAVE-VERSION-DATE), -
//               ARCHIVE-NAME=archive, -
//               OPERATION-CONTROL=*PAR(REPORT=*FULL,OUTPUT=l.archive)
//END
```

After the backup with *BY-CATALOG-MODIFICATION (here all files that have been modified since the last backup on 14.04.) files FILE2 and FILE4 are modified and FILE5 deleted.

3rd day: Date: 16.04.2016

```
/START-HSMS
//BACKUP-FILES FILE-NAMES=file*, -
//               SELECT-FILES=*BY-CATALOG-MODIFICATION, -
//               ARCHIVE-NAME=archive, -
//               OPERATION-CONTROL=*PAR(REPORT=*FULL,OUTPUT=l.archive)
//END
```

After the backup with *BY-CATALOG-MODIFICATION (here all files that have been modified since the last backup on 15.04.) the archive has the following content:

```
//SHOW-ARCHIVE ARCHIVE-NAME=archive,SELECT=*SAVE-VERSIONS
```

... (*Extract of the output*)

```
M SAV-DATE SAV-TIME  EXP-DATE  SFID             SEL-F  BC IND  USER-ID  SV-NAME
  16-04-14 10:20:02  16-07-23  S.160414.102001  ALL-F  D       UHU
  16-04-15 10:12:44  16-07-24  S.160415.101243  CAT-F  D       UHU
  16-04-16 09:26:12  16-07-25  S.160416.092611  CAT-F  D       UHU
           //SHOW-ARCHIVE ARCHIVE-NAME=archive
```

... (*Extract of the output*)

```
M FILE-NAME                                 VERS SAV-DATE SAV-TIME EXP-DATE TYPE
  FILE1                                     1    16-04-14 10:20:02 16-07-23 FULL
  FILE2                                     1    16-04-14 10:20:02 16-07-23 FULL
  FILE2                                     1    16-04-15 10:12:44 16-07-24 FULL
  FILE2                                     1    16-04-14 10:20:02 16-07-23 FULL
  FILE4                                     1    16-04-14 10:20:02 16-07-23 FULL
  FILE4                                     1    16-04-16 09:26:12 16-07-25 FULL
  FILE5                                     1    16-04-14 10:20:02 16-07-23 FULL
```

Restore of all files:

```
//RESTORE-FILES FILE-NAMES=*ALL, -
//              ARCHIVE-NAME=archive, -
//              OPERATION-CONTROL=*PARAMETERS(REPORT=*FULL,OUTPUT=l.restore)
```

With this all files FILE1 through FILE5 are restored, regardless of whether they have been deleted in the meantime.

```
/SHOW-FILE-ATTRIBUTES FILE-NAME=file*,INFORMATION=*PAR(HISTORY=*YES)
%00000003 :SBZ3:$UHU.FILE1
%  ------------------------------ HISTORY    ------------------------------
%  CRE-DATE   = 2016-02-17  ACC-DATE   = 2016-02-17  CHANG-DATE = 2016-02-17
%  CRE-TIME   =   14:15:21  ACC-TIME   =   14:15:21  CHANG-TIME =   14:15:21
%  ACC-COUNT  = 1           S-ALLO-NUM = 0
%00000003 :SBZ3:$UHU.FILE2
%  ------------------------------ HISTORY    ------------------------------
%  CRE-DATE   = 2016-04-15  ACC-DATE   = 2016-04-15  CHANG-DATE = 2016-04-15
%  CRE-TIME   =   10:18:32  ACC-TIME   =   10:18:32  CHANG-TIME =   10:18:32
%  ACC-COUNT  = 6           S-ALLO-NUM = 0
%00000003 :SBZ3:$UHU.FILE3
%  ------------------------------ HISTORY    ------------------------------
%  CRE-DATE   = 2016-01-22  ACC-DATE   = 2016-01-22  CHANG-DATE = 2016-01-22
%  CRE-TIME   =   13:10:20  ACC-TIME   =   13:10:20  CHANG-TIME =   13:10:20
%  ACC-COUNT  = 4           S-ALLO-NUM = 0
%00000003 :SBZ3:$UHU.FILE4
%  ------------------------------ HISTORY    ------------------------------
%  CRE-DATE   = 2016-04-15  ACC-DATE   = 2016-04-15  CHANG-DATE = 2016-04-15
%  CRE-TIME   =   10:18:52  ACC-TIME   =   10:18:52  CHANG-TIME =   10:18:52
%  ACC-COUNT  = 5           S-ALLO-NUM = 0
%00000003 :SBZ3:$UHU.FILE5
%  ------------------------------ HISTORY    ------------------------------
%  CRE-DATE   = 2016-02-17  ACC-DATE   = 2016-02-17  CHANG-DATE = 2016-02-17
%  CRE-TIME   =   13:42:11  ACC-TIME   =   13:42:11  CHANG-TIME =   13:42:11
%  ACC-COUNT  = 3           S-ALLO-NUM = 0
%:SBZ3: PUBLIC:      5 FILES RES=      15  FREE=       10  REL=       0 PAGES
```

Restore with the last save version:

```
//RESTORE-FILES FILE-NAMES=*ALL, -
//               ARCHIVE-NAME=archive, -
//               SELECT-SAVE-VERSIONS=*BY-ATTRIBUTES, -
//               OPERATION-CONTROL=*PARAMETERS(REPORT=*FULL,OUTPUT=l.restore)
```

With this only those files are restored which are contained in the last save version. Files saved earlier are not taken into account.

```
/SHOW-FILE-ATTRIBUTES FILE-NAME=file*
%       3 :SBZ3:$UHU.FILE2
%       3 :SBZ3:$UHU.FILE4
```

Version backup in SM-environment

Within the example version backup in SM environment is considered.

```
/START-HSMS
//CRA ARCHIVE-NAME=SYSVERS.SMS,ENVIRONMENT=*SYSTEM-MANAGED(CATALOG-ID=SMS),- ———————— (1)
//    ALLOWED-USAGE=*VERSIONBACKUP,-
//    USER-ACCESS=*ALL-USERS(ACCESS=*WRITE),-
//    DIRECTORY-NAME=:SMS:SYSVERS.SMS.DIR,RETENTION-PERIOD=365

//MSP SM-PUBSET-ID=SMS,SYSVERSION=SYSVERS.SMS ———————————————————————————————— (2)

//CREATE-MANAGEMENT-CLASS ENVIRONMENT=*SYSTEM-MANAGED(CATALOG-ID=SMS),- ———————————— (3)
//    MANAGEMENT-CLASS=SOURCES,-
//    MIGRATION-ATTRIBUTES=*PARAMETERS(FROM-S0=*PARAMETERS(UNUSED-DAYS=0)),-
//    BACKUP-ATTRIBUTES=*PARAMETERS(RETENTION-PERIOD=365),-
//    VERSION-BACKUP-ATTR=*PARAMETERS(NUM-OF-BACKUP-VERS=20)
//CREATE-MANAGEMENT-CLASS ENVIRONMENT=*SYSTEM-MANAGED(CATALOG-ID=SMS),-
//    MANAGEMENT-CLASS=LISTINGS,-
//    MIGRATION-ATTRIBUTES=*PARAMETERS(FROM-S0=*PARAMETERS(UNUSED-DAYS=0)),-
//    BACKUP-ATTRIBUTES=*PARAMETERS(RETENTION-PERIOD=365),-
//    VERSION-BACKUP-ATTR=*PARAMETERS(NUM-OF-BACKUP-VERS=10)
//END

/MODIFY-FILE-ATTRIBUTES FILE-NAME=:SMS:*.SRC*,- ———————————————————————————— (4)
/     SUPPORT=*PUBLIC-DISK(MANAGEMENT-CLASS=SOURCES)
/MODIFY-FILE-ATTRIBUTES FILE-NAME=:SMS:*.LST*,-
/     SUPPORT=*PUBLIC-DISK(MANAGEMENT-CLASS=LISTINGS)
```

(1) An HSMS administrator creates public version backup archive in SM environment.

(2) The version backup archive is defined as a system version backup archive for an SM environment.

(3) 2 management classes are created with specifying different values for versioning attribute NUM-OF-BACKUP-VERSION.

(4) The management classes are assigned to some files.

Let us consider the following files and their backup attributes. Some of them will have a mangement class assigned. All the files have a defined value of NUM-OF-BACKUP-VERSIONS in the catalog entries. The values are different from the values defined in the management classes:

```
/sh :sms:*.lst*, back=*y
```

```
%0000000003 :SMS:$TSOS.NB.FILE1.LST
%  ------------------------------ BACKUP       ------------------------------
%  BACK-CLASS = A           SAVED-PAG  = COMPL-FILE  VERSION    = 1
%  MIGRATE    = ALLOWED     MAN-CLASS  = LISTINGS
%  #BACK-VERS = 20
%0000000003 :SMS:$TSOS.NB.FILE2.LST
%  ------------------------------ BACKUP       ------------------------------
%  BACK-CLASS = A           SAVED-PAG  = COMPL-FILE  VERSION    = 1
%  MIGRATE    = ALLOWED     MAN-CLASS  = LISTINGS
%  #BACK-VERS = 20
%0000000003 :SMS:$TSOS.NB.FILE3.LST
%  ------------------------------ BACKUP       ------------------------------
%  BACK-CLASS = A           SAVED-PAG  = COMPL-FILE  VERSION    = 1
%  MIGRATE    = ALLOWED     MAN-CLASS  = LISTINGS
%  #BACK-VERS = 20
```

```
/sh :sms:*.src*, back=*y
```

```
%0000000003 :SMS:$TSOS.NB.FILE1.SRC
%  ------------------------------ BACKUP       ------------------------------
%  BACK-CLASS = A           SAVED-PAG  = COMPL-FILE  VERSION    = 1
%  MIGRATE    = ALLOWED     MAN-CLASS  = SOURCES
%  #BACK-VERS = 15
%0000000003 :SMS:$TSOS.NB.FILE2.SRC
%  ------------------------------ BACKUP       ------------------------------
%  BACK-CLASS = A           SAVED-PAG  = COMPL-FILE  VERSION    = 1
%  MIGRATE    = ALLOWED     MAN-CLASS  = SOURCES
%  #BACK-VERS = 15

%0000000003 :SMS:$TSOS.NB.FILE3.SRC
%  ------------------------------ BACKUP       ------------------------------
%  BACK-CLASS = A           SAVED-PAG  = COMPL-FILE  VERSION    = 1
%  MIGRATE    = ALLOWED     MAN-CLASS  = SOURCES
%  #BACK-VERS = 15
%:SMS:  PUBLIC:      8 FILES RES=        24 FRE=        16 REL=         0 PAGES
```

```
/sh :sms:*.rep*, back=*y
```

```
%0000000003 :SMS:$TSOS.NB.FILE1.REP
%  ------------------------------ BACKUP       ------------------------------
%  BACK-CLASS = A           SAVED-PAG  = COMPL-FILE  VERSION    = 1
%  MIGRATE    = ALLOWED
%  #BACK-VERS =
5
```

Do version backup of the files with //BACKUP-FILE-VERSION statement specifying MANAGEMENT-CLASS as *ANY.

```
/HSMS
//BFV FILE-NAMES=(:SMS:NB.*SRC*,:SMS:NB.*LST*,:SMS:NB.*REP*),-
//ENVIRONMENT=*SYSTEM-MANAGED(CATALOG-ID=SMS),-
//MANAGEMENT-CLASS=*ANY,-
//TO-STORAGE=*S1-STORAGE-LEVEL,-
//OPERATION-CONTROL=*PARAMETERS(REPORT=*FULL,OUTPUT=:SMS:REP1)
```

After request is completed, let see the content of the version backup archive:

```
//SHA ARCHIVE-NAME=SYSVERS.SMS,-
//ENVIRONMENT=*SYSTEM-MANAGED(CATALOG-ID=SMS),-
//SELECT=*FILES(INFORMATION=*VERSION)
```

It can be seen that NUM-OF-BACK-VERS values from management classes have been taken for files assigned to them; while for files, which are not assigned to a management class, the value from their catalog entries.

```
SHOW-ARCHIVE (FILES)                      SHOW-FILE-VERSIONS = DIFFERENT
ENVIRONMENT     = SM(SMS)                 ARCHIVE-NAME = $SYSHSMS.SYSVERS.SMS
CATALOG-ID      = SMS                     SV-NAME  = ANY
USER-ID         = TSOS                    SV-DATE  = INTERVAL  EARLIEST LATEST
FILE-SAVE-STATE = ANY                     EXP-DATE = ANY
INFORMATION     = VERSION
-------------------------------------------------------------------------------
M FILE-NAME                                 V#  O-DATE O-TIME NUM OBS M S D-DATE
  NB.FILE1.LST                              1   190301 150636 10  NO  N Y
  NB.FILE1.REP                              1   190301 150636 5   NO  N Y
  NB.FILE1.SRC                              1   190301 150636 20  NO  N Y
  NB.FILE2.LST                              1   190301 150636 10  NO  N Y
  NB.FILE2.REP                              1   190301 150636 5   NO  N Y
  NB.FILE2.SRC                              1   190301 150636 20  NO  N Y
  NB.FILE3.LST                              1   190301 150636 10  NO  N Y
  NB.FILE3.SRC                              1   190301 150636 20  NO  N Y
```

Here is the report of the version backup run:

```
A***  BACKUP-FILE-VERSIONS           HSMS V12.0        FULL            REPORT  *** 2019-03-
01  15:06:41     PAGE    1
     REQUEST-ENVIRONMENT=SM(SMS)
     REQUEST-NAME=BFV#2917 REQUEST-DATE=2019-03-01 15:06:36 USER-ID=SYSHSMS  REQUEST-
STATE=COMPLETED


 STATEMENT LISTING:

 BFV         FILE-NAMES=(:SMS:NB.*SRC*,  :SMS:NB.*LST*,  :SMS:NB.*REP*),
ENVIRONMENT=*SYSTEM-MANAGED(CATALOG-ID=SMS),
             MANAGEMENT-CLASS = *ANY,  TO-STORAGE = *S1-STORAGE-LEVEL, OPERATION-CONTROL =
*PARAMETERS(REPORT = *FULL,
             OUTPUT=:SMS:NB.REP1)



             SAVE-FILE ATTRIBUTES
             TO-STORAGE                              : S1-STORAGE-LEVEL
```

```
             RETENTION-PERIOD                          : 365

             SAVE-VERSION ATTRIBUTES
             SAVE-VERSION-NAME                         :

             SELECT-FILES PARAMETER
             MODIFIED-FILES  PARTIAL-FILE-SAVE       : NO
             MAX-BACKUP-CLASS                          : STD
  %  HSM0030 REQUEST 'BFV#2917' CREATED IN ENVIRONMENT 'SM(SMS)' WITH DATE '2019-03-01' AND
TIME '15:06:36'
 %  ARC0002 STATEMENT ACCEPTED. ARCHIVE SEQUENCE NUMBER 'A.190301.150636', VERSION '12.0A10'
 %  ARC0802 COMPUTED BLOCK-SIZE VALUE IS '239'
%  ARC0033 ARCHIVE SUBTASK TSN '6AG4' GENERATED
 %  ARC0815 SUBTASK '0' HAS TRANSFERRED '9' PAM PAGES FOR '8' FILES AND '0' JVS IN '0'
SECONDS
A***  BACKUP-FILE-VERSIONS            HSMS V12.0        FULL            REPORT  *** 2019-03-
01  15:06:41     PAGE    4
     REQUEST-ENVIRONMENT=SM(SMS)
     REQUEST-NAME=BFV#2917 REQUEST-DATE=2019-03-01 15:06:36 USER-ID=SYSHSMS  REQUEST-
STATE=COMPLETED   WITH ERRORS
                                       SAVE FILE IDENTIFIER - S.
190301.150636
            SUBSAVE
             NUMBER       VSNS

                0        0:HSMS


                      ***   CATALOG - SMS       USER - TSOS       ***
            FILE/JOB VARIABLE NAME                   LASTPG/   SAVE VERSION  SAVE  INPUT
SUB OUTPUT
                                           VERS       SIZE    IDENTIFIER   TYPE   VSN
SAVE DISK(S)

 NB.FILE1.LST                                 1          1  190301.150636  FULL  SMK.00
0  0:HSMS
 NB.FILE1.REP                                 1          1  190301.150636  FULL  SMK.00
0  0:HSMS
 NB.FILE1.SRC                                 1          1  190301.150636  FULL  SMK.00
0  0:HSMS
 NB.FILE2.LST                                 1          1  190301.150636  FULL  SMK.00
0  0:HSMS
 NB.FILE2.REP                                 1          1  190301.150636  FULL  SMK.00
0  0:HSMS
 NB.FILE2.SRC                                 1          1  190301.150636  FULL  SMK.00
0  0:HSMS
 NB.FILE3.LST                                 1          1  190301.150636  FULL  SMK.00
0  0:HSMS
 NB.FILE3.SRC                                 1          1  190301.150636  FULL  SMK.00
0  0:HSMS

     NUMBER OF FILES=             8 GLOBAL SIZE=           8 START= 2019-03-01 15:06:36
END= 2019-03-01 15:06:41



 ***      E N D   O F               HSMS V12.0        FULL            REPORT  *** 2019-03-
01  15:06:41     ***
```

Let’s change versioning attribute of management class SOURCES.

```
//MODIFY-MANAGEMENT-CLASS ENVIRONMENT=*SYSTEM-MANAGED(CATALOG-ID=SMS),-
//MANAGEMENT-CLASS=SOURCES,-
//VERSION-BACKUP-ATTR=*PARAMETERS(NUM-OF-BACKUP-VERS=32)
```

And repeat the run without updating any of the files.

```
//BFV FILE-NAMES=(:SMS:NB.*SRC*,:SMS:NB.*LST*,:SMS:NB.*REP*),-
//ENVIRONMENT=*SYSTEM-MANAGED(CATALOG-ID=SMS),-
//MANAGEMENT-CLASS=*ANY,-
//TO-STORAGE=*S1-STORAGE-LEVEL,-
//OPERATION-CONTROL=*PARAMETERS(REPORT=*FULL,OUTPUT=:SMS:NB.REP2)
```

As a result no files have been saved. However, files which are assigned to the updated management class, have been affected: the new value of NUM-OF-BACKUP-VERS has been written into the directory.

```
A***  BACKUP-FILE-VERSIONS            HSMS V12.0        FULL            REPORT  *** 2019-03-
01  15:13:51     PAGE    1
     REQUEST-ENVIRONMENT=SM(SMS)
     REQUEST-NAME=BFV#2917 REQUEST-DATE=2019-03-01 15:13:46 USER-ID=SYSHSMS  REQUEST-
STATE=COMPLETED


 STATEMENT LISTING:

 BFV         FILE-NAMES=(:SMS:NB.*SRC*,  :SMS:NB.*LST*,  :SMS:NB.*REP*),
ENVIRONMENT=*SYSTEM-MANAGED(CATALOG-ID=SMS),
             MANAGEMENT-CLASS = *ANY,  TO-STORAGE = *S1-STORAGE-LEVEL, OPERATION-CONTROL =
*PARAMETERS(REPORT = *FULL,
             OUTPUT=:SMS:NB.REP2)



             SAVE-FILE ATTRIBUTES
             TO-STORAGE                          : S1-STORAGE-LEVEL
             RETENTION-PERIOD                    : 365

             SAVE-VERSION ATTRIBUTES
             SAVE-VERSION-NAME                   :

             SELECT-FILES PARAMETER
             MODIFIED-FILES  PARTIAL-FILE-SAVE       : NO
             MAX-BACKUP-CLASS                    : STD
  %  HSM0030 REQUEST 'BFV#2917' CREATED IN ENVIRONMENT 'SM(SMS)' WITH DATE '2019-03-01' AND
TIME '15:13:46'
 %  ARC0002 STATEMENT ACCEPTED. ARCHIVE SEQUENCE NUMBER 'A.190301.151347', VERSION '12.0A10'
 %  ARC0802 COMPUTED BLOCK-SIZE VALUE IS '239'


 %  ARC0033 ARCHIVE SUBTASK TSN '6AG5' GENERATED
 %  ARC0815 SUBTASK '0' HAS TRANSFERRED '0' PAM PAGES FOR '0' FILES AND '0' JVS IN '0'
SECONDS
 %  ARC0037 SPECIFIED FILE NAME ':SMS:$TSOS.NB.*LST*' NOT PROCESSED
 %  ARC0037 SPECIFIED FILE NAME ':SMS:$TSOS.NB.*REP*' NOT PROCESSED
```

```
A***  BACKUP-FILE-VERSIONS            HSMS V12.0        FULL            REPORT  *** 2019-03-
01  15:13:51     PAGE    2
     REQUEST-ENVIRONMENT=SM(SMS)
     REQUEST-NAME=BFV#2917 REQUEST-DATE=2019-03-01 15:13:46 USER-ID=SYSHSMS  REQUEST-
STATE=COMPLETED   WITH ERRORS
                        ***   CATALOG - SMS       USER - TSOS       ***
            FILE/JOB VARIABLE NAME                   LASTPG/   SAVE VERSION  SAVE  INPUT
SUB OUTPUT
                                              VERS       SIZE    IDENTIFIER   TYPE   VSN
SAVE DISK(S)

 NB.FILE1.SRC                                    1          1  190301.151347        SMK.00
0  DIR-UPD: #BACK-VERS
 NB.FILE2.SRC                                    1          1  190301.151347        SMK.00
0  DIR-UPD: #BACK-VERS
 NB.FILE3.SRC                                    1          1  190301.151347        SMK.00
0  DIR-UPD: #BACK-VERS

     NUMBER OF FILES=              3 GLOBAL SIZE=           3 START= 2019-03-01 15:13:47
END= 2019-03-01 15:13:51



 ***      E N D   O F                HSMS V12.0        FULL            REPORT  *** 2019-03-
01  15:13:51     ***
```

Let’s check the content of the version backup archive:

```
//SHA ARCHIVE-NAME=SYSVERS.SMS,-
//ENVIRONMENT=*SYSTEM-MANAGED(CATALOG-ID=SMS),-
//SELECT=*FILES(INFORMATION=*VERSION)
```

Files assigned to modified management class now have new values for NUM-OF-BACK-VERS (see the column ‘NUM’):

```
SHOW-ARCHIVE (FILES)                     SHOW-FILE-VERSIONS = DIFFERENT
ENVIRONMENT     = SM(SMS)                ARCHIVE-NAME = $SYSHSMS.SYSVERS.SMS
CATALOG-ID      = SMS                    SV-NAME  = ANY
USER-ID         = TSOS                   SV-DATE  = INTERVAL  EARLIEST LATEST
FILE-SAVE-STATE = ANY                    EXP-DATE = ANY
INFORMATION     = VERSION
-------------------------------------------------------------------------------
M FILE-NAME                                V#  O-DATE O-TIME NUM OBS M S D-DATE
  NB.FILE1.LST                             1   190301 150636 10  NO  N Y
  NB.FILE1.REP                             1   190301 150636 5   NO  N Y
  NB.FILE1.SRC                             1   190301 150636 32  NO  N Y
  NB.FILE2.LST                             1   190301 150636 10  NO  N Y
  NB.FILE2.REP                             1   190301 150636 5   NO  N Y
  NB.FILE2.SRC                             1   190301 150636 32  NO  N Y
  NB.FILE3.LST                             1   190301 150636 10  NO  N Y
  NB.FILE3.SRC                             1   190301 150636 32  NO  N Y
```

4.2 Archival

Archival is the long-term saving of files and job variables that are no longer required online at the processing level to storage level S1 or S2 on shared disks or Net-Storage.
Archival saves storage space at the processing level because the files and job variables can be deleted after archival. Archival is also used for documentation purposes, e.g. if legal regulations prescribe that data must be retained for specific periods of time.
It is possible to define file expiration dates for the files saved in long-term archives independently of the volume retention period.

The archival function is available to all users if the archive is created accordingly.

4.2.1 Archiving BS2000 files

BS2000 files are archived by means of the HSMS statement ARCHIVE-FILES.

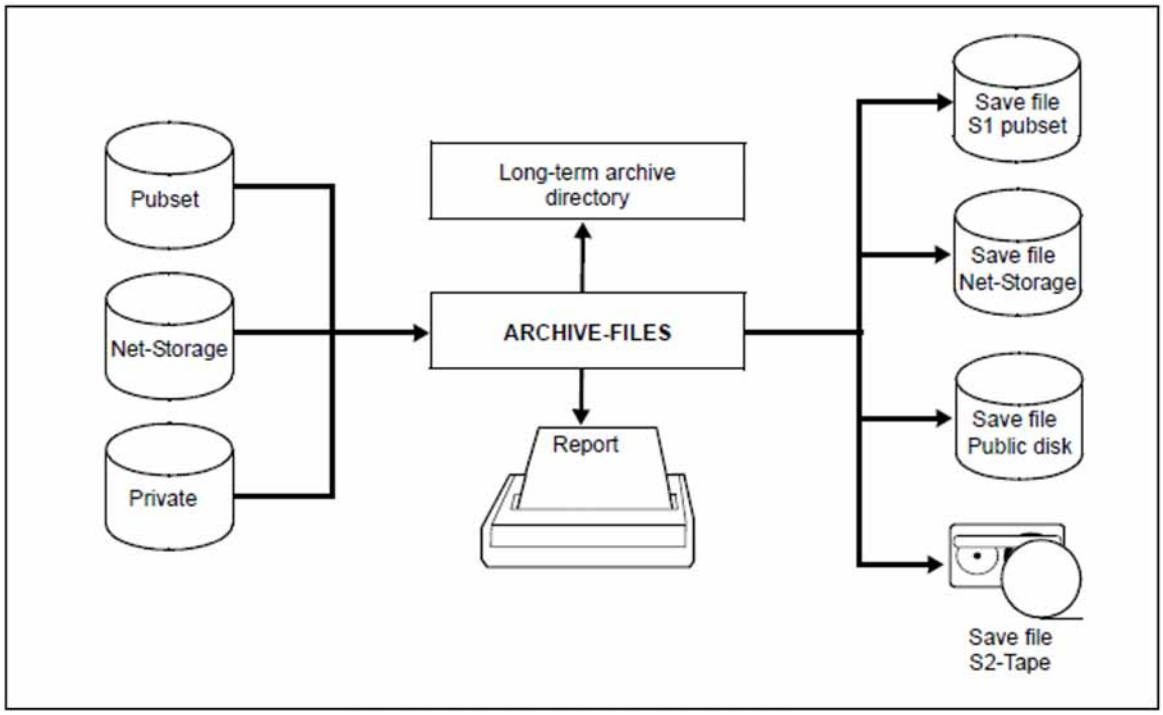


Figure 11: Archival using ARCHIVE-FILES

A global long-term archive that is defined in the SF environment can contain either SF files or SM files. A long-term archive that is local for the SM pubset can only contain SM files.

If the HSMS statement ARCHIVE-FILES refers to an archive that is defined in the central HSMS parameters, the files to be processed can be located either in the SF environment or in the SM environment. If the HSMS statement ARCHIVE-FILES refers to a local
SM pubset archive, the files to be edited can only be located in this SM environment.
A local SM pubset archive can only be stored at storage level S1 or S2. Other specifications in the TO-STORAGE operand are rejected.

The ARCHIVE-FILES statement offers the operands FILE-NAMES, EXCEPT-FILE-NAMES and SELECT-FILES for the selection of BS2000 files to be archived.

It is, of course, possible to specify a list of file names or to select BS2000 files in screen masks (see section "Selection of BS2000 files, job variables and node files").

Migrated BS2000 files can be archived as well. The backup includes the catalog entries from the catalog and the data from the migration level. BS2000 files which have already been migrated for a certain period (user-definable) can be archived selectively using the SELECT-FILES operand in the ARCHIVE-FILES statement. The migrated files are copied directly from the migration archive to the long-term archive.

You can also use the HSMS statement ARCHIVE-FILES to archive files of which you are the co-owner (see the “SECOS” manual [16]).

Job variables can also be entered, using the HSMS statement ARCHIVE-FILES with the operand JV-NAMES. The job variables are saved in exactly the same way as with the BACKUP-FILES.

The archive owner can use the SAVE-FILE operand for archiving to define whether

- the BS2000 files to be archived are to be written to the archive’s standard save file (=*STD)
- a new save file is to be created for this archival operation (=*NEW)
- an existing save file is to be continued (=*CONTINUE).

Continuation is only supported for save files on storage level S2.

If they are not the archive owner, nonprivileged users can only perform archiving by using the SAVE-FILE=*STD operand.

HSMS also enables large files (>= 32 GB) to be archived. No particular provisions are required to do this. However, when restoring large files, bear in mind their special characteristics (see section "Restoring large files (> 32 GB)".

Encrypted files are saved in encrypted format and restored again in the original encrypted format. The associated crypto password need not be specified for saving or restoring. Other accesses to the restored file require the crypto password to be specified again.

4.2.1.1 Long-term system archive

Archived data is managed in long-term archives. These are created by the HSMS administrator as public archives with write access (USER-ACCESS=*ALL-USERS and ACCESS=*WRITE).

Long-term system archives for BS2000 files are normally addressed by their symbolic name SYSARCHIVE. This is the default long-term archive to which HSMS statements refer unless another archive is explicitly specified.

4.2.1.2 File expiration date

The FILE-EXPIRATION-DATE operand can be used in an archival request to define a file expiration date for the files to be archived; this is a logical retention period that may differ from the physical retention period defined for the volume containing the save version. The HSMS administrator can define for public long-term archives that the logical retention period for individual files may be longer than the physical retention period for the volumes. In this case, the HSMS administrator must take appropriate administrative measures to protect a volume against overwriting once the physical retention period has expired:

- This can be done, for instance, by reorganizing the long-term archives and by reshuffling the volumes before the retention period expires. Reshuffling is the copying of save versions to a volume with a higher expiration date. This can be restricted to those save versions whose file expiration date has not yet been reached.
- Another possibility is to use the SAVE-FILE-RETPD-UPDATE operand. This operand allows the retention period of a save file to be automatically extended while a new save version is created or the file expiration date is changed. This change also affects the MAREN catalog.

The function SAVE-FILES=*RETENTION-PERIOD(...) in the HSMS statement MODIFY-ARCHIVE can also be used to extend the retention period of a save file. With save files on magnetic tape cartridge, this extension also affects the MAREN catalog entry of the volume and with save files on disk it also affects the catalog entry.

The logical expiration date for a save version can be defined in two ways:

- Implicitly without specifying the FILE-EXPIRATION-DATE operand or by specifying FILE-EXPIRATION-DATE=*STD

  In the latter case the expiration date is calculated as follows:

  (Creation-Date + Retention-Period [+ Continuation-Period])
  (Creation-Date + Retention-Period [+ Continuation-Period])
- By explicitly specifying a value (not equal to *STD) in the FILE-EXPIRATION-DATE operand.

If the expiration date of an archived file/job variable (see catalogue entry for the EXPIR-DATE file) is earlier than the defined expiration date for the save version, a warning is issued and the archival request is assigned the substatus COMPLETED WITH WARNINGS.

4.2.1.3 Additional information

Since archived data may sometimes have to be retained for a fairly long time, it may be useful to store information about the data together with the save data. During archival, a brief descriptive text may be entered (DESCRIPTOR operand) as well as a more extensive comment (USER-INFORMATION operand), describing the save version and containing information about the archived files, respectively.
Both texts can be output by means of SHOW-ARCHIVE. With save files on magnetic tape cartridge, this extension also affects the MAREN catalog, and with save files on disk it also affects the catalog entry.

Additional information can be modified at a later date, using:

```
//MODIFY-ARCHIVE ...,SAVE-VERSIONS=*MODIFY-USER-RECORD(...)
```

The date of the original backup remains unchanged if a save file is copied. This original save date can thus be used to select a save version in the COPY-SAVE-FILE and RESTORE-SAVE-FILE statements.

4.2.1.4 Archival options

The archive directory that was brought up to date upon archival can also be written onto the archival volume at the end of archival. The archival volume then contains in addition to the saved data a directory of the data contained on it.

Saving of the archive directory is initiated by specifying SAVE-DIRECTORY=*YES. It is not saved by default.

The HSMS administrator is permitted to save the archive directory, as is also the archive owner and the co-owner of the archive directory.

The following applies to saving the archive directory:

- If no data was saved with the request, the directory is not saved either.
- Only a full backup of the archive directory is carried out, even if incremental backups are implemented.
- The archive directory is not distributed to different volumes. Saving of the archive directory and the volume used for this are displayed in the report.

When the archive directory has been saved it can be restored from the volume in the event of an emergency (if it no longer exists or is not readable on S0). To do this you use the IMPORT-FILES statement with FILE-NAMES=*DIRECTORY specified and the volume noted in the report on the backup.

Furthermore the attributes of an archive are also stored in the archive directory. If the archive definition has been lost the archive can be restored from the archive directory using the CREATE-ARCHIVE statement:

```
//CREATE-ARCHIVE ARCHIVE-NAME=...,
   DIRECTORY-NAME=<directory>(NEW-DIRECTORY=*NO(RECONSTRUCT-ARCHIVE=*YES))
```

To permit individual elements to be restored from the long-term backup of large PLAM libraries (with RESTORE-LIBRARY-ELEMENTS) the SAVE-PLAM-INFO=*YES option must be used when saving. This is the case when the long-term archive has the attribute SAVE-PLAM-INFO=*YES when it is saved (see CREATE-ARCHIVE or MODIFY-ARCHIVE-ATTRIBUTES).

Special aspects of SAM node files

There are two possible ways to access a SAM node file from BS2000:

- one is to maintain the SAM structures in the BS2000 application by adding them with Net-Client during the write process when the files are migrated from the Net-Server or
- without conversion, i.e. without adding SAM structures to the data blocks and without code conversion to EBCDIC (this is called RAW mode).

The access type is controlled via the SAVE-SAM-STRUCTURE=*NO operand of the SAVE-OPTIONS option. Please also see section “Special aspects of SAM node files” in chapter "Save options".

4.2.1.5 Automatic duplication to a shadow archive

The save file of a long-term archive for BS2000 files can be automatically duplicated if it is stored at storage level S2 and the long-term archive concerned was assigned a shadow archive. The automatic duplication mechanism must also be activated in the HSMS statement ARCHIVE-FILES. Here the value *ALLOWED must be specified for SHADOW-COPY in the OPERATION-CONTROL operand of the HSMS statement ARCHIVE-FILES (default setting).

The save file is copied into the assigned shadow archive at the end of the archival run. The copy in the shadow archive is given the same save version ID and save file ID as the original, i.e. it receives the same time stamp.

If the files are to be deleted after archival (DELETE-FILES=*YES), they are not deleted until the save file has been copied.

If a save file in a long-term archive is to be extended, the automatic duplication mechanism also attempts to extend the save file with the same SFID in the assigned shadow archive. If this save file does not exist, duplication (and possible deletion of the files) is not carried out and message HSM0318 is issued.

4.2.1.6 Notes on using standard save files

Standard save files are only stored at storage level S2.

If a value other than *S2-STORAGE-LEVEL is specified in the TO-STORAGE operand when archiving using SAVE-FILE=*STD, a new save file is created in accordance with the TO-STORAGE specification. If a standard save file is saved regularly in accordance with the archive attributes, it is continued when subsequent archive operations take place with TOSTORAGE=*S2-STORAGE-LEVEL.

4.2.2 Restoring archived BS2000 files

BS2000 files can be restored after archival by means of the HSMS statement RESTORE-FILES.

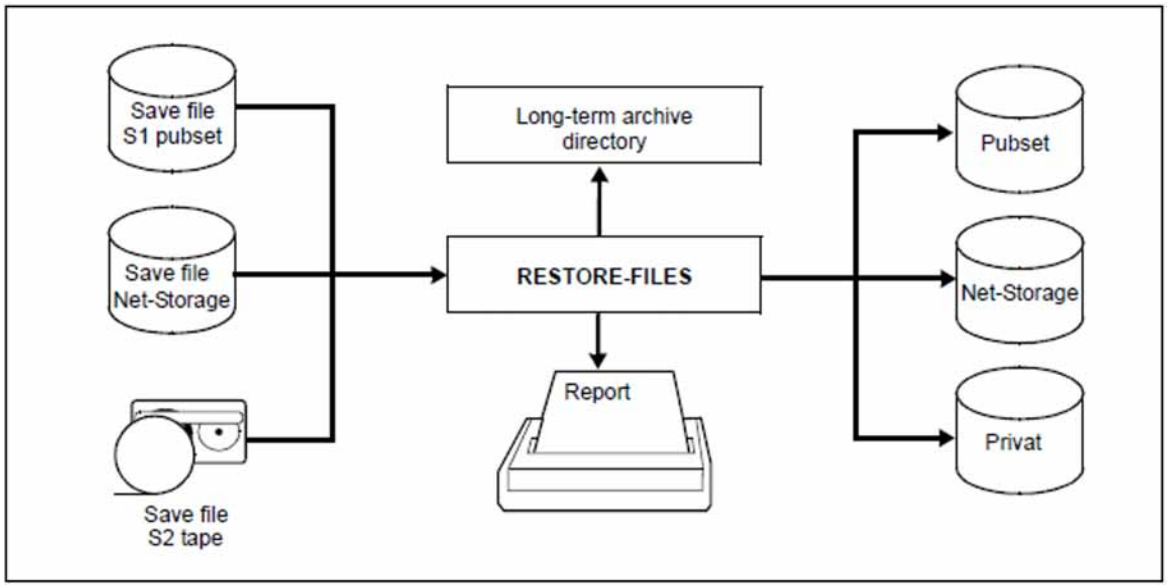


Figure 12: Restore using RESTORE-FILES

When restoring from long-term archives, objects from SF and SM pubsets may be included in the same run because the archiving function is not restricted to a specific environment.

Nonprivileged users are authorized to restore data belonging to their own user ID from the default system archives or other archives to which they have access. In addition, they are authorized to restore data belonging to other user IDs from the default system archives or other archives to which they have access, provided that they are co-owners of this data. They can restore the data to their own or to another user ID. For more detailed information on co-ownership, refer to the “SECOS” manual [16].
A restricted form of the HSMS statement RESTORE-FILES is available to nonprivileged users for this purpose.

In the event of restoration from a long-term archive, the files are created with the DMS default catalog attributes. In particular files and job variables are assigned a new creation date. The passwords, however, are transferred from the archive. Optionally the files can be restored with their original catalog attributes (in particular with the original creation date) by specifying DATE-AND-PROTECTION= *ORIGINAL-ATTRIBUTES.

4.2.2.1 Restoring large files (> 32 GB)

Large files (>= 32 GB) can only be restored on pubsets with the attributes LARGE-OBJECTS=*YES and LARGE-FILES-ALLOWED=*YES.

If a file >= 32 GB is read into a pubset with the attribute LARGE-FILES-ALLOWED=*NO, this will always cause a error. This also applies, even if the file has been selected implicitly, for example, when a number of files have been selected using a selection method such as partial selection or a wildcard selection.

Backups from older BS2000 versions can also be restored in higher BS2000 versions without restrictions, this applies to standard pubsets and pubsets for which the attribute has been set to LARGE-OBJECTS=*YES.

4.2.2.2 Selecting save versions

The selection of save versions for *backed up* files is described in "Selecting save versions" (in "Restoring backed up BS2000 files and job variables"); the same procedure applies to *archived* files. Users should bear in mind, however, that the specification of SELECT-SAVE-VERSION=*LATEST causes only those files to be restored which were actually saved in the last save version. If an error caused only the catalog entry of a file (CNS, OPER) to be saved, none of the previous save versions will be searched for that file.

4.2.2.3 Restoring from a shadow archive

If a long-term archive was assigned a shadow archive, the shadow archive can be used for restoration. This is helpful if the original long-term archive has been destroyed or the associated volumes cannot be accessed.

4.2.2.4 Restoring elements from a PLAM library

If in a long-term backup not only the library files are saved via the elements but also the metainformation, library elements can also be restored selectively from the long-term backup (as with backup archives) using the RESTORE-LIBRARY-ELEMENTS statement. In the case of a long-term backup the metainformation is saved only if the save option SAVE-PLAM-INFO=*YES is set for the long-term archive.

If a PLAM library is to be saved with the metadata about the library elements, the library must be located at S0 level. When PLAM libraries are migrated this metadata is not saved.

The library must be accessible on S0, in other words may not be migrated. The elements can be overwritten and renamed as required.

The backup version of the library containing the specified elements must be known.

The library elements can also be restored to a target library which is not the source library.

The element names of a PLAM library saved with PLAM-INFO=*YES are only contained in the PLAM information in the save file but not in the archive directory. The element names of a PLAM library saved in this way can be listed using the ELEMENTS=*LIST-ALL-TO-REPORT operand in the RESTORE-LIBRARY-ELEMENTS statement.

4.2.2.5 Restoring Net-Storage files

If files on Net-Storage are also saved in a long-term backup, restoration of the files can be restricted to Net-Storage (STORAGE-TYPE operand).

Net-Storage files of the BS2000 or node file type, whose SAM structure has been saved, can be restored to different volume types. During this process, their file type can be changed. In contrast, net storage files of the file type node file, whose SAM structure has not been saved, can only be restored on a Net-Storage volume and with their original file type node file.

4.2.3 Archiving node files of the BS2000-UFS and remote nodes S0

Node files of the BS2000-UFS and remote nodes S0 can be archived with the HSMS statement ARCHIVE-NODE-FILES. Individual files can be selected via their names (using wildcards) and via the SELECTION-BOUNDARY operand. The archival may optionally be restricted to the explicitly specified files or, if archiving a directory, include all its subdirectories and subordinated files in the file system or file tree (see section "Selection of node files of the BS2000-UFS").

The scope of archiving is shown by the table in "Scope of a backup".

Nonprivileged users can only archive node files residing on the local BS2000-UFS to which they have read access. These comprise all node files for which nonprivileged users have read authorization and for whose parent directories they have execute authorization.

The HSMS administrator can archive all node files residing on the local BS2000-UFS or any remote file system.

4.2.3.1 Long-term system archive

Archived data is managed in long-term archives. These are created by the HSMS administrator as public archives with write access (USER-ACCESS=*ALL-USERS and ACCESS=*WRITE).

Long-term system archives for node files are normally addressed by their symbolic name SYSNODEARCHIVE.

4.2.3.2 File expiration date

The FILE-EXPIRATION-DATE operand can be used in an archival request to define a file expiration date for the files to be archived; this is a logical retention period that may differ from the physical retention period defined for the volume containing the save version. The HSMS administrator can define for public long-term archives that the logical retention period for individual files may be longer than the physical retention period for the volumes. In this case, the HSMS administrator must take appropriate administrative measures to protect a volume against overwriting once the physical retention period has expired. One possibility, for instance, is to reshuffle the volumes before the retention period expires. Reshuffling is the copying of save versions to a volume with a higher expiration date. This can be restricted to those save versions whose file expiration date has not yet been reached.

4.2.3.3 Additional information

Since archived data may sometimes have to be retained for a fairly long time, it may be useful to store information about the data together with the save data. During archival, a brief descriptive text may be entered (DESCRIPTOR operand) as well as a more extensive comment (USER-INFORMATION operand), describing the save version and containing information about the archived files, respectively. Both texts can be output by means of SHOW-ARCHIVE.

Additional information can be modified at a later date, using:

```
//MODIFY-ARCHIVE ...,SAVE-VERSIONS=*MODIFY-USER-RECORD(...)
```

The date of the original backup remains unchanged if a save file is copied. This original save date can thus be used to select a save version in the COPY-SAVE-FILE and RESTORE-SAVE-FILE statements.

4.2.3.4 Automatic duplication to a shadow archive

The save file of a long-term archive for node files of BS2000-UFS can be automatically duplicated if the long-term archive concerned was assigned a shadow archive. For HSMS administrators the automatic duplication mechanism must also be activated. I.e. in the OPERATION-CONTROL operand of the HSMS statement ARCHIVE-NODE-FILES, the value *ALLOWED must be specified for SHADOW-COPY (default setting).

The save file is copied into the assigned shadow archive at the end of the archival run. The copy in the shadow archive is given the same save version ID and save file ID as the original, i.e. it receives the same time stamp.

If a save file in a long-term archive is to be extended, the automatic duplication mechanism also attempts to extend the save file with the same SFID in the assigned shadow archive.

If the tape copy is placed in a specially protected room immediately after the shadow archive has been backed up a new save file must always be generated in the shadow archive, also when the save file in the main archive is updated. This mode is set using the archive attribute SHADOW-COPY=*ALLOWED-AND-NEW-SFID.

4.2.4 Restoring archived node files

The HSMS statement RESTORE-NODE-FILES enables you to restore archived node files to BS2000-UFS or remote nodes S0.

Nonprivileged users are authorized to restore data from the local BS2000-UFS. The following restrictions apply:

- From a system archive, they can restore their own node files.
- From a private archive, each user can restore any data that he/she archived earlier.

The node files are restored with the same attributes they had at the time of archival (ownership and access rights, time stamp,...).

4.2.4.1 Selecting save versions

The selection of save versions for *backed up* files is described on "Selecting save versions" (subchapter of "Restoring backed up BS2000 files and job variables"); the same procedure applies to *archived* files. Users should bear in mind, however, that the specification of SELECT-SAVE-VERSION=*LATEST causes only those files to be restored which were actually saved in the last save version. If an error caused only the catalog entry of a file (CNS, OPER) to be saved, none of the previous save versions will be searched for that file.

4.2.4.2 Restoring from a shadow archive

If a long-term archive was assigned a shadow archive, the shadow archive can be used for restoration. This is helpful if the original long-term archive has been destroyed or the associated volumes cannot be accessed.

4.2.5 Example of long-term archival

A long-term archive is created:

```
//CREATE-ARCHIVE ARCHIVE-NAME=ARC.ARF,ALLOWED-USAGE=*ARCHIVAL,
                 DIRECTORY-NAME=DIR.ARF,RETENTION-PERIOD=9
```

Some files are archived, and the archive directory is written last onto the save volume:

```
//ARCHIVE-FILES FILE-NAMES=U.,SAVE-DIRECTORY=*YES,ARCHIVE-NAME=ARC.ARF,
               OPERATION-CONTROL=*PARAMETERS(REPORT=*FULL,OUTPUT=REPORT.FULL)
```

Extract of the log file edited for printing:

```
SAVE FILE IDENTIFIER - S.160516.155430     SAVE-VERSION-DATE=16-05-16  SAVE-VERSION-TIME=15:
54:32

%  ARC0820 DIRECTORY ':A:$USERA.DIR.ARF' SAVED ON VSN 'TAPE01'

                      ***   CATALOG - A         USER - USERA      ***
          FILE/JOB VARIABLE NAME                 LASTPG/  SAVE   INPUT DEV  SUB  OUTPUT
                                          VERS       SIZE  TYPE    VSN  TYP SAVE  VSN(S)

DIR.ARF                                      1          6  FULL  PUBA00  D     0  TAPE01
U.1                                          1          1  FULL  PUBA00  D     0  TAPE01
U.2                                          1          1  FULL  PUBA00  D     0  TAPE01
U.3                                          1          1  FULL  PUBA00  D     0  TAPE01

    ***  DIRECTORY NAME IS :A:$USERA.DIR.ARF

NUMBER OF FILES=           4 GLOBAL SIZE=        9 START= 2016-05-16 15:54:18 END= 2016-05-
16 15:55:30
```

The files are subsequently deleted.

Restoring the files from long-term archival

If the long-term archive and the archive directory are deleted (or are lost), the archive directory must be imported before the files are restored:

```
//IMPORT-FILES FILE-NAMES=*DIRECTORY,SAVE-FILE=*BY-VOLUME(VOLUMES=TAPE01),
               OPERATION-CONTROL=*PARAMETERS(REPORT=*FULL,OUTPUT=REPORT.FULL)
```

Extract of the log file edited for printing:

```
                            SAVE FILE IDENTIFIER - S.160516.155430
 SUBSAVE
  NUMBER       VSNS

     0        TAPE01

            ***   CATALOG - A         USER - USERA      ***
 FILE/JOB VARIABLE NAME                   LASTPG/   SAVE VERSION  SAVE  INPUT   SUB OUTPUT
                                  VERS       SIZE    IDENTIFIER   TYPE   VSN   SAVE DISK(S)


DIR.ARF                              1          6  160516.155432  FULL  TAPE01   0  PUBA00
```

The archive can now be created again with the archive directory:

```
//CREATE-ARCHIVE ARCHIVE-NAME=ARC.ARF,ALLOWED-USAGE=*ARCHIVAL,
                 DIRECTORY-NAME=DIR.ARF(
                           NEW-DIRECTORY=*NO(RECONSTRUCT-ARCHIVE=*YES))
```

The files can then be restored:

```
//RESTORE-FILES FILE-NAMES=*ALL,ARCHIVE-NAME=ARC.ARF,
               OPERATION-CONTROL=*PARAMETERS(REPORT=*FULL,OUTPUT=REPORT.FULL)
```

4.3 Migration of BS2000 files

Migration is restricted to BS2000 files on shared volumes. It is the process of moving inactive data from the processing level to a background level. The migrated data is moved to a migration archive and the storage space it occupied at the processing level is released. However, migrated files retain their catalog entry at the processing level and can continue to be addressed via this entry. In particular, the data is recalled implicitly (i.e. automatically) to the processing level when an access attempt is made by the BS2000 Data Management System.

The migration function is available to all users if the archive is created accordingly. However, the HSMS administrator can restrict its usage.

Migration can be carried out only to pubsets that have been taken under HSMS management.

Files which are stored on a Net-Storage volume assigned to the pubset cannot be migrated.

Files which are stored on a disk storage system and are mirrored remotely in another disk storage system (e.g. Symmetrix disks mirrored with SRDF) should not be migrated if the remote host is unable to access the S2 processing level.

You can also use the HSMS statement MIGRATE-FILES to migrate files of which you are the co-owner (see the “SECOS” manual [16]).

When a MIGRATE-FILES statement is issued on a host that manages both SF and
SM pubsets, only objects that are located either on an SF or on and SM pubset can be migrated at any one time. Furthermore, when migrating objects to SM pubsets, only objects of an SM pubsets can be processed in each run.

Migration control and migration archive management by the HSMS administrator are described in section "Migration control and management of migration archives".

An attempt to access a migrated file causes a DMS error

- if HSMS is not available on the system
- if an implicit recall is prohibited
- if an implicit recall failed.

4.3.1 Migrate files

Files are migrated to a background level by means of the HSMS statement MIGRATE-FILES.

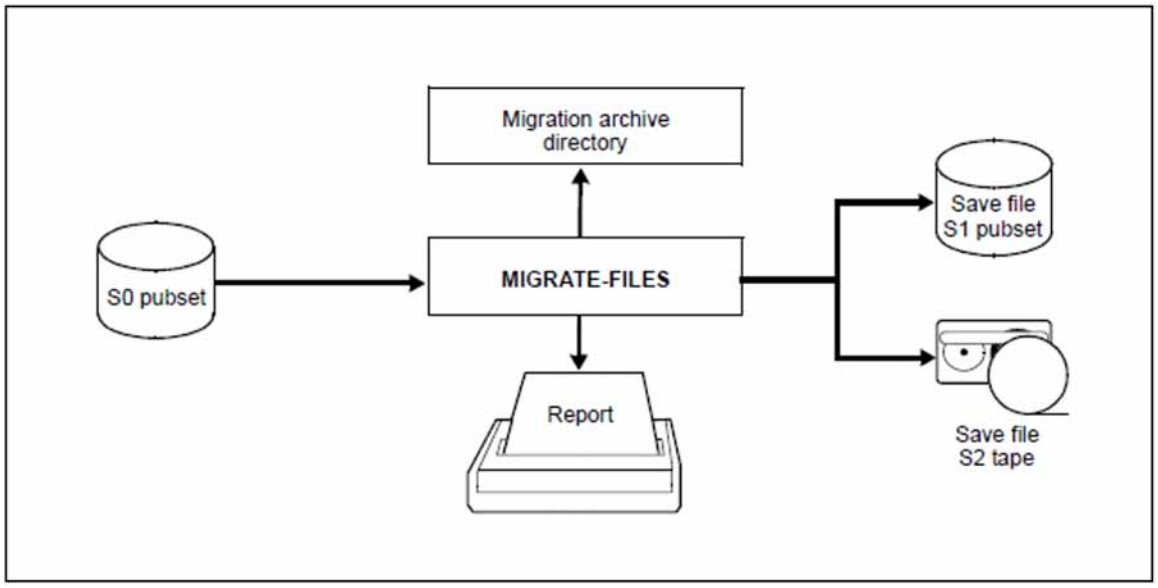


Figure 13: Migration using MIGRATE-FILES (nonprivileged user)

Only files cataloged on an S0 pubset having an assigned migration archive can be migrated.

If migration is allowed only for files that have already previously been backed up (see “HSMS Vol. 2” [1], BACKUP-MANDATORY operand in the HSMS statements CREATE-SM-PUBSET-PARAMETERS and MODIFY-HSMS-PARAMETERS), before initiating migration make sure that the current status of the file has been saved to a backup archive.

The files can be migrated only to the standard save file of a default system archive. It is therefore not possible to specify an archive for migration.

Via the operand TO-STORAGE, the user can define whether the files are to be migrated to S1 or S2 (assuming the HSMS administrator has permitted migration to S1).

If overwriting with binary zeros has been specified for deletion of the migrated files, the released storage space is also overwritten in the case of migration.

The MIGRATE-FILES statement offers various options for selecting the files to be migrated, in particular via the length of time during which no access has been made to the files (UNUSED-DAYS=) and via the minimum and maximum file size (MINIMUM-SIZE/ MAXIMUM-SIZE).

The files to be migrated can optionally be selected according to a predefinable minimum number of extents. The operand, MINIMUM-EXTENTS, which is reserved for HSMS administrators, performs this.

Of course it is also possible to specify a list of file names and to select files in a dialog (see section "Selection of BS2000 files, job variables and node files").

4.3.1.1 Migrating certain quantities

The HSMS administrator and nonprivileged users can use a statement that will migrate files until the storage space made available by this process reaches a specifiable number of PAM pages:

```
//MIGRATE-FILES ...(RELEASE-PAGES=<zahl>)
```

HSMS migrates files in descending order according to inactive date, i.e. it first migrates the files which have not been accessed for the longest period of time (LAST ACCESS DATE) and continues to do so until the specified number of free pages is reached. In terms of the files, this is based on the number of occupied pages.

Even in this case, only files from the set selected via FILE-NAMES (plus additional operands) are migrated. By default (RELEASE-PAGES=*MAXIMUM), all files of this selected set are migrated.

4.3.1.2 Quick migration without tape processing (quick migration)

The HSMS administrator and nonprivileged users migrate all files which can be migrated quickly, i.e. without tape processing, with one statement.

```
//MIGRATE-FILES ...(MIGRATION-INFO=*REMIGRATION)
```

From the set of files selected HSMS migrates only those files which have not been modified since the last recall and for which the backup file still exists. These files are immediately, i.e. without tape access, assigned only the reference to their existing backup file (S1 or S2).

When quick migration takes place, the storage space occupied is immediately released and no additional storage space is required on S1 or S2 for the migration.

4.3.1.3 File migration inhibit

The file owner can define via the catalog entry whether or not the file is to be exempted from migration. This is done via the MIGRATE operand of the CREATE-FILE or MODIFY-FILE-ATTRIBUTES command:

- MIGRATE=*ALLOWED allows a file to be migrated (default)
- MIGRATE=*INHIBITED prohibits the migration (Exception: see "Files typically exempted from migration").

The HSMS administrator can, however, state that MIGRATE= *INHIBITED files can be migrated.

User IDs that have the right to make physical allocations also have the MIGRATE=*FORBIDDEN option. This inhibits migration completely. A note is entered in the catalog entry of the file and can be output using the SHOW-FILE-ATTRIBUTES command.

4.3.1.4 Files typically exempted from migration

The following types of files cannot be migrated to a background level:

- files not yet contained in a save file (HSMS or ARCHIVE) (except if the file has save class “E”), if the BACKUP-MANDATORY operand in the MODIFY-HSMS-PARAMETERS statement has the value *YES
- files added to the except file (see section "Except file")
- open files
- temporary files
- files without a space allocation
- files under the user ID SYSHSMS
- Files on Net-Storage
- files on private volumes
- file generation groups (FGGs)
- system files such as the catalog (but not IPLR or the startup file)
- files to be repaired.
- files cataloged with:
  - AVAILABILITY=*HIGH
  - MIGRATION=* INHIBITED and FILE-INHIBIT=* RESPECTED in the MIGRATE-FILES statement or in the last MODIFY-SM-PUBSET-PARAMETERS statement of the SM pubsets in question or in the last MODIF-HSMS-PARAMETERS statement.
  - MIGRATION=*FORBIDDEN

If a user includes any file in an HSMS statement that is exempted from migration for any of the above-mentioned reasons, the system will issue appropriate warnings relating to those files. If, however, *OWN or *ALL is specified, no such warnings will be displayed.

Example:

```
% HSM0550 WARNING: SOME FILES NOT MIGRATED (REASON 'NET-STORAGE')
```

4.3.2 Migration by the HSMS administrator

Migration is always particularly urgent when space becomes “tight” on a pubset, i.e. when approaching or reaching certain saturation levels. HSMS offers the following functions to support the HSMS administrator:

- Comprehensive options for informing the HSMS administrator of pubset usage (see next section).
- File migration until a certain amount of storage space has been released (see section "Migrating certain quantities"); for the HSMS administrator, this can be done according to system or pubset.
- By generating repeat jobs which are controlled by job variables supplied with values by HSMS the administrator can ensure that an automatic response takes place when saturation levels are reached (see next section).

Save files can be migrated from S1 to S2, to a more “cost-effective” storage level.

4.3.2.1 Information options

HSMS supports the HSMS administrator in the migration of files by providing various information options via the SHOW-PUBSET-USAGE statement.

- INFORMATION=*SUMMARY provides a summary of the total pubset capacity under HSMS management and the percentage of used, free and migrated pages.
- INFORMATION=*UNUSED-DAYS provides a summary of inactive files in the system (according to pubset), i.e. files which are ready for migration because they have not been accessed for a considerable period of time.

For details on these information options, refer to the HSMS statement SHOW-PUBSET-USAGE in “HSMS Vol. 2”[1].

4.3.2.2 Migrating files automatically

At regular intervals, HSMS checks whether a saturation level has been reached on a pubset. Depending on this check, HSMS supplies the $SYSHSMS.SYS.HSM.MIGRATE.<pubset-id> job variable, on a pubset-specific basis, with values for all pubsets entered in the control file.
HSMS also carries out similar checks for all volume sets belonging to an SM pubset under HSMS control. HSMS then delivers the job variables $SYSHSMS.SYS.HSM.MIGRATE.<vol-id> which are located on the corresponding SM pubset.

By interrogating this job variable in enter jobs or programs, the HSMS administrator can provide for an automatic response to any saturation states and thus migrate enough files to eliminate the saturation levels on the SF pubsets or the volume sets of the SM pubset. The number of necessary pages depends on the saturation levels set and is likewise passed in the job variable. The quantity to be migrated can then be specified in MIGRATE-FILES, using the RELEASE-PAGES operand.

Unless they already exist, the job variables are automatically generated by HSMS at startup. The job variables are stored as non-shareable under the SYSHSMS ID (on the home pubset for an SM pubset and on the SM pubset for the included volume sets). The attributes of the job variables must not be modified.

If HSMS is not loaded or loaded only in definition/information mode, the job variables contain X'FF...' for the first 40 bytes. In the other modes, the job variables are updated on startup for SF pubsets or on import of each such SM pubset and every 10 minutes thereafter. The saturation level reached on a given pubset is entered in the job variable for the pubset.

The contents of the job variable have the following meaning:

Position	Length	Meaning
1	1	Pubset status: A: accessible N: catalog ID not available I: inaccessible S: shared slave R: remote
2	1	Fill character
3	1	Pubset saturation level
4	2	Fill character
6	10	Number of occupied PAM pages
16	5	Fill character
21	10	Total capacity of the pubset in PAM pages
31	5	Fill character
36	10	Number of PAM pages exceeding the saturation level indicated, if one has been reached
46	3	Fill character

49	12	Date at which the job variable was set to the current value
61	5	Fill character
66	8	Time at which the job variable was set to the current value
74	2	Fill character
76	181	Fill character

Note

For SM pubsets, this mechanism is available at volume set level only and not for the entire SM pubset.

This information can be used to control the specific migration operation.

Examples of an enter job and a program can be found at the end of this section, starting on "Migration examples".

4.3.3 Migrated files

After migration, a file only has a catalog entry at the processing level: it does not occupy any storage space at that location. The catalog entry indicates where the data in this file is located, i.e. to which background level – S1 or S2 – it has been migrated.

Depending on the background level to which the file has been migrated, PUB/S1 or PUB/S2 is output for SUPPORT. NONE is entered for EXTENTS.
The file size and last-page pointer values remain unchanged by migration. The storage level to which the file has been migrated is output under STOR-LEVEL.

The following command serves to obtain a list of all migrated files:

```
/SHOW-FILE-ATTRIBUTES FILE-NAME=*ALL, -
/ SELECT=*BY-ATTRIBUTES(STORAGE-LEVEL=(S1,S2))
```

Sample output for migrated files

```
/SHOW-FILE-ATTRIBUTES FILE-NAME=TEST.HSMS,INFORMATION=*ALL-ATTRIBUTES
%0000000543#:10SN:$MANUALE.TEST.HSMS
%  ------------------------------ HISTORY      ------------------------------
%  CRE-DATE   = 2016-09-20  ACC-DATE   = 2016-09-29  CHANG-DATE = 2016-09-29
%  CRE-TIME   =   15:15:13  ACC-TIME   =   10:34:05  CHANG-TIME =   10:34:05
%  ACC-COUNT  = 4           S-ALLO-NUM = 0
%  ------------------------------ SECURITY     ------------------------------
%  READ-PASS  = NONE        WRITE-PASS = NONE        EXEC-PASS  = NONE
%  USER-ACC   = OWNER-ONLY  ACCESS     = WRITE       ACL        = NO
%  AUDIT      = NONE        FREE-DEL-D = *NONE       EXPIR-DATE = 2016-09-20
%  DESTROY    = NO         FREE-DEL-T = *NONE       EXPIR-TIME =   00:00:00
%  SP-REL-LOCK= NO          ENCRYPTION = *NONE
%  ------------------------------ BACKUP       ------------------------------
%  BACK-CLASS = A          SAVED-PAG  = COMPL-FILE  VERSION    = 2
%  MIGRATE    = ALLOWED     STOR-LEVEL = S2
%  ------------------------------ ORGANIZATION ------------------------------
%  FILE-STRUC = ISAM        BUF-LEN    = STD(1)      BLK-CONTR  = PAMKEY
%  IO(USAGE)  = READ-WRITE  IO(PERF)   = STD        DISK-WRITE = IMMEDIATE
%  REC-FORM   = (V,N)       REC-SIZE   = 512
%  KEY-LEN    = 10          KEY-POS    = 5
%  AVAIL      = *STD
%  WORK-FILE  = *NO         F-PREFORM  = *K         S0-MIGR    = *ALLOWED
%  ------------------------------ ALLOCATION   ------------------------------
%  SUPPORT    = PUB/S2      S-ALLOC    = 543         HIGH-US-PA = 521
%  EXTENTS     VOLUME     DEVICE-TYPE
%   NONE        NONE         NONE
%:2OSG: PUB/S2:      1 FILE  RES=       543 FRE=        0 REL=       22 PAGES

/SHOW-FILE-ATTRIBUTES FILE-NAME=TEST.
%       543#:1OSN:$MANUALE.TEST.HSMS
%         3 :1OSN:$MANUALE.TEST.MSG
%         3#:1OSN:$MANUALE.TEST.VOLX
%:1OSG: PUBLIC:     1 FILE  RES=         3  FREE=       2  REL=       0 PAGES
%:1OSG: PUB/S2:     2 FILES RES=       546  FREE=       2  REL=       0 PAGES
```

Migrated files are identified by a number sign (#) between the file size value and the catalog ID.

As long as a file is migrated, the following applies to accessing this file and its catalog entry:

- Changes to file attributes such as protection attributes, passwords, etc. are possible and remain in effect after the file is recalled.
- However, the file cannot be renamed; attempts to do so will be rejected with an error message.
- Changes to the file size (SPACE operand in MODIFY-FILE-ATTRIBUTES) are accepted. They do not take effect until the file is recalled.
- The file can be deleted at any time. A migrated file whose catalog entry has been deleted is invalid for HSMS. After all, it cannot be recalled without a catalog entry.
- Overwriting the file (OPEN OUTPUT, OUTIN or UPDATE) is permitted at the processing level. The file is then no longer considered as migrated and also becomes invalid for the migration archive.

4.3.4 Migration and large files (> 32 GB)

BS2000 also supports files >= 32 GB, but only on pubsets with the attributes LARGE-OBJECTS=*YES and LARGE-FILES-ALLOWED=*YES.
HSMS also enables these files to be migrated and then retrieved.

If large files are not moved from pubsets with the above attributes then there are no restrictions regarding files < 32 GB.

4.3.5 Migration and Concurrent Copy function

HSMS cannot migrate files during a Concurrent Copy (CCOPY) session. The following is therefore recommended: the migration inhibit for all files that are planned for a CCOPY session should be activated. In addition, the system administrator should enter the following HSMS statement:

```
//MODIFY-HSMS-PARAMETERS MIGRATION-CONTROL=*PAR(EXCEPT-FILE=...)
```

Where the EXCEPT-FILE operand is used to specify a list of files that are to be excluded from migration.

4.3.6 Recalling migrated files

The opposite of migration, the copying of migrated data back to the processing level, is called recall. The number of files that can be recalled to S0 is restricted by the space allocated to the individual user IDs.

Only files cataloged as migrated files and only that version in the migration archive which matches the catalog entry can be recalled. The current state of a file at the processing level cannot be changed by recall.

Nonprivileged users can recall migrated files of a foreign user ID if they are co-owners or the file protection attributes permit access from another user ID.

Warning

> The following file attributes are modified by a recall: CFID (coded file ID), DEVICE-TYPE, SUPPORT, VOLUME and EXTENTS. This must be taken into consideration in applications which evaluate these attributes.

HSMS offers two different ways of recalling a migrated file, which are described below.

4.3.6.1 Explicit recall by HSMS statement

First HSMS is called and then the file is explicitly recalled by means of the HSMS statement RECALL-MIGRATED-FILES.

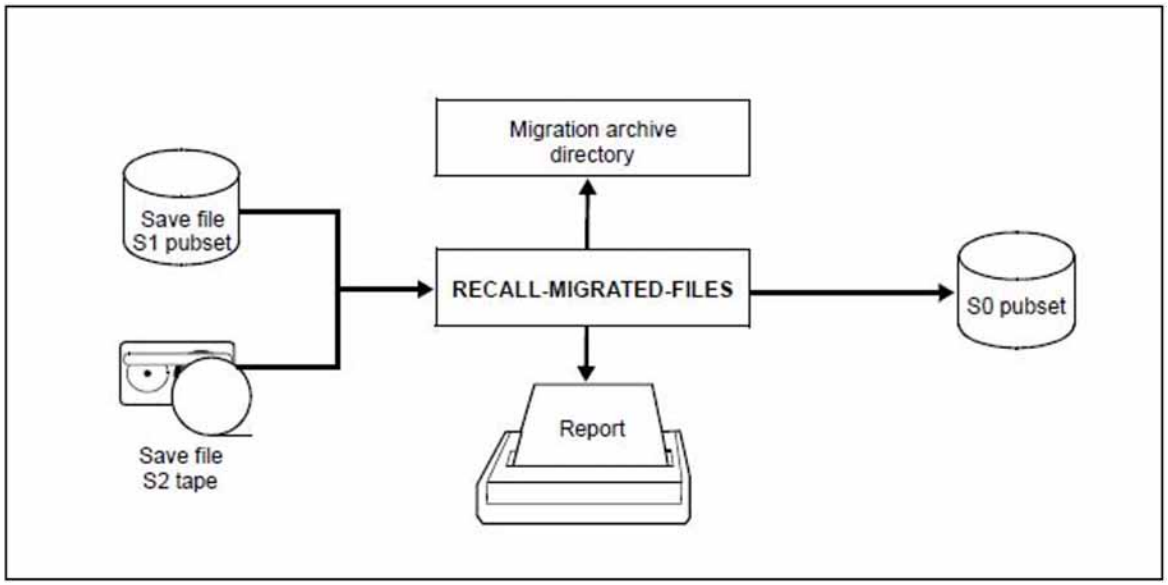


Figure 14: Explicit recall using RECALL-MIGRATED-FILES

Via the FROM-STORAGE operand, the HSMS administrator can specify the storage level from which he or she wishes to recall the files. If the operand is omitted, the files are recalled from the most recent save file in the appropriate migration archive, regardless of the storage level involved (even if it is a copy).
This last procedure is standard for nonprivileged users.

4.3.6.2 Implicit recall on file access

A migrated file can also be recalled without the user issuing an explicit request to HSMS. When a user or user program tries to access a migrated file, the BS2000 Data Management System (DMS) deduces from the catalog entry that the file is not available at the processing level and must first be recalled. If a file is requested with /SECURE-RESOURCE-ALLOCATION with a wait time specification, and if access operations such as OPEN INPUT, INOUT or UPDATE are used, DMS automatically (implicitly) recalls the file.
If the file is recalled implicitly, it is retrieved from the last save file (including copies) on the storage level entered in the catalog.

Implicit recalls are handled differently, depending on whether the file is accessed by means of a SECURE-RESOURCE-ALLOCATION command or an OPEN. A request using the command

```
/SECURE-RESOURCE-ALLOCATION FILE=<file name>,WAIT=...
```

with a wait time before the beginning of processing has the following advantages:

- All files needed for a job (up to 48) can be addressed with a single command and recalled via a single request. As a result, fewer HSMS calls and possibly fewer tape requests are required.
- There is a defined waiting point after which processing either can start or must be aborted. “Unpleasant surprises” caused by missing files during processing can thus be avoided.
- For recall requests initiated in this manner, HSMS outputs a summary report which is otherwise output only if errors occur in implicit recall operations.

The length of the wait time specified via the WAIT operand should be selected so that S2 can also be accessed during this period. If no wait time is specified, DMS reports an error in reserving the file.

If the recall request is started following an OPEN, the following cases have to be distinguished:

- Recall in interactive mode from storage level S1:
  The recall is always executed and no message is output.
- Recall in interactive mode from storage level S2:
  HSMS outputs a message stating that storage level S2 is being accessed and asking the user whether recall is to be started.
  If the user confirms the recall request or does not react within the preset wait time, HSMS issues a second message. Following this, HSMS starts the recall operation, which cannot be aborted after this point.
  If the user does not confirm the recall request, HSMS does not start the recall operation and access to the file is rejected.

  The HSMS administrator can influence the behavior of HSMS by deactivating the interrupt mechanism via the CANCEL-AT-RECALL operand of the HSMS statement MODIFY-HSMS-PARAMETERS. This causes HSMS to suppress message output and to start the recall operation without requesting confirmation. The HSMS administrator can also set the maximum wait time to any value between 10 seconds and 60 minutes using the MAXIMUM-WAIT-TIME operand. However, it is advisable to specify a short wait time for the interrupt mechanism.
- Recall in batch mode:
  The recall is always executed.
- During a recall, the file is first imported under the storage space of the SYSHSMS user ID before being assigned to the final user ID. The SYSHSMS user ID must therefore have sufficient storage space – even for very large files. That is why the storage space extension function (PUBLIC-SPACE-EXCESS) should be allowed when setting up the SYSHSMS user ID.

If you are working in interactive mode, files are automatically recalled from storage level S2 only during the defined tape sessions; otherwise you will receive a message when accessing a migrated file.

Using the backup server

In HSMS V11.0 and higher, the agreed backup server can be used to implicitly recall files which were migrated from shared pubsets, see section "Backup server"). For this purpose, a relevant value must be set in the relevant pubset environment for the archive attribute BACKUP-SERVER-USAGE of the assigned default system archive SYSMIGRATE.

In an SF pubset environment, SYSMIGRATE is a pubset-specific migration archive, or, if not defined, a globally defined migration archive. In an SM pubset environment, SYSMIGRATE is always a pubset-specific migration archive.

If defined, the backup server specified in the local system is used when BACKUP-SERVER-USAGE=*STD is set.

If BACKUP-SERVER-USAGE=*‘NO is set, the backup server functionality is not used, even when a backup server is defined in the local system. Processing takes place in master/slave mode.

If no default system archive SYSMIGRATE is defined, the recall also takes place in master/slave mode.

4.3.7 Using RESTORE on migrated files

With migrated files, HSMS can only administer the required backup data if all the data of the migrated files is included in the backup. This is essential if data lost is to be restored without problems.

HSMS handles migrated files as follows during the restore:

- **Version control**

  When RESTORE is used (for example, during the restore, activation or importation) by a nonprivileged user, migrated files are always brought to storage level S0.

- **Recovering from an S0 crash**

  The HSMS administrator can opt to restore only the catalog entries of migrated files. This can be carried out using the operand MIGRATED-FILES=*CATALOG-ONLY in the RESTORE-FILES statement. This modifies the catalog entries so that they point to the most recent appropriate data in the migration archive (this is an implicit EXCHANGE).

  If the migration archive contains no appropriate data for a file, the system indicates an error. The catalogue entry remains. The system outputs an error message of this kind in connection with files which have been restored or deleted since the last backup, provided the migration area was reorganized after the restoration or deletion. Any such residual inconsistencies can be corrected by

  - restoring the affected files to S0 or a specified migration level. Therefore the HSMS statement REPAIR-CATALOG-BY-RESTORE is provided.
  - deleting the catalog entry.

- **Recovering from an S1 crash**

  Different approaches can be taken:

  - The REPAIR-CATALOG-BY-EXCHANGE statement, provided copies of the data exist in storage level S2 or any other migration archive..
  - The REPAIR-CATALOG-BY-RESTORE statement, provided backups of the data migrated to S1 exist in the backup archive.

  The two approaches can be combined. First, the identifying information of those files for which the migration archive contains data is updated to select the appropriate save files (using the REPAIR-CATALOG-BY-EXCHANGE statement). Then, those files for which the migration archive no longer contains data but which have data in the backup archive are restored from the backup archive using REPAIR-CATALOG-BY-RESTORE. Both statements affect only those migrated files which have the same file status (CFID) in the backup file, which became invalid and is in repair.

- **Recovering from an S2 crash**

  Various approaches can be taken to recover from an S2 crash or destruction of a save file in the migration archive:

  - The REPLACE-SAVE-FILE-BY-EXCHANGE statement, provided the migration archive contains a copy of the save file.
  - The REPLACE-SAVE-FILE-BY-RESTORE statement, provided the backup archive contains backups of the save file data.
  - The COPY-SAVE-FILE statement issued from an long-term archive and followed by a REPAIR-CATALOG-BY-EXCHANGE or REPLACE-SAVE-FILE-BY-EXCHANGE.

- **Recovering from a crash on S0 and the migration levels**

  The storage level S0 can be restored first using the operand MIGRATED-FILES= *CATALOG-ONLY of the RESTORE-FILES statement. The HSMS statements REPAIR-CATALOG-BY-EXCHANGE and REPAIR-CATALOG-BY-RESTORE can subsequently be used to activate or restore the data on the migration level.

> For the troubleshooting of S1 and S2 crashes, it is always required to keep copies of the migrated data. At the time of migration this is guaranteed by the global, respectively SM pubset-specific, HSMS parameter BACKUP-MANDATORY=*YES only being able to migrate files that have been saved before. The backups created here are generally shared after the expiration of the retention period in the backup (//MODIFY-ARCHIVE ... SAVE-FILES = *DELETE(...)) and are then unavailable as backup copy. It is therefore crucial to also save the migrated data from time to time. This is done using //BACKUP-FILES ... SAVE-OPTIONS=*PARAMETERS(SAVE-DATA=*S2-S1-S0)

4.3.8 Migration to S0

Migration to S0 moves a file from a volume set of an SM pubset to another volume set of the same SM pubset. This enables you to

- reorganize an SM pubset
- redistribute the files on an SM pubset when a new, empty volume set is added
- move to a different volume set all the files of a volume set that are to be removed from an SM pubset.

Migration to S0 is not implemented in HSMS by means of a new statement. The existing HSMS functions allow the HSMS administrator to carry out the migration to S0 in three consecutive steps which can be put together in an SDF (-P) procedure:

1. Select files to be migrated using the HSMS statement SELECT-FILE-NAMES. Below are two examples:
   - To reorganize a pubset, all the files are selected that are not located on the best volume set in terms of their attributes:

     ```
     //SELECT-FILE-NAMES FILE-NAMES=:<sm-pubset-id>:$*.*, -
        SELECT-FROM=*CATALOG(STORAGE-LEVEL=*S0, -
           SUPPORT=*SYSTEM-MANAGED-PUBSET -
                   (ALLOCATION-QUALITY=*NOT-BEST-VOLUME-SET)), -
        OUTPUT=<files-to-move.list>
     ```

   - To remove a volume sets from an SM pubset, all the files are selected that are located on that volume set:

     ```
     //SELECT-FILE-NAMES FILE-NAMES=:<sm-pubset-id>:$*.*, -
        SELECT-FROM=*CATALOG(STORAGE-LEVEL=*S0, -
          SUPPORT=*SYSTEM-MANAGED-PUBSET(VOLUME-SET-ID=<volume-set-id>)), -
        OUTPUT=<files-to-move.list>
     ```

2. Migration of files to S2 level:

   ```
   //MIGRATE-FILES ENVIRONMENT=*SYSTEM-MANAGED(CATALOG-ID=<sm-pubset-id>) , -
         FROM-STORAGE=*S0-STORAGE-LEVEL(FILE-NAMES=*FROM-FILE( -
            files-to-move.list>),TO-STORAGE=*S2-STORAGE-LEVEL)
   ```

3. Recall files to S0 level with the HSMS statement RECALL-MIGRATED-FILES:
   - If files are to be recalled to the volume set that best matches their attributes, the following statement must be used:

     ```
     //RECALL-MIGRATED-FILES FILE-NAMES=*FROM-FILE(files.to.move.list) , -
        ENVIRONMENT=*SYSTEM-MANAGED(CATALOG-ID=<sm-pubset-id>)
     ```

   - If files are to be stored on a certain volume set of the SM pubset, the following statement must be used:

     ```
     //RECALL-MIGRATED-FILES FILE-NAMES=*FROM-FILE(files.to.move.list) , -
        ENVIRONMENT=*SYSTEM-MANAGED(CATALOG-ID=<sm-pubset-id>, -
            NEW-DATA-SUPPORT=<volume-set-id>)
     ```

4.3.9 Moving all a user's files

The HSMS administrator can move all a user’s files to a different pubset or a different user ID – independently of whether or not the files are migrated. If files are moved to another user ID, this user ID must be present in the target catalog ID before the transfer is made.

To move files, the HSMS administrator must execute the following HSMS statements in the specified order:

1. A COPY-SAVE-FILE statement must be executed for all the affected save files that contain data belonging to the user ID in question:

```
//COPY-SAVE-FILE SAVE-FILE-ID=<save file>,SELECT-FILES=$USER., -
   NEW-FILE-NAMES=*BY-RULE(NEW-CATALOG-ID=<new cat-id>, -
   NEW-USER-ID=<new user name>), -
   MIGRATION-STATE=*MIGRATED-FILE, -
   ARCHIVE-NAME=<name of the *SYSMIGRATE archive of the old cat-id>, -
   TO-ARCHIVE-NAME=<name of the *SYSMIGRATE archive of the new cat-id>,
   ...
```

   This HSMS statement generates a new save file to which the migrated files of the new user point.

2.
```
//BACKUP-FILES FILE-NAMES=$USER., -
   NEW-FILE-NAMES=*BY-RULE(NEW-CATALOG-ID=<new cat-id>, -
   NEW-USER-ID=<new user name>), -
   SELECT-FILES=*ALL-FILES, -
   SAVE-OPTIONS=*PARAMETERS(SAVE-DATA=*S0), -
   ARCHIVE-NAME=<archive name>, ...
```

   The archive specified in this HSMS can be the default backup archive *SYSBACKUP of the new catalog ID.

3.
```
//RESTORE-FILES FILE-NAMES=$<new user name>., -
   MIGRATED-FILES=*CATALOG-ONLY, -
   ARCHIVE-NAME=<archive name>, ...
```

   After restoration, all the catalog IDs point to the save file which was generated using the COPY-SAVE-FILE statement (see 1.).

4.3.10 Migration examples

This section provides examples dealing with the following subjects:

- migration with compression
- Migration by selection criteria
- Migration by quantity
- Eliminating memory bottlenecks on S0
- Pubset-specific migration with display of pubset usage
- automatic initiation of migration requests by means of ENTER jobs
- Starting migration requests automatically by means of a program

The examples are based on the sample configuration described in "Creating an HSMS configuration (example)".

Migration with compression

Files are migrated to storage level S1 and compressed in the process. The save files are compared with the migrated files.

```
//MIGRATE-FILES -   ————————————————————————————————————————————————  (1)
//  FROM-STORAGE=*S0-STORAGE-LEVEL(FILE-NAMES=$MANUAL.FILE.0*, -
//    COMPRESS-FILES=*YES, -
//    TO-STORAGE=*S1-STORAGE-LEVEL), -
//  OPERATION-CONTROL=*PAR(REPORT=*FULL, -
//    OUTPUT=HSMS.MAN.R.MGF.1, -
//    WAIT-FOR-COMPLETION=*YES)
% HSM0003 HSMS STATEMENT COMPLETED
//END
% HSM0014 HSMS PROGRAM TERMINATED
Report HSMS.MAN.R.MGF.1 (extract):   ——————————————————————————————  (2)

 ***  MIGRATE - FILES               HSMS V12.0       FULL        REPORT  *** 2016-08-12  14:
48:31    PAGE    2
REQUEST-NAME=MGF#0AAK REQUEST-DATE=2016-08-12 14:48:09 USER-ID=SYSHSMS  REQUEST-
STATE=COMPLETED WITHOUT ERROR
%  ARC0002 STATEMENT ACCEPTED. ARCHIVE SEQUENCE NUMBER 'A.160812.144811', VERSION '11.0'
%  ARC0033 ARCHIVE SUBTASK TSN 0AAO GENERATED
                                   SAVE FILE IDENTIFIER - S.160812.144813
SUBSAVE
NUMBER       VSNS
   0        0:2BC
                                   SAVE FILE IDENTIFIER - S.160812.144813
                        ***   CATALOG - 2BY       USER - MANUAL     ***
**   OUTPUT SAVE VERSION: SAVE-VERSION-DATE=16-08-12  SAVE-VERSION-TIME=14:48:13  *
           FILE/JOB VARIABLE NAME                  LASTPG/  SAVE    INPUT DEV  SUB  OUTPUT
                                                  VERS     SIZE  TYPE    VSN  TYP SAVE  VSN(S)
FILE.01                                            1          3  FULL  2BY.01  D     0  0:2BC
FILE.02                                            1          8  FULL  2BY.01  D     0  0:2BC
FILE.03                                            1         13  FULL  2BY.01  D     0  0:2BC
FILE.04                                            1         18  FULL  2BY.01  D     0  0:2BC
FILE.05                                            1         16  FULL  2BY.00  D     0  0:2BC
FILE.06                                            1         11  FULL  2BY.00  D     0  0:2BC
FILE.07                                            1          6  FULL  2BY.00  D     0  0:2BC
 ***      E N D   O F                HSMS V12.0       FULL         REPORT  *** 2016-08-12
12:19:05     ***

/SHOW-FILE-ATT FILE-NAME=$MANUAL.FILE.0*, -   ——————————————————————  (3)
/  INFORMATION=SPACE-SUMMARY
%:2BY:  PUB/S1:     7  FILES RES=        90  FRE=       15 REL=       9 PAGES
/SHOW-FILE-ATT FILE=:2BC:$*.ARCHIVE.SAVE.FILE.160812.144813., -
/  INFORMATION=SPACE-SUMMARY
%:2BC:  PUBLIC:     1  FILE  RES=        12  FRE=        2 REL=       0 PAGES
```

(1) The files named FILE.0.. of user ID MANUAL are migrated to the system migration archive.

(2) Extract from the HSMS-generated report of the migration run with output of the SFID.

(3) Comparison of the migrated files with the save files to which they were written shows the reduced space requirements of the compressed save file. Since the migrated files no longer occupy storage space, the difference in space has been saved.

Migration by selection criteria

The files are selected and migrated on the basis of the retention period criterion.

```
/SHOW-FILE-ATTRIBUTES FILE-NAME=$MANUAL.FILE.1*, -   ————————————————————  (1)
/  SELECT=*BY-ATTRIBUTES(STORAGE-LEVEL=*S0, -
/  EXPIRATION-DATE=*INTERVAL(TO=+365)), -
/  OUTPUT=#SELECT-LIST(FORM-NAME=*FILE-NAME)
%:2BY:  PUBLIC:    7  FILES RES=      90  FREE=       15  REL=        9 PAGES
/START-HSMS
//MIGRATE-FILES -   ————————————————————————————————————————————————————  (2)
//  FROM-STORAGE=*S0-STORAGE-LEVEL(FILE-NAMES=*FROM-FILE -
//    (#SELECT-LIST),COMPRESS-FILES=*YES, -
//    TO-STORAGE=*S1-STORAGE-LEVEL), -
//  OPERATION-CONTROL=*PAR(REPORT=*FULL, -
//    OUTPUT=HSMS.MAN.R.MGF.2, -
//    WAIT-FOR-COMPLETION=*YES)
% HSM0003 HSMS STATEMENT COMPLETED
//END
% HSM0014 HSMS PROGRAM TERMINATED
```

(1) The SHOW-FILE-ATTRIBUTES command writes a list of all files named FILE.1.. which belong to user ID MANUAL and whose retention period expires within the next 365 days to a temporary file. This selects those files whose retention period is situated within the retention period of the save file of the migration archive.

(2) The files contained in the list created by the command are migrated to the system migration archive.

Migration by quantity

Files are migrated until a certain number of PAM pages are free.

```
//MIGRATE-FILES -  ————————————————————————————————————————————————————  (1)
//  FROM-STORAGE=*S0-STORAGE-LEVEL(FILE-NAMES=$MANUAL.FILE.2*, -
//    RELEASE-PAGES=100,-  ——————————————————————————————————————————————  (2)
//    COMPRESS-FILES=*YES,TO-STORAGE=*S1-STORAGE-LEVEL), -
//  OPERATION-CONTROL=*PAR(REPORT=*FULL, -
//    OUTPUT=HSMS.MAN.R.MGF.3, -
//    WAIT-FOR-COMPLETION=*YES)
%  HSM0003 HSMS STATEMENT COMPLETED
//END
% HSM0014 HSMS PROGRAM TERMINATED
```

(1) The files named FILE.2.. of user ID MANUAL are migrated to the system migration archive.

(2) HSMS stops migrating files as soon as at least 100 PAM pages are free, i.e. only a subset of the specified files is migrated.

Eliminating memory bottlenecks on S0

In order to create space on public volumes, files which have not been accessed for a least 28 days are migrated to S1. Specifying SHOW-PUBSET-USAGE
INFORMATION=*UNUSED-DAYS beforehand enables users to see whether the space released in this manner will be enough.

```
//MIGRATE-FILES FROM-STORAGE=*S0-STORAGE-LEVEL -  ————————————————————  (1)
// (FILE-NAMES=:2BY:$*.,TO-STORAGE=*S1-STORAGE-LEVEL, -
//    UNUSED-DAYS=28), -
//  OPERATION-CONTROL=*PAR(REPORT=*FULL, -
//    OUTPUT=HSMS.MAN.R.MGF.1, -
//    WAIT-FOR-COMPLETION=*YES)
% HSM0003 HSMS STATEMENT COMPLETED
//END
% HSM0014 HSMS PROGRAM TERMINATED
```

(1) All files on pubset 2BY which have not been accessed for at least 28 days are migrated to storage level S1. Specifying *ALL is not practical due to the pubset-specific assignment of the system migration archives; *ALL always refers to all files of all pubsets.

Pubset-specific migration with display of pubset usage

A few large files are migrated in order to demonstrate the effect of migration on pubset allocation.

```
//SHOW-PUBSET-USAGE ──────────────────────────────────────────────────────────  (1)
```

```
SHOW-PUBSET-USAGE                                          INFORMATION = SUMMARY
CATALOG-ID = ALL
------------------------------------------------------------------------------
 PUBSET  ST    CAPACITY   %USED  %AVAIL    %MIG     S1-MIG   S1-USED  S1-AVAIL
 BVWC    S0      225657    87.0    13.0      .0          0    593178     70167
 2BC     S1      663345    89.4    10.6      .0
 2BY     S0      663345    77.5    22.5      .0        135    593178     70167
...
...
...
NEXT-PAGE: +  (+, -, ++, --, E)
%  HSM0012 END OF OUTPUT LIST REACHED
```

```
% HSM0003 HSMS STATEMENT COMPLETED
//MIGRATE-FILES -   ──────────────────────────────────────────────────────────  (2)
//  FROM-STOR=*S0-STOR(F-NAMES=:2BY:$MANUAL.MAX-SIZE.*, -
//    MIN-SIZE=150,MAX-SIZE=2500,TO-STOR=*S1-STOR), -
//  OPER-CONTROL=*PAR(REPORT=*NONE)
% HSM0003 HSMS STATEMENT COMPLETED
//MIGRATE-FILES -
//  FROM-STOR=*S0-STOR(F-NAMES=:BVWC:$MANUAL.MAX-SIZE.*, -
//    MIN-SIZE=150,MAX-SIZE=2500,TO-STOR=*S1-STOR), -
//  OPER-CONTROL=*PAR(REPORT=*NONE)
% HSM0003 HSMS STATEMENT COMPLETED
//MIGRATE-FILES -   ──────────────────────────────────────────────────────────  (3)
//  FROM-STOR=*S0-STOR(F-NAMES=:2BY:$MANUAL.MAX-SIZE.*, -
//    MIN-SIZE=2500,TO-STOR=*S2-STOR), -
//  OPER-CONTROL=*PAR(REPORT=*NONE)
% HSM0003 HSMS STATEMENT COMPLETED
//MIGRATE-FILES -
//  FROM-STOR=*S0-STOR(F-NAMES=:BVWC:$MANUAL.MAX-SIZE.*, -
//    MIN-SIZE=2500,TO-STOR=*S2-STOR), -
//  OPER-CONTROL=*PAR(REPORT=*NONE)
% HSM0003 HSMS STATEMENT COMPLETED
//END
% HSM0014 HSMS PROGRAM TERMINATED
/SHOW-FILE-ATTR :*:$MANUAL.MAX-SIZE.*   ──────────────────────────────────────  (4)
%0000001302#:BVWC:$MANUAL.MAX-SIZE.2
%0000095004#:BVWC:$MANUAL.MAX-SIZE.4
%0000080004 :BVWC:$MANUAL.MAX-SIZE.6
%0000001302#:2BY:$MANUAL.MAX-SIZE.1
%0000095004#:2BY:$MANUAL.MAX-SIZE.3
%0000080004 :2BY:$MANUAL.MAX-SIZE.5
%SUM    PUBLIC:      2 FILES RES=    160008 FRE=   160008 REL=   160008 PAGES
%SUM    PUB/S1:      2 FILES RES=      2604 FRE=     2050 REL=     2046 PAGES
%SUM    PUB/S2:      2 FILES RES=    190008 FRE=   160012 REL=   160008 PAGES
//SHOW-PUBSET-USAGE ------------------------------------------------- (05)
```

```
SHOW-PUBSET-USAGE                                          INFORMATION = SUMMARY
CATALOG-ID = ALL
------------------------------------------------------------------------------
 PUBSET  ST    CAPACITY   %USED  %AVAIL    %MIG     S1-MIG   S1-USED  S1-AVAIL
 BVWC    S0      225657    44.3    55.7    42.6       1302    593772     69573
 2BC     S1      663345    89.5    10.5      .0
 2BY     S0      663345    63.0    37.0    14.5       1437    593772     69573
...
...
...
NEXT-PAGE: +  (+, -, ++, --, E)
%  HSM0012 END OF OUTPUT LIST REACHED
```

```
% HSM0003 HSMS STATEMENT COMPLETED
//END
% HSM0014 HSMS PROGRAM TERMINATED
```

(1) Display of pubset usage. No files have been migrated. S0 pubsets show a utilization level of over 70%.

(2) All files under the user ID MANUAL begining with the name MAX-SIZE. and between 150 and 2500 pages large are migrated to S1. In other words, the medium-sized files are written to disk. No report is generated. Any file information can be fetched by reference to the catalog.

(3) All files under the user ID MANUAL begining with the name MAX-SIZE. and larger than 2500 pages are migrated to S2. In other words, these files are written to magnetic tape cartridge.

(4) A SHOW-FILE-ATTRIBUTES statement displays the migrated files and the storage level to which they have been written.
MANUAL.MAX-SIZE.5 and.6 have not been migrated, as they are protected by a migration inhibit.

(5) The pubset usage is displayed again: the percentage of free pages on the S0 pubsets has increased. (% USED and %AVAIL still show 100%, even though %MIG is now also greater than 0. Both values refer solely to the data currently on the pubset, not to the migrated data.)

Starting migration requests automatically by means of an ENTER job

On S0 pubset 2BY, files are to be migrated automatically once saturation level 4 or 5 has been reached. This is achieved by starting the ENTER job after HSMS loading. Execution comments are documented in the job.

```
/.AUTOMIG  SET-LOGON-PARAMETERS
/          REMARK ================================================
/          REMARK === CATID and STORAGE LEVEL of the pubset to     ===
/          REMARK === be monitored and, possibly generated from    ===
/          REMARK === this, the name of the MIGRATION JV can be    ===
/          REMARK === passed as QUASI PARAMETERS in temp. JV!      ===
/          REMARK ================================================
```

```
/          SET-JV-LINK JV-NAME=#CATID
/          MODIFY-JV   JV-NAME(#CATID),VALUE='2BY'
/          REMARK >>>>>> Match CATID >>>>>>^<<<<<< Match CATID <<
/          REMARK
/          SET-JV-LINK JV-NAME=#ST-LEVEL
/          MODIFY-JV   JV-NAME(#ST-LEVEL),VALUE='S0'
/          REMARK >>>>>> Match STORAGE LEVEL >^<<<<<< Match ST-LEVEL <
/          REMARK
/          SET-JV-LINK JV-NAME=#MIGJV
/          MODIFY-JV   JV-NAME(#MIGJV),                        -
/                      VALUE='$SYSHSMS.SYS.HSM.MIGRATE.&(#CATID)'
/          REMARK
/          REMARK
/          REMARK ================================================
/          REMARK === Synchr. wait for HSMS subsystem activation    ==
/          REMARK === Spin-off if not loaded after waiting time!    ==
/          REMARK ================================================
/          REMARK
/          WAIT-EVENT  UNTIL=                                  -
/               JV((&(#MIGJV),1,4) NE X'FFFFFFFF')
/          REMARK ================================================
/          REMARK  When saturation level 4 or 5 is reached, space can
/          REMARK  be made with the MIGRATION function.
/          REMARK
/          REMARK  Releasing the number of pages entered in the
/          REMARK  MIGRATION JV is sufficient to reach the next-lower
/          REMARK  saturation level
/          REMARK ================================================
/.WAIT     WAIT-EVENT UNTIL=                                   -
/               JV((&(#MIGJV),3,1) GE C'4'),                   -
/               TIMEOUT-LABEL=TIMEOUT
/          REMARK ================================================
/          REMARK == When HSMS terminates, this job is also terminated
/          REMARK == terminate
/          REMARK ================================================
/          SKIP-COMMANDS TO-LABEL=LOGOFF,                      -
/            IF=JV((&(#MIGJV),3,1) EQ X'FF')
/          REMARK ================================================
/          REMARK ==== Extract number of pages to be released     ====
/          REMARK ==== and issue migration request                ====
/          REMARK ================================================
/          SET-JV-LINK JV-NAME=#PAGES
/          MODIFY-JV   JV-NAME(#PAGES),                        -
/                 VALUE=JV(JV-NAME=&(#MIGJV),
/                          POSITION=36, LENGTH=10)
/          REMARK
/          REMARK
/          SKIP-COMMANDS  TO-LABEL=MIGS1,                      -
/                 IF=JV(#ST-LEVEL EQ 'S1')
/.MIGS0    REMARK ================================================
/          REMARK ==== For S0 pubset:  normal migration ....      ====
/          REMARK ================================================
/          ASSIGN-SYSDTA TO-FILE=*SYSCMD
/          START-HSMS
//MIGRATE-FILES FROM-STORAGE=*S0-STORAGE-LEVEL                  -
//            (FILE-NAMES=:&(#CATID):$*.,                       -
//            TO-STORAGE=*S1-STORAGE-LEVEL),                    -
//          RELEASE-PAGES=&(#PAGES),                            -
//          OPERATION-CONTROL=*PARAMETERS(EXPRESS-REQUEST=*YES)
```

```
//END
/          SKIP-COMMANDS TO-LABEL=CONT
/.MIGS1    REMARK ================================================
/          REMARK ==== For S1 pubset:  ... reorganization         ====
/          REMARK ================================================
/          ASSIGN-SYSDTA TO-FILE=*SYSCMD
/          START-HSMS
//MIGRATE-FILES FROM-STORAGE=*S1-STORAGE-LEVEL                -
//           (S1-PUBSET-ID=&(#CATID),                         -
//            TO-STORAGE=*S1-STORAGE-LEVEL),                  -
//          RELEASE-PAGES=&(#PAGES),                          -
//          OPERATION-CONTROL=*PARAMETERS(EXPRESS-REQUEST=*YES)
//END
/.CONT     REMARK ================================================
/          REMARK ==== Wait until HSMS modifies JV               =====
/          REMARK ================================================
/          SET-JV-LINK JV-NAME=#LAST-TIME
/          MODIFY-JV   JV-NAME(#LAST-TIME),                  -
/                 VALUE=JV(JV-NAME=&(#MIGJV),                -
/                          POSITION=50, LENGTH=25)
/          WAIT-EVENT UNTIL=JV                               -
/            ((&(#MIGJV),50,25) NE (#LAST-TIME,1,25))
/          REMARK
/          REMARK ================================================
/          REMARK ==== As long as HSMS is active, the pubset       ===
/          REMARK ==== continues to be monitored                   ===
/          REMARK ================================================
/          SKIP-COMMANDS TO-LABEL=WAIT,                      -
/             IF=JV((&(#MIGJV),3,1) NE X'FF')
/          SKIP-COMMANDS TO-LABEL=LOGOFF
/          REMARK
/.TIMEOUT  REMARK ================================================
/          REMARK ==== Waiting time expired, but event has not     ===
/          REMARK ==== occurred; tolerance: up to 10 wait cycles!  ===
/          REMARK ================================================
/          SKIP-COMANDS  TO-LABEL=WAIT,                      -
/             IF=JV($SYSJV.COUNTER LE '0010')
/          SET-JOB-STEP
/          REMARK ================================================
/          REMARK ==== Spin-off due to /WAIT-EVENT, timeout or     ===
/          REMARK ==== real error; report printed                  ===
/          REMARK ================================================
/          LOGOFF  SYSTEM-OUTPUT=*PRINT
/.LOGOFF   REMARK ================================================
/          REMARK ==== Normal termination after unloading HSMS     ===
/          REMARK ================================================
/          LOGOFF  SYSTEM-OUTPUT=*DELETE
```

Starting migration requests automatically by means of a program

Two events are defined in the program: reaching saturation level 1 and termination of the HSMS session. Once the saturation level has been reached, files are migrated via the HSMS macro. (Use of the HSMS macro is discussed in section "Calling HSMS from programs".)

Execution comments are documented in the program.

```
JVTEST   START
         PRINT NOGEN
         AMODE ANY
         RMODE ANY
         GPARMOD 31
         BASR  R10,0
         USING *,R10
*
*     Definition of event codes for two cases:
*     1. Pubset allocation exceeds the specified saturation
*        level 1 and files are to be migrated.
*     2. The HSMS session is terminated, and this program is to be
*        terminated as a consequence.
*
         ENAEI EINAME=MIGRATE,EIIDRET=IDMIGR
         ENAEI EINAME=TERMINAT,EIIDRET=IDTERM
*
*
*     The event of saturation level 1 being exceeded
*     is to be handled asynchronously; a contingency is
*     defined for this purpose.
*
         ENACO CONAME=COMIGR,COADAD=CONTADD,COIDRET=IDCONT
*
*
*     The conditions for the events defined above are
*     specified:
*     1. Saturation level 1 is exceeded in the pubset with catalog ID
*        '2BC', i.e. the associated job variable assumes the value '1'
*        in position 4.
*     2. HSMS is unloaded and the pubset is no longer supported by HSMS,
*        i.e. the job variable for the pubset assumes the value
*        x'ff...ff' in the first 40 positions.
*
         ONEVT '($SYSHSMS.SYS.HSM.MIGRATE.2BC ,4,1)=C''1''',           -
               EIID=IDMIGR,POST='MIGR'
*
         ONEVT '($SYSHSMS.SYS.HSM.MIGRATE.2BC ,4,1)=X''FF''',          -
               EIID=IDTERM,POST='TERM'

*
         XC    RESPONSE,RESPONSE
*
*
*      Signals are requested for the defined events.
*
*      Waiting asynchronously for saturation level 1 to be
*      exceeded (waiting time: 10 minutes):
*
```

```
         SOLSIG EIID=IDMIGR,                                          -
               COID=IDCONT,LIFETIM=600
*
*      Waiting synchronously for HSMS to be terminated
*      (waiting time: 12 minutes):
*
         SOLSIG EIID=IDTERM,COND=UNCOND,                              -
               LIFETIM=720,RPOSTAD=RESPONSE
*
DISEI    DS    0H
*
*      Deactivating the declared event codes:
*
         DISEI EIID=IDMIGR
         DISEI EIID=IDTERM
*
TERM     DS    0H
*
*      Program termination
*
         TERM
*
         EJECT ,
CONTA    DS    0H
*************************************************************************
*                                                                       *
*      Runtime code for contingency handling on exceeding               *
*      the saturation level                                             *
*                                                                       *
*************************************************************************
         BALR  10,0
         USING *,10
*
*
*      Start message: 'MIGRATION CONTINGENCY STARTED'
*
*
         WROUT TEXT1,RETCO
*
*      Calling the proposed migration statement via the
*      HSMS macro
*
         PRINT GEN
         HSMS  MF=S,ADDR=HSMSSTRG,LENGTH=255
         PRINT NOGEN
*
*
*      TERMINATING THE CONTINGENCY
*
*
RETCO    DS    0H
         RETCO
*************************************************************************
*                                                                       *
*     Definition of the migration statement to be executed:             *
*     files comprising at least 5000 PAM pages are to be                *
*     migrated to S1.                                                   *
*                                                                       *
*************************************************************************
```

```
HSMSSTRG DC    CL255'MIGRATE-FILES FROM-STORAGE=*S0-STORAGE-LEVEL/
               (FILE-NAMES=:2BC:$*.,RELEASE-PAGES=5000,TO-STORAGE-LEVEL=/
               *S1-STORAGE-LEVEL)'
*
*
*      Definition of the message text
*
TEXT1    DC    Y(TEXT1E-TEXT1)
         DS    XL3
         DC    C'MIGRATION CONTINGENCY STARTED'
TEXT1E   EQU   *
*
IDMIGR   DS    F
IDTERM   DS    F
IDCONT   DS    F
RESPONSE DS    2F
*
CONTADD  DC    A(CONTA)
R10      EQU   10
         END
```

4.3.11 Remigration examples

Example 1

In the SM pubset SMS1 user TSOS owns files Y2 through Y12, each of which is 3 PAM pages in size. Files Y10, Y11, and Y12 were migrated to level S2 on 08/05/2016 and later retrieved to level S0.

Since then they have only been accessed in read mode.

The migration statement below creates 5 PAM pages of free space in pubset SMS1 by migrating files to level S2:

```
//MIGRATE-FILES ENVIRONMENT=*SYSTEM-MANAGED(CATALOG-ID=SMS1),
     FROM-STORAGE=*S0-STORAGE-LEVEL(FILE-NAMES=:SMS1:Y*,
     RELEASE-PAGES=5),OPERATION-CONTROL=*PARAMETERS(REPORT=*FULL)
```

```
***  MIGRATE - FILES                 HSMS V11.0        FULL            REPORT
** 2016-05-21  12:44:17     PAGE    1
    REQUEST-ENVIRONMENT=SM(SMS1)
    REQUEST-NAME=MGF#0015 REQUEST-DATE=2016-05-21 12:44:15 USER-ID=SYSHSMS  REQ
EST-STATE=COMPLETED   WITHOUT ERROR
STATEMENT LISTING:
MIGRATE-FILES   ENVIRONMENT=*SYSTEM-MANAGED(CATALOG-ID=SMS1),     FROM-STORAGE=
*S0-STORAGE-LEVEL(FILE-NAMES=:SMS1:Y*,
                RELEASE-PAGES=5),OPERATION-CONTROL=*PARAMETERS(REPORT=*FULL)

                ENVIRONMENT                              : SM(SMS1)
                ARCHIVE-NAME                             : $SYSHSMS.MAX

                SAVE-FILE ATTRIBUTES
                TO-STORAGE                               : S2-STORAGE-LEVEL
                DEVICE-TYPE                              : TAPE-C4
                RETENTION-PERIOD                         : 66

                SAVE-VERSION ATTRIBUTES
                SAVE-VERSION-NAME                        : MIGRATE

                     ***   CATALOG - SMS1      USER - TSOS       ***
 **  OUTPUT SAVE VERSION: SAVE-VERSION-DATE=16-05-08  SAVE-VERSION-TIME=12:12:03
  *
  FILE/JOB VARIABLE NAME                  LASTPG/      SAVE     INPUT   DEV    SUB   OUTPUT
                                                         VERS       SIZE     TYPE
VSN      TYP   SAVE
VSN(S)
  Y11                                                      1            0      FULL
*REMIG     D        0
TAPE01
  Y12                                                      1            0      FULL
*REMIG     D        0
TAPE01
     NUMBER OF FILES=              0 GLOBAL SIZE=           0 START= 2016-05-21
12:44:15 END= 2016-05-21 12:44:17
 ***      E N D   O F                HSMS V11.0        FULL            REPORT
*** 2016-05-21  12:44:17     ***
```

Remigratable files Y11 and Y12 are migrated with priority. Remigration takes place in the HSMS server task without storing the data on S2 again.

Example 2

Another migration statement is now used to migrate the remaining files Y2 through Y10 to S2, Y10 being remigrated. Migration takes place in 2 steps here.

- In the first step the remigratable files are migrated (in the HSMS server task).
- In the second step the remaining files are migrated by an archive subtask in accordance with the old procedure.

```
//MIGRATE-FILES ENVIRONMENT=*SYSTEM-MANAGED(CATALOG-ID=SMS1),
     FROM-STORAGE=*S0-STORAGE-LEVEL(FILE-NAMES=:SMS1:Y*),
     OPERATION-CONTROL=*PARAMETERS(REPORT=*FULL)
```

```
A***  MIGRATE - FILES                 HSMS V12.0        FULL            REPORT
*** 2016-05-21  12:45:45     PAGE    1
     REQUEST-ENVIRONMENT=SM(SMS1)
     REQUEST-NAME=MGF#0015 REQUEST-DATE=2016-05-21 12:45:17 USER-ID=SYSHSMS  REQ
UEST-STATE=COMPLETED   WITHOUT ERROR
 STATEMENT LISTING:
 MIGRATE-FILES   ENVIRONMENT=*SYSTEM-MANAGED(CATALOG-ID=SMS1),    FROM-STORAGE=*
S0-STORAGE-LEVEL(FILE-NAMES=:SMS1:Y*),
                 OPERATION-CONTROL=*PARAMETERS(REPORT=*FULL)
                 ENVIRONMENT                              : SM(SMS1)
                 ARCHIVE-NAME                             : $SYSHSMS.MAX
                 SAVE-FILE ATTRIBUTES
                 TO-STORAGE                               : S2-STORAGE-LEVEL
                 DEVICE-TYPE                              : TAPE-C4
                 RETENTION-PERIOD                         : 66
                 SAVE-VERSION ATTRIBUTES
                 SAVE-VERSION-NAME                        : MIGRATE
 %  ARC0033 ARCHIVE SUBTASK TSN '0AAG' GENERATED
 %  ARC0815 SUBTASK '0' HAS TRANSFERRED '9' PAM PAGES FOR '8' FILES AND '0' JVS
IN '0' SECONDS
                                      SAVE FILE IDENTIFIER - S.160521.124519
            SUBSAVE
             NUMBER       VSNS
                    0            TAPE02

                                      SAVE FILE IDENTIFIER - S.160521.124519
                       ***   CATALOG - SMS1      USER - TSOS       ***
 **  OUTPUT SAVE VERSION: SAVE-VERSION-DATE=16-05-08  SAVE-VERSION-TIME=12:12:03
  *
  FILE/JOB VARIABLE NAME                 LASTPG/     SAVE       INPUT    DEV    SUB   OUTPUT
                                                        VERS      SIZE    TYPE
VSN     TYP   SAVE
VSN(S)
  Y10                                                    1           0    FULL
*REMIG     D         0
TAPE01

                                     SAVE FILE IDENTIFIER - S.160521.124519

                      ***   CATALOG - SMS1      USER - TSOS       ***
**  OUTPUT SAVE VERSION: SAVE-VERSION-DATE=16-05-21  SAVE-VERSION-TIME=12:45:20
 *
  FILE/JOB VARIABLE NAME                 LASTPG/     SAVE       INPUT   DEV    SUB   OUTPUT
                                                        VERS      SIZE     TYPE
```

```
VSN    TYP   SAVE
VSN(S)
 Y2                                                    1           1      FULL
PUBK00    D        0
TAPE02
 Y3                                                    1           1      FULL
PUBK00    D        0
TAPE02
 Y4                                                    1           1      FULL
PUBK00    D        0
TAPE02
 Y5                                                    1           1      FULL
PUBK00    D        0
TAPE02
 Y6                                                    1           1      FULL
PUBK00    D        0
TAPE02
 Y7                                                    1           1      FULL
PUBK00    D        0
TAPE02
 Y8                                                    1           1      FULL
PUBK00    D        0
TAPE02
 Y9                                                    1           1      FULL
PUBK00    D        0
TAPE02
     NUMBER OF FILES=              8 GLOBAL SIZE=           8 START= 2016-05-21
12:45:18 END= 2016-05-21 12:45:45
 ***      E N D   O F                HSMS V12.0        FULL            REPORT
*** 2016-05-21  12:45:45     ***
```

Example 3

The basis here is once more the set of files from example 1. The MIGRATE statement below creates free space in pubset SMS1 by migrating only remigratable files to S2 without storing the data.

```
//MIGRATE-FILES ENVIRONMENT=*SYSTEM-MANAGED(CATALOG-ID=SMS1),
     FROM-STORAGE=*S0-STORAGE-LEVEL(FILE-NAMES=:SMS1:Y*,
     MIGRATION-INFO=*REMIGRATION),OPERATION-CONTROL=*PARAMETERS(REPORT=*FULL)
```

```
A***  MIGRATE - FILES                HSMS V12.0        FULL            REPORT
*** 2016-05-21  13:04:35     PAGE    1
     REQUEST-ENVIRONMENT=SM(SMS1)
     REQUEST-NAME=MGF#0015 REQUEST-DATE=2016-05-21 13:04:32 USER-ID=SYSHSMS  REQ
UEST-STATE=COMPLETED   WITHOUT ERROR
 STATEMENT LISTING:
 MIGRATE-FILES   ENVIRONMENT=*SYSTEM-MANAGED(CATALOG-ID=SMS1),     FROM-STORAGE=
*S0-STORAGE-LEVEL(FILE-NAMES=:SMS1:Y*,
                 MIGRATION-INFO=*REMIGRATION),OPERATION-CONTROL=*PARAMETERS(REPO
RT=*FULL)
                 ENVIRONMENT                              : SM(SMS1)
                 ARCHIVE-NAME                             : $SYSHSMS.MAX
                 SAVE-FILE ATTRIBUTES
                 TO-STORAGE                               : S2-STORAGE-LEVEL
                 DEVICE-TYPE                              : TAPE-C4
                 RETENTION-PERIOD                         : 66
                 SAVE-VERSION ATTRIBUTES
                 SAVE-VERSION-NAME                        : MIGRATE
                       ***   CATALOG - SMS1      USER - TSOS       ***
 **  OUTPUT SAVE VERSION: SAVE-VERSION-DATE=16-05-08  SAVE-VERSION-TIME=12:12:03
  *
FILE/JOB VARIABLE NAME               LASTPG/      SAVE       INPUT    DEV    SUB   OUTPUT
                                                    VERS    SIZE       TYPE
VSN       TYP   SAVE
VSN(S)
  Y10                                                  1         0       FULL
*REMIG       D       0
TAPE01
  Y11                                                  1         0       FULL
*REMIG       D       0
TAPE01
  Y12                                                  1         0       FULL
*REMIG       D       0
TAPE01
     NUMBER OF FILES=              0 GLOBAL SIZE=           0 START= 2016-05-21
13:04:33 END= 2016-05-21 13:04:35
 ***      E N D   O F                HSMS V12.0        FULL            REPORT
*** 2016-05-21  13:04:35     ***
```

4.4 Data transfer of BS2000 files and job variables

HSMS can be used to transfer files, job variables and catalog entries (of files on private volumes) to other BS2000 systems or other user IDs. To do this, the data is written to magnetic tape cartridges, private disk, public disk, or Net-Storage (exported) and then loaded (imported) at the transfer destination, i.e. the desired system or desired user ID. The generated magnetic tape cartridges are ARCHIVE-compatible, which means that either HSMS or ARCHIVE must be installed on the other system. It is also possible to import magnetic tape cartridges written with the ARCHIVE EXPORT function. Moreover, data can be written to the volume or imported without a catalog ID.

Directories which are compatible with ARCHIVE directories can be used for data transfer. The directories can be written to the volume after the exported data.

4.4.1 Export files and job variables

The HSMS statement EXPORT-FILES is used for exporting files and job variables.

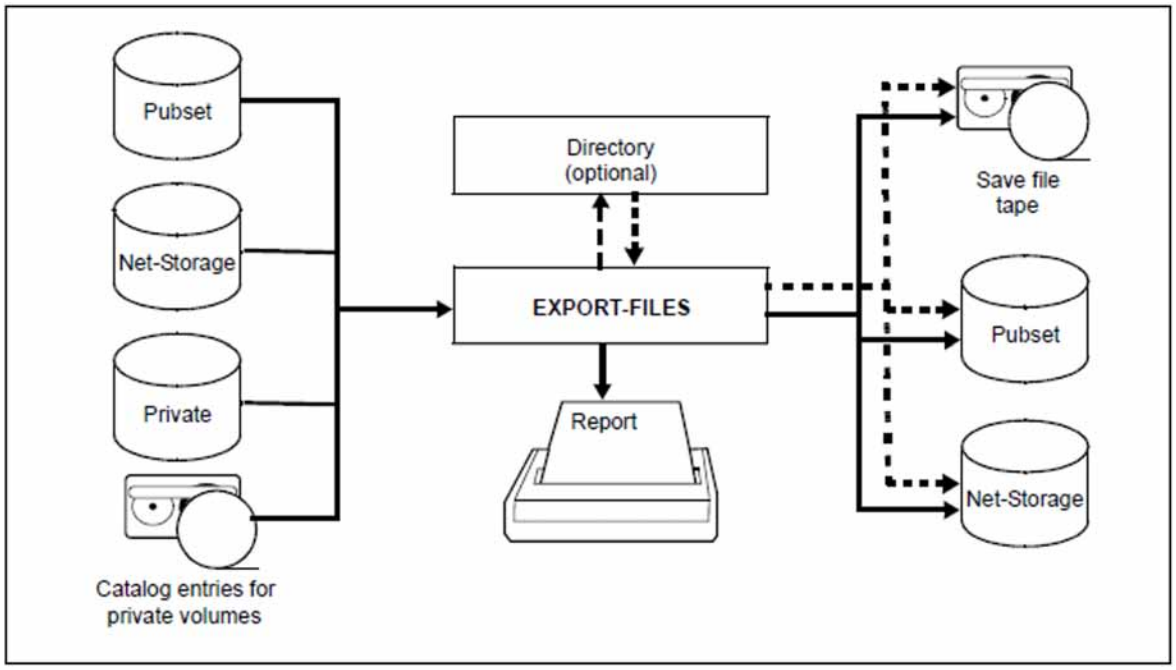


Figure 15: Exporting files with EXPORT-FILES

The files and job variables to be exported from pubsets or private disks can be selected according to the type of volume containing the files (SUPPORT operand). When files on Net-Storage are selected (STORAGE-TYPE operand), the Net-Storage files can also be restricted to the file type BS2000 or node file.

Of course, you can also specify a list of file names (see section "Selection of BS2000 files, job variables and node files").

Migrated files (without recall) can also be exported. The backup includes the catalog entries from the catalog and the data from the migration level.

Files or job variables which are protected by a password can be exported only if the password is entered. The PASSWORDS operand is used for this purpose.

When files are exported without a directory file, shareable files belonging to other user may be saved as well.

When EXPORT-FILES is used to export files/job variables without a directory file, the NEW-FILE-NAMES/NEW-JV-NAMES operand can be used to specify new names.

During the export operation, you can use the SAVE-FILE operand to determine whether:

- a new save file is generated for this export run (=*NEW)
- an existing save file is continued (=*CONTINUE).

If the data is to be imported to a different user ID, a shareable save file must be created:

```
SAVE-FILE=*NEW(USER-ACCESS=*ALL-USERS)
```

The save file can also be protected by a separate password which must be specified in order to import the save file.

4.4.1.1 Exporting catalog entries

HSMS generally saves and archives both the data and the catalog entry of a file (if the data is available). When exporting files, however, it is possible to write to the export volume only the catalog entries of files located on magnetic tape cartridges or on private disk (save type CATL).

For files located on private disk, this is done by specifying

```
SUPPORT=*PRIVATE-DISK(...,CATALOG-ENTRIES-ONLY=*YES)
```

For files on tape, you must specify

```
SUPPORT=*TAPE
```

When files are exported, subsequent importation into a different system environment can be facilitated by carrying out the following:

- Specifying whether certain file attributes should be adopted from the file's catalog entry or set to defaults. (FILE-ATTRIBUTES=*KEEP/*RESET-TO-STD operand).
- Specifying whether the file's catalog ID and user ID should be included in the save file (using the operand CAT-AND-USER-ID=*KEEP/*IGNORE).
  If the catalog ID and user ID are not included in the save file, the files can be imported without needing to know the exporting catalog ID and user ID.

4.4.1.2 Directory

When exporting files, a directory for the data written to the export volume can be created, if desired. This directory is equivalent to the archive directory of the other HSMS basic functions and is compatible with ARCHIVE directory files.

The directory is created via the DIRECTORY-NAME operand. You can also specify whether a new directory should be created and whether the directory should be written to the volume as the last file (SAVE-DIRECTORY=*NO /*YES).

When importing files, the directory can then be the first file read from the volume. It provides a summary of the importable data on the other BS2000 system. However, only the HSMS administrator can use the HSMS statement SHOW-ARCHIVE to output the files of other user IDs.

4.4.2 Importing exported files and job variables

The opposite process, namely loading exported data from magnetic tape cartridges, private disk, or Net-Storage (created with either HSMS or ARCHIVE) to another BS2000 system or another user ID, is called importing.

The HSMS statement IMPORT-FILES is available for this purpose.

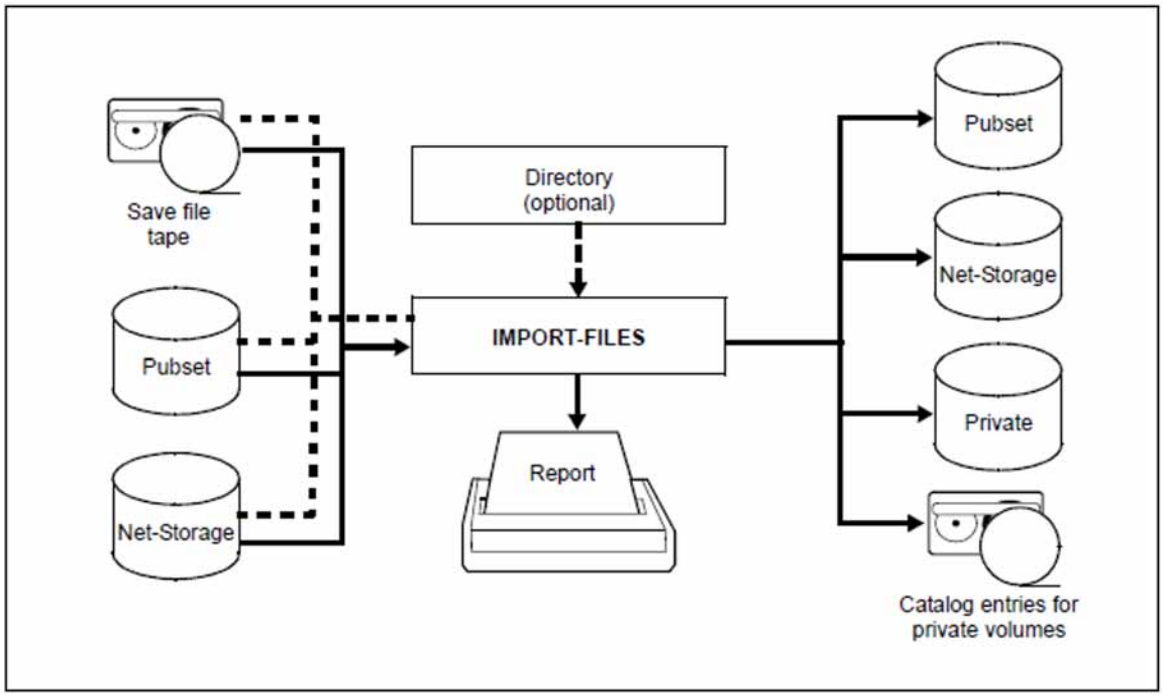


Figure 16: Importing files with IMPORT-FILES

When importing files, the passwords for the data and possibly for the save file must be specified (PASSWORDS operand).

During the import operation, the default catalog ID is added to data exported without a catalog ID.

When importing save files created in several parallel runs, a separate *GROUPED-BY-RUN entry is required for each parallel run (see example on "Examples of data transfer").

4.4.2.1 Importing files with directory

If a directory was created and written to the volume during the export operation, it can be used for importing. In order to do this, the directory must be imported in a separate import run. This run is implemented even if no directory is used.
The other data can be imported in a second run with specification of the directory:

```
//IMPORT-FILES SAVE-FILE=*FROM-DIRECTORY -
// (DIRECTORY-NAME=<verzeichnisname>)
```

4.4.2.2 Importing with MAREN support

If no directory was created during the export operation, all the VSNs and the volume type of the backup volumes would have to be specified for importing. When MAREN V12.0 or higher is used, the input save file can also only be specified with its save file ID, and MAREN determines the associated VSNs and volume type:

```
//IMPORT-FILES SAVE-FILE=*BY-VOLUME-CATALOG -
// (SAVE-FILE-ID=S.yymmdd.hhmmss)
```

4.4.2.3 Renaming files and job variables

When importing files, the NEW-FILE-NAMES operand must be used to rename files if the user ID and configuration are different. This cannot be done for individual files but only for all files imported by the same request; it takes place according to defined rules:

- The files can be given a prefix or suffix.
- The files can be moved to another catalog ID, provided the user involved has an entry in this pubset’s user catalog.
- If the save file was created with USER-ACCESS=*ALL-USERS, the files can be renamed to another user ID. If the files are not shareable, this option is available only to the HSMS administrator.

The same procedure applies when using the NEW-JV-NAMES operand for job variables. As always, the BS2000 naming conventions must be observed when renaming files.

4.4.2.4 Import directory

When the directory has been saved, it can be restored from save file on tape, Net-Storage, or private or public disk in the event of an emergency (if it no longer exists or is not readable). To do this you use the IMPORT-FILES statement with *DIRECTORY and SAVE-FILE specified: Here, it is necessary to differentiate between the following cases:

- The save file is located on public disk or it was created in HSMS V10.0A or higher in *HSMS-V10-COMPATIBLE mode on Net-Storage.
  In these cases SAVE-FILE=*BY-PUBLIC-DISK must be specified with the pubset ID and SVID:

```
//IMPORT-FILES FILE-NAMES=*DIRECTORY, SAVE-FILE=*BY-PUBLIC-DISK-
//  (SAVE-FILE-ID = S.yymmdd.hhmmss, PUBSET-ID = <cat-id>)
```

- The save file is located on tape or private disk or it was created in HSMS V9.0B or in HSMS V10.0A or higher in *HSMS-V9-COMPATIBLE mode on Net-Storage.
  In this case SAVE-FILE=*BY-VOLUME must be specified.

```
//IMPORT-FILES FILE-NAMES=*DIRECTORY, SAVE-FILE=*BY-VOLUME(
      (SAVE-FILE-ID=S.yymmdd.hhmmss,
       VOLUMES=<vsn 1..6>, DEVICE-TYPE=<c-string 1..8>)
```

Save files on tape have a several SVID structure. In this case you need to specify an SVID, and in any other case the SFID. The SFID, SVID and volume or pubset are noted in the report on the backup.

4.4.2.5 Importing files on Net-Storage

In BS2000/OSD-BC V9.0 and higher a pubset can be provided with additional storage space which a net server makes available as Net-Storage. This enables files from pubsets to reside either on local pubset disks or remotely on a Net-Storage. BS2000 OSD/BC V10.0 and higher supports both the file type BS2000 and node file on Net-Storage.

The ORIGINAL-SUPPORT operand can be used to specify that files are to be imported from a Net-Storage (STORAGE-TYPE operand). Importing of Net-Storage files can also be restricted to the file type BS2000 or node file (FILE-TYPE operand). By default all Net-Storage files are imported irrespective of the file type.

The NEW-SUPPORT operand can be used to specify that files are to be stored on a Net-Storage when they are imported.

BS2000 files from public volume or Net-Storage files of the type BS2000 cannot be imported to a Net-Storage as node files. Net-Storage files of the file type node file, whose SAM structure has been saved, can also be imported to Net-Storage or a public volume. During this process, their file type can be changed. In contrast, net storage files of the file type node file, whose SAM structure has not been saved, can only be restored on a Net-Storage volume and with their original file type node file.

4.4.2.6 Importing files from the save file of a pubset or of the Net-Storage

Even if the archive directory of a backup on public disk or Net-Storage no longer exists or the files were exported without a directory, the files can be imported again. In these cases SAVE-FILE=*BY-PUBLIC-DISK must be specified with the pubset ID and SVID:

```
//IMPORT-FILES FILE-NAMES=..., SAVE-FILE=*BY-PUBLIC-DISK-
//  (SAVE-FILE-ID = S.yymmdd.hhmmss, PUBSET-ID = <cat-id>)
```

In the case of Net-Storage, the import is possible only if the save file was generated on Net-Storage in *HSMS-V10-COMPATIBLE mode in HSMS V10.0A or higher. Otherwise files can still be imported using SAVE-FILE=*BY-VOLUME():

```
//IMPORT-FILES FILE-NAMES=..., SAVE-FILE=*BY-VOLUME-
//  (SAVE-FILE-ID = S.yymmdd.hhmmss, VOLUMES = <vsn 1..6>, DEVICE-TYPE = NETSTOR)
```

4.4.3 Examples of data transfer

This section provides examples dealing with the following subjects:

- Transferring files in several parallel runs
- Transferring catalog entries of private disks
- transfer to another processor with directory
- transfer to another default catalog ID.
- Transferring files to a public volume and Net-Storage

Transferring files in several parallel runs

The files of three users are to be transferred and processed in parallel.

```
/START-HSMS
//EXPORT-FILES -  ————————————————————————————————————————————  (1)
//  F-NAMES=($MANUAL.FILE.0*,$USEROLD.FILE.1*, -
//     $USERNEW.FILE.2*), -
//  DIR-NAME=HSMS.MAN.EXF.DIR.1(NEW-DIR=*YES), -
//  TO-STOR=*TAPE(VOL=(HSMS11,HSMS22,HSMS33)), -
//  SAVE-F=*NEW(RET-PER=0), -
//  OPER-CONTROL=*PAR(REPORT=*FULL,PAR-RUNS=3, - ————————————————————  (2)
//     OUT=HSMS.MAN.R.EXF.1,WAIT-F-C=*YES)
% HSM0003 HSMS STATEMENT COMPLETED
//SHOW-ARCHIVE -  ————————————————————————————————————————————  (3)
//  ARCH-NAME=*BY-DIR(DIR-NAME=HSMS.MAN.EXF.DIR.1), -
//  SELECT=*SAVE-F
```

```
 SHOW-ARCHIVE (SAVE-FILES)                INFORMATION       = SUMMARY
 ARCHIVE-NAME    = BY-DIR(:2BY:$TSOS.HSMS.MAN.EXF.DIR.1)
 SAVE-FILE-STATE = ANY                    SAVE-FILE-STORAGE = ANY
 CREATED-BEFORE  = LATEST                 EXPIRATION-BEFORE = LATEST
 ------------------------------------------------------------------------------
 M SFID             CREA-DATE  EXP-DATE  OBS  ACCESS ST  DEVICE    #VOL #SV #R
 UNS
   S.160812.155059  16-08-12   16-08-12  YES  ALL    TAP TAPE-C4   3    1   3
 ....
 NEXT-PAGE: +  (+, -, ++, --, E)
 %  HSM0012 END OF OUTPUT LIST REACHED
```

```
% HSM0003 HSMS STATEMENT COMPLETED
//END
% HSM0014 HSMS PROGRAM TERMINATED
```

```
/START-HSMS
//IMPORT-FILES F-NAMES=*ALL, -  ————————————————————————————————  (4)
//  SAVE-F=*BY-VOL(VOL= -
//    (*GROUPED-BY-RUN(HSMS11), -  ————————————————————————————————  (5)
//     *GROUPED-BY-RUN(HSMS22), -
//     *GROUPED-BY-RUN(HSMS33))), -
//  REPL-F-AND-JV=*YES, -
//  OPER-CONTROL=*PAR(REPORT=*FULL,PAR-RUNS=3, -  ————————————————————  (6)
//    OUT=HSMS.MAN.R.IMF.1, -
//    WAIT-F-C=*YES)
% HSM0003 HSMS STATEMENT COMPLETED
//END
% HSM0014 HSMS PROGRAM TERMINATED
```

(1) The files of three different user IDs are written onto magnetic tape cartridges in one export run. (The report of this run serves as an example for discussing the report format, see section "Reports"f.)

(2) Three parallel runs are defined, over which HSMS distributes the magnetic tape cartridges specified under VOLUMES.

(3) As the export run was effected with a directory, the attributes of the generated save file can be obtained via SHOW-ARCHIVE. The files were written to three volumes (#VOL) in three parallel runs (#RUNS).

(4) The files are read in again in an import run, the save file being defined by the volumes.

(5) A separate *GROUPED-BY-RUN entry is made for each magnetic tape cartridge of a parallel run. VOLUME=(HSMS11,HSMS22,HSMS33) could be entered alternatively. However, *GROUPED-BY-RUN results in a runtime improvement.

(6) Moreover, three parallel runs must be declared.

Transferring catalog entries of private disks

The catalog entries of private disks are transferred to another BS2000 processor.

```
/START-HSMS
//EXPORT-FILES F-NAMES=$MANUAL.FILE.*, -   ————————————————————————————  (1)
//  SUP=*PRIV-DISK(VOL=*ALL,CAT-ENTRIES-ONLY=*YES), -   ————————————————  (2)
//  DIR-NAME=*NONE, -
//  TO-STOR=*TAPE(VOL=HSMS33), -
//  SAVE-F=*NEW(RET-PER=0), -
//  OPER-CONTROL=*PAR(REPORT=*FULL)
% HSM0002 HSMS STATEMENT ACCEPTED
//END
% HSM0014 HSMS PROGRAM TERMINATED
```

On another processor:

```
/START-HSMS
//IMPORT-FILES F-NAMES=*ALL, -   ————————————————————————————————————  (3)
//  SAVE-F=*BY-VOL(VOL=HSMS33), -
//  ORIG-SUP=*PRIV-DISK(VOL=WORKKB), -
//  REPL-F-AND-JV=*YES, -
//  OPER-CONTROL=*PAR(REPORT=*FULL)
% HSM0002 HSMS STATEMENT ACCEPTED
//END
% HSM0014 HSMS PROGRAM TERMINATED
```

(1) In an export run, the catalog entries of all those files on private disks whose names begin with the string “$manual.file” are written to cartridge. No directory is created.

(2) This entry CATALOG-ENTRIES-ONLY causes only the catalog entries but not the data to be included in the save file.

(3) The catalog entries of the files matching the search string and residing on private disk WORKKB are read in on a BS2000 processor in an import run. The save file is determined via the cartridge.

Transfer to another processor with directory

Various files are to be transferred to another BS2000 processor under a different user ID. A directory is used.

```
/START-HSMS
//EXPORT-FILES F-NAMES=$MANUAL.FILE.*, -  ————————————————————————  (1)
//  DIR-NAME=$MANUAL.HSMS.MAN.EXF.DIR.4 -
//    (NEW-DIR=*YES,SAVE-DIR=*YES), -
//  TO-STOR=*TAPE(VOL=HSMS33), -
//  SAVE-F=*NEW(RET-PER=0), -
//  OPER-CONTROL=*PAR(REPORT=*FULL)
% HSM0002 HSMS STATEMENT ACCEPTED
//END
% HSM0014 HSMS PROGRAM TERMINATED
```

On another processor:

```
/START-HSMS
//IMPORT-FILES F-NAMES=*DIRECTORY, -  ————————————————————————————  (2)
//  NEW-F-NAMES=*BY-RULE(NEW-USER-ID=USERNEW2), -
//  SAVE-F=*BY-VOL(VOL=HSMS33), -
//  REPL-F-AND-JV=*YES, -
//  OPER-CONTROL=*PAR(REPORT=*FULL)
% HSM0002 HSMS STATEMENT ACCEPTED
//IMPORT-FILES F-NAMES=$MANUAL.FILE., -  ————————————————————————  (3)
//  NEW-F-NAMES=*BY-RULE(NEW-USER-ID=USERNEW2), -
//  SAVE-F=*FROM-DIR(DIR-NAME=$USERNEW2.HSMS.MAN.EXF.DIR.4), -
//  REPL-F-AND-JV=*YES, -
//  OPER-CONTROL=*PAR(REPORT=*FULL)
% HSM0002 HSMS STATEMENT ACCEPTED
//END
% HSM0014 HSMS PROGRAM TERMINATED
```

(1) All files. of user ID MANUAL whose name begins with FILE are written to a cartridge using EXPORT-FILES. For the run, a new directory is created and written onto the export volume. The report is directed to SYSLST.

(2) In a first import run the directory is read in. The save file is determined via the volume.

(3) Subsequently an import run for the other files is started using the directory. The directory could also be used to select the files via SELECT-FILE-NAMES prior to the run.

Transfer to another default catalog ID

A user is to maintain his files on a different default pubset in future. For this purpose, the existing files must be transferred to the new default pubset.

```
/START-HSMS
//EXPORT-FILES F-NAMES=$MANUAL.*, -   ————————————————————————————————  (1)
//  TO-STOR=*TAPE(VOL=HSMS33), -
//  SAVE-F=*NEW(RET-PER=0), -
//  OPER-CONTROL=*PAR(REPORT=*FULL, -
//    OUT=HSMS.MAN.R.EXF.6,WAIT-F-C=*YES)
% HSM0003 HSMS STATEMENT COMPLETED
//END
% HSM0014 HSMS PROGRAM TERMINATED
/MODIFY-USER-ATTRIBUTES MANUAL,DEFAULT-PUBSET=2BC, -   ————————————————  (2)
/  PUBLIC-VOLUME-SET=2BY
/START-HSMS
//IMPORT-FILES F-NAMES=:2BY:$MANUAL.*, -   ————————————————————————————  (3)
//  NEW-F-NAMES=*BY-RULE(NEW-CAT-ID=2BC), -
//  SAVE-F=*BY-VOL(VOL=HSMS33), -
//  OPER-CONTROL=*PAR(REPORT=*FULL, -
//    OUT=HSMS.MAN.R.IMF.6,WAIT-F-C=*YES)
% HSM0003 HSMS STATEMENT COMPLETED
//END
% HSM0014 HSMS PROGRAM TERMINATED
```

(1) All files of user ID MANUAL are written to a cartridge using the HSMS statement EXPORT-FILES.

(2) The default catalog ID of user ID MANUAL is altered.

(3) The files are imported and written to the default catalog ID using the NEW-FILE-NAMES operand. The old catalog ID must be known, as it is no longer the default for this user ID.

Note

When the export run is executed with CATALOG-ID-MODE=*NO, renaming is not required when the import run takes place. HSMS then automatically writes to the default catalog ID.

Transferring files to a public volume and Net-Storage

The files are copied to a public volume:

```
//EXPORT-FILES FILE-NAMES=LM.TEST.*, -
//  TO-STORAGE=*PUBLIC-DISK(PUBSET-ID=IBA6), -
//   OPERATION-CONTROL=*PAR (REPORT=*FULL,   -
//    OUTPUT=LM.EXP.PUB.REP)

% HSM0003 HSMS STATEMENT COMPLETED
```

The files are transferred to Net-Storage:

```
//EXPORT-FILES FILE-NAMES=LM.TEST.*, -
//  TO-STORAGE=*NET-STORAGE(VOLUMES=NATB00), -
//   OPERATION-CONTROL=*PAR (REPORT=*FULL,   -
//    OUTPUT=LM.EXP.NET.REP)

% HSM0003 HSMS STATEMENT COMPLETED
```

On a different server:

```
//IMPORT-FILES FILE-NAMES=LM.TEST,
//  SAVE-FILE=*BY-PUBLIC-DISK( -
//     SAVE-FILE-ID=S.161228.144335,PUBSET-ID=IBA6), ——————————————————  (1)
//  OPERATION-CONTROL=*PARAMETERS( -
//     REPORT=*FULL,OUTPUT=LM.EXP.REP)

% HSM0003 HSMS STATEMENT COMPLETED
```

(1) The NATB00 Net-Storage volume is assigned to the IBA6 pubset. This pubset can be used for files transfers.

4.5 Duplicating save files

HSMS enables the duplication (copying) of backup files both within an archive and from one archive to another. The right to duplicate save files is restricted to the owner of the archive involved and the HSMS administrator. The copied save file and the BS2000 files, job variables, node files and save versions it contains are entered in the archive directory.

The duplicate function can be used for the following tasks:

Task	DMS files	Node files
Moving save files from disk to tape	X	X
Duplicating save files as a precaution against loss of data	X	X
Reducing the number of volumes by combining save files (see example in "Examples")	X	X
Reshuffling volumes within long-term archives	X	X
Reorganizing a migration archive	X	–

X The task is supported.

– The task is not supported.

4.5.1 Copying save files

Save files can be copied using the HSMS statements COPY-SAVE-FILE, COPY-NODE-SAVE-FILE and COPY-EXPORT-SAVE-FILE.

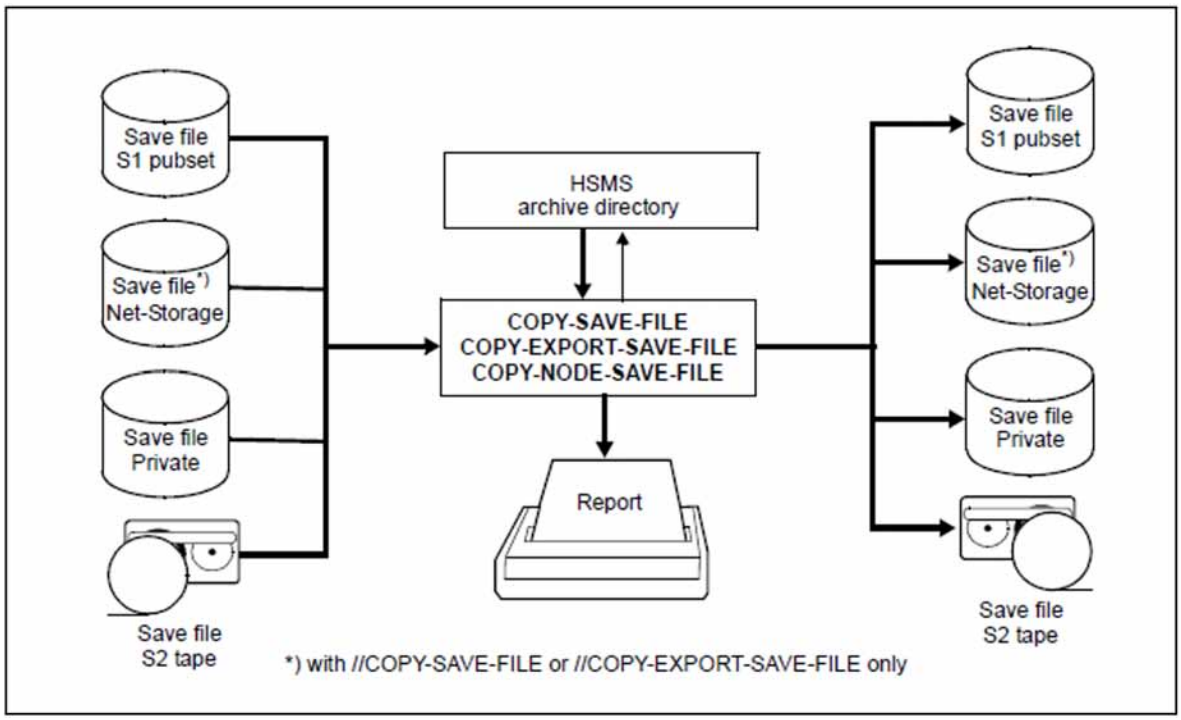


Figure 17: Copying save files

It is possible to select specific save versions, BS2000 files, job variables and node files from the save file for copying.

The new save file and the save versions managed therein receive either the same time stamp or a new one, depending on the input and output archives.

When copying save versions of a migration or long-term archive the original creation date (`ORIGINAL-DATE`) is also recorded. The original creation date of a save version of a migration or long-term archive contains the date and time (precise to the second) of the original migration or original archival.

The original creation date can be output using the following statement:

```
//SHOW-ARCHIVE SELECT=*SAVE-VERSIONS(INFORMATION=*USER-INFORMATION)
```

When a save version is generated through archival, it is assigned the original creation date which corresponds to its SVID.

When a save version from S1 level, public disk, private disk, or Net-Storage is copied to S2 level, all files and job variables are copied to a save version.

4.5.1.1 Managing duplicate save files

//COPY-SAVE-FILE and //COPY-NODE-SAVE-FILE can be used to duplicate save files, thereby increasing data security by managing double save files. Unless otherwise specified, the last save file is copied in its entirety to storage level S2. Only the archive name must be specified.

If save files are continued over a long period of time, they can be successively duplicated even during the continuation period:

```
SELECT-SAVE-VERSIONS=*BY-ATTRIBUTES(
                        SAVE-VERSION-DATE=*INTERVAL(CREATED-AFTER=<datum>))
```

Each save version added on the specified date is copied to the save file for the copy.

Save files for data transfer

Save files created with //EXPORT-FILES can be duplicated with //COPY-EXPORT-SAVE-FILES. By default, the copy is created on magnetic tape cartridge. The copy can also be created on public disk or Net-Storage.

The data contained can be updated with //UPDATE-EXPORT-SAVE-FILE.

4.5.1.2 Copying within backup archives

A backup which was written to an S1 pubset or a private disk outside of a tape session can be transferred at a later point in time to storage level S2 (i.e. to magnetic tape cartridge).

Only the current save files can be copied from backup archives (SAVE-FILE-ID=*LATEST). Even if the save versions contain a new date, the chronological order of the backups is retained in the archive with the copies.

When copying incremental backups, the CNS entries are also copied to the new save file, so that the effect of //RESTORE-FILES with SAVE-VERSIONS=*LATEST remains the same.

4.5.1.3 Copying within migration archives

Save files in a migration archive can be copied in order to move them to a more cost-effective storage level and to reorganize the migration archive (see section "Copying the save files").

Save files in migration archives cannot be copied to private disk or S1.

4.5.1.4 Copying within long-term archives

Archive tapes can be reshuffled after a prolonged storage time in order to prevent the volumes becoming outdated. Copying these archive tapes can be restricted to those save versions whose file expiration date has not yet been reached:

```
//COPY-SAVE-FILE SELECT-SAVE-VERSIONS=*BY-ATTRIBUTES -
// (EXPIRATION-AFTER=<datum>)
```

When a save version is copied the copy is assigned the original creation date of the original.

Save files in long-term archives cannot be copied to private disk.

4.5.1.5 Copying from one archive to another

It is possible to copy save files from one archive to another. The TO-ARCHIVE-NAME operand of the HSMS statements COPY-SAVE-FILE and COPY-NODE-SAVE-FILE is used to specify a target archive for the copy operation. The following options are supported:

- **Copying from one backup archive to another**
  The system creates an identical copy, in other words the time stamp of the save version copy is identical to that of the original; the CNS entries are also copied.

  Note

  The time stamp of the save version copy must be higher than the highest time stamp of the target archive.
- **Copying from one long-term archive to another**
  The copy receives a new time stamp. When a save version is copied the copy is assigned the original creation date of the original.
- **Copying from one migration archive to another**
  The copy receives a new time stamp. When a save version generated through migration is copied the copy is assigned the original creation date which corresponds to the SVID of the original. When a save version which has an original creation date is copied the copy is assigned the original creation date of the original.
- **Copying from a migration archive to a long-term archive**
  The copy receives a new time stamp. When a save version generated through migration is copied the copy is assigned the original creation date which corresponds to the SVID of the original. When a save version which has an original creation date is copied the copy is assigned the original creation date of the original.
- **Copying from a long-term archive to a migration archive**
  If the migration save file is lost, the copy in the long-term archive can be copied to the migration archive as a replacement. When a save version is copied the copy is assigned the original creation date of the original.
- **Copying from a long-term archive or migration archive to a backup archive where more than one save version is permitted per save file**
  The copy of the save version contains the same time stamp as the original. The time stamp of the save version copy must be higher than the highest time stamp of the target archive.
- **Copying from a backup archive or long-term to the associated shadow archive or from a shadow archive to the associated backup archive or long-term**
  As when copying between backup archives, an identical copy is created with the same time stamp. The time stamp does not have to be the highest of the target archive. The copy process is rejected, however, if the target archive already contains a save version with the same time stamp.
- **Copying from one version backup archive to another**
  Copying from version backup archive to the archive of another type is not possible and vice versa copying from archives of other types to a version backup archive will also be rejected. Moreover, copying between version backup archives is also not supported.

4.5.1.6 Permissible target storage media with the COPY-SAVE-FILE statement

The target storage medium when copying save files depends on the type of archive in which files which are to be copied are stored:

- **Copying from a backup archive**

  In an SM environment, copying can only take place to storage levels S1 and S2.

  In an SF environment, copying can take place to storage levels S1 and S2, to private disk, and to Net-Storage. The HSMS administrator can also copy to public disk.

  When only the last save version is selected, copying can take place to storage level S1 both from a backup archive containing multiple save versions and from a backup archive with SINGLE-SVID structure.

  When copying takes place from a backup archive containing multiple save versions to storage level S1, to public disk, or to private disk and not within the same archive, a separate save file is created for each save version of the original save file on the target medium.

- **Copying from a long-term archive**

  In an SM environment, copying can only take place to storage levels S1 and S2.

  In an SF environment, copying can take place to storage levels S1 and S2 and to Net-Storage. The HSMS administrator can also copy to public disk.

  Copying to private disk is not possible from a long-term archive.

  When copying takes place from a long-term archive containing multiple save versions to storage level S1, private disk, public disk, or Net-Storage, a separate save file is created for each save version of the original save file on the target medium.

  The COPY-SAVE-FILE statement only enables save files to be continued to magnetic tape cartridge.

- **Copying from a migration archive**

  Copying from migration archives can only take place to storage level S2.

4.5.1.7 Behavior of the COPY-SAVE-FILE statement in conjunction with save file names independent of VSNs

Copying to storage level S1, to public disk, or to Net-Storage is rejected if a save file with the same SFID already exists there.

4.5.1.8 Behavior of the COPY-SAVE-FILE ... TO-STORAGE=*S1-STORAGE-LEVEL command depending on the environments in which the original and the target archive are defined

The result of the COPY-SAVE-FILE statement with the TO-STORAGE=*S1-STORAGE-LEVEL specification depends on the environment, in which the original and the target archive are defined.

- If the target archive is defined in the SF environment, the S1 level definition is taken from the global HSMS parameters. A global S1 pubset is either a traditional SF pubset or an S1-SM pubset. For an S1-SM pubset all volume sets should be defined with USAGE=*STD and the save files can be saved to any volume set of this S1-SM pubset. Only the control volume set should not be used. This can be prevented by defining a respective restriction with the MODIFY-PUBSET-RESTRICTIONS statement.
- If the target archive is defined in the SM environment, the S1 level definition is taken from the local SM pubset parameters of the environment. In HSMS V11.0A and higher, the S1 level can be restricted to a volume set or distributed to all volume sets under HSMS control.

The following table shows the possible results of the COPY-SAVE-FILE ... TO-STORAGE=*S1-STORAGE-LEVEL statement depending on the environment of the respective archives:

- Original archive *SINGLE-FEATURE

Target archive environment	**S1 level in target environment** (global S1 level for SF environments)	**Result of the COPY-SAVE-FILE ... TO-STORAGE=*S1-STORAGE-LEVEL statement**
*SINGLE-FEATURE	NOT-DEFINED	This statement is rejected with the HSM0214 message.
*SINGLE-FEATURE	SF1	The statement is accepted. The copy of the save file is created on the SF1 SF pubset. SF1 may also be an S1-SM-pubset. In this case, the copy of the save file is stored on a USAGE=*STD volume set. File and job variable names remain the same, if not otherwise specified.
*SYSTEM-MANAGED(CATALOG-ID = SM1)	NOT-DEFINED	This statement is rejected with the HSM0214 message.
*SYSTEM-MANAGED(CATALOG-ID = SM1)	HSM1	The statement is accepted. The copy of the backup file is created on the HSM1 volume set of the SM1 SM pubset. Files and job variables are automatically renamed. The catalogue ID is changed in SM1.
*SYSTEM-MANAGED(CATALOG-ID = SM1)	*ALL-HSMS-CONTROLLED	The statement is accepted. The copy of the save file is created on one or more HSMS controlled volume sets of the SM1 SM pubset (depending on the capacity). Files and job variables are automatically renamed. The catalogue ID is changed in SM1.

- Original archive *SYSTEM-MANAGED(CATALOG-ID = SM1)

Target archive environment	**S1 level in target environment** (global S1 level for SF environments)	**Result of the COPY-SAVE-FILE ... TO-STORAGE=*S1-STORAGE-LEVEL statement**
*SINGLE-FEATURE	NOT-DEFINED	This statement is rejected with the HSM0214 message.
*SINGLE-FEATURE	SF1	The statement is accepted. The copy of the save file is created on the SF1 SF pubset. SF1 may also be an S1-SM-pubset. In this case, the copy of the save file will be stored on a USAGE=*STD volume set. If the target file is not a long-term archive, the renaming of files/job variables has to be explicitly determined: the catalogue ID must be changed. Otherwise, the statement is rejected with the HSM02A3 message.
*SYSTEM-MANAGED(CATALOG-ID = SM1)	NOT-DEFINED	This statement is rejected with the HSM0214 message.
*SYSTEM-MANAGED(CATALOG-ID = SM1)	HSM1/ *ALL-HSMS-CONTROLLED	The statement is accepted. The copy of the save file is created on HSM1 or on one or more HSMS controlled volume sets of the SM1 SM pubset (depending on the capacity). Note: if the original save file is also stored on the S1 level and //COPY-SAVE-FILE is to be copied from one backup archive to another, no copy is created. Reason: the time stamp of the original and the target file, and therefore their file names, are identical.
*SYSTEM-MANAGED(CATALOG-ID = SM2)	NOT-DEFINED	This statement is rejected with the HSM0214 message.
*SYSTEM-MANAGED(CATALOG-ID = SM2)	HSM2/ *ALL-HSMS-CONTROLLED	The statement is accepted. The copy of the save file is created on HSM1 or on one or more HSMS controlled volume sets of the SM1 SM pubset (depending on the capacity). Files and job variables are automatically renamed. The catalogue ID is changed in SM2.

4.5.2 Examples

This section provides examples dealing with the following subjects:

- combining save files
- moving save files from S1 to S2
- Moving node save files from S1 to S2

combining save files

Various archive operations have been performed to several save files on several cartridges. These save files are to be combined. Combination is only possible and feasible for migration and long-term archives.

The new save file still contains the original save versions.

```
/START-HSMS
//COPY-SAVE-FILE ARCH-NAME=$SYSHSMS.HSMS.AR.2BY, -   ————————————————————  (1)
//  S-F-ID=S.160812.142005,SAVE-F=*NEW, -
//  TO-STOR=*S2-STOR(VOL=*FROM-POOL), -
//  OPER-CONTROL=*PAR(REPORT=*FULL,OUT=HSMS.MAN.R.CSF.13, -
//    WAIT-F-C=*YES)
% HSM0003 HSMS STATEMENT COMPLETED
//COPY-SAVE-FILE ARCH-NAME=$SYSHSMS.HSMS.AR.2BY, -   ————————————————————  (2)
//  S-F-ID=S.160812.142018,SAVE-F=*CONT(S-F-ID=*LATEST), -
//  TO-STOR=*S2-STOR, -
//  OPER-CONTROL=*PAR(REPORT=*FULL,OUT=HSMS.MAN.R.CSF.14, -
//    WAIT-F-C=*YES)
% HSM0003 HSMS STATEMENT COMPLETED
//END
% HSM0014 HSMS PROGRAM TERMINATED
```

Report HSMS.MAN.R.CSF.13 (extract): ———————————————————————————— (3)

```
 *** COPY-SAVE-FILE                HSMS V12.0        FULL         REPORT  *** 2016-08-12
14:21:53      PAGE    3
REQUEST-NAME=CSF#0AAK REQUEST-DATE=2016-08-12 14:21:40 USER-ID=SYSHSMS  REQUEST-
STATE=COMPLETED   WITHOUT ERROR
                                     SAVE FILE IDENTIFIER - S.160812.142142
           SUBSAVE
            NUMBER       VSNS
               0       HSMS33
                                     SAVE FILE IDENTIFIER - S.160812.142142
                        ***   CATALOG - 2BY       USER - MANUAL     ***
** OUTPUT SAVE VERSION: SAVE-VERSION-DATE=16-08-12  SAVE-VERSION-TIME=14:21:43  *
FILE/JOB VARIABLE NAME                   LASTPG/  SAVE   INPUT DEV  SUB  OUTPUT
                                  VERS     SIZE  TYPE    VSN  TYP SAVE  VSN(S)
FILE.01                              1        3  FULL  HSMS11  T     0  HSMS33
FILE.02                              1        8  FULL  HSMS11  T     0  HSMS33
FILE.03                              1       13  FULL  HSMS11  T     0  HSMS33
FILE.11                              1        3  FULL  HSMS11  T     0  HSMS33
FILE.12                              1        8  FULL  HSMS11  T     0  HSMS33
FILE.13                              1       13  FULL  HSMS11  T     0  HSMS33
FILE.21                              1        3  FULL  HSMS11  T     0  HSMS33
FILE.22                              1        8  FULL  HSMS11  T     0  HSMS33
FILE.23                              1       13  FULL  HSMS11  T     0  HSMS33
 ***     E N D   O F                 HSMS V12.0        FULL         REPORT  *** 2016-08-12
14:21:53     ***
```

Report HSMS.MAN.R.CSF.14 (extract): ——————————————————————— (4)

```
 ***   COPY-SAVE-FILE                HSMS V12.0        FULL         REPORT  *** 2016-08-12
14:23:51    PAGE    3
REQUEST-NAME=CSF#0AAK REQUEST-DATE=2016-08-12 14:21:54 USER-ID=SYSHSMS  REQUEST-
STATE=COMPLETED WITHOUT ERROR
                                     SAVE FILE IDENTIFIER - S.160812.142142
            SUBSAVE
             NUMBER       VSNS
                0       HSMS33
                                     SAVE FILE IDENTIFIER - S.160812.142142
                       ***   CATALOG - 2BY       USER - MANUAL     ***
**  OUTPUT SAVE VERSION: SAVE-VERSION-DATE=16-08-12  SAVE-VERSION-TIME=14:21:57  *
           FILE/JOB VARIABLE NAME                   LASTPG/  SAVE   INPUT DEV  SUB  OUTPUT
                                             VERS     SIZE  TYPE    VSN  TYP SAVE  VSN(S)
FILE.04                                         1       18  FULL  HSMS22  T     0  HSMS33
FILE.05                                         1       16  FULL  HSMS22  T     0  HSMS33
FILE.06                                         1       11  FULL  HSMS22  T     0  HSMS33
FILE.14                                         1       18  FULL  HSMS22  T     0  HSMS33
FILE.15                                         1       16  FULL  HSMS22  T     0  HSMS33
FILE.16                                         1       11  FULL  HSMS22  T     0  HSMS33
FILE.24                                         1       18  FULL  HSMS22  T     0  HSMS33
FILE.25                                         1       16  FULL  HSMS22  T     0  HSMS33
FILE.26                                         1       11  FULL  HSMS22  T     0  HSMS33
**  OUTPUT SAVE VERSION: SAVE-VERSION-DATE=16-08-12  SAVE-VERSION-TIME=14:21:58  *
MAX-SIZE.1                                      1      277  FULL  HSMS22  T     0  HSMS33
MAX-SIZE.3                                      1    14998  FULL  HSMS22  T     0  HSMS33
 ***     E N D   O F                 HSMS V12.0        FULL         REPORT  *** 2016-08-12
14:23:51     ***
```

```
//MODIFY-ARCHIVE ARCH-NAME=$SYSHSMS.HSMS.AR.2BY, -  ————————————————————  (5)
//  SAVE-F=*DEL(S-F-ID=(S.160812.142005,S.160812.142018))
% HSM0003 HSMS STATEMENT COMPLETED
//SHOW-ARCHIVE ARCH-NAME=$SYSHSMS.HSMS.AR.2BY,SEL=*VOLUMES ————————————  (6)
```

```
SHOW-ARCHIVE (VOLUMES)
ARCHIVE-NAME = $SYSHSMS.HSMS.AR.2BY
VOLUME-STATE = ANY
------------------------------------------------------------------------------
  VSN      SFID              VOLUME-STATE    EXP-DATE   DEVICE     OWNER
  HSMS11                     AVAILABLE                  TAPE-C4    POOL
  HSMS22                     AVAILABLE                  TAPE-C4    POOL
  HSMS33   S.160812.142142   OBSOLETE        16-08-12   TAPE-C4    POOL
...
NEXT-PAGE: +  (+, -, ++, --, E)
%  HSM0012 END OF OUTPUT LIST REACHED
```

```
% HSM0003 HSMS STATEMENT COMPLETED
//END
% HSM0014 HSMS PROGRAM TERMINATED
```

(1) The save file identified by the SFID is copied to a new save file. The volume is taken from the archive pool.

(2) Another save file is copied. Specifying CONTINUE=*LATEST causes the save file created via the preceding COPY-SAVE-FILE statement to be continued. Both of the old save files are now in a save file on cartridge, as indicated by the reports.

(3) The report of the first copy requests is output.

(4) The report of the second copy requests is output.

The two reports list the save files and the files contained in them. Both reports have the same SFID.

(5) Both of the original save files are released.

(6) The HSMS11 and HSMS22 cartridges have been released. Only the HSMS33 cartridge to which data has just been written is still in use.

Moving save files from S1 to S2

In this example the backups of the DMS files are to take place first at S1 level instead of S2 level. These are regularly moved from S1 level to S2 level provided the save versions have already been residing at S1 level for more than three days.

Daily backup with the BACKUP-FILES statement:

```
//BCF FILE-NAMES=DMS. -
//   ,SELECT-FILES=*ALL-FILES -
//   ,ARCHIVE-NAME=MANUAL.DMS -
//   ,TO-STORAGE=*S1-STORAGE-LEVEL -
//   ,OPERATION-CONTROL=*PARAMETERS(REPORT=*FULL,OUTPUT=#2)
```

Overview of the save versions after three saves, before MOVE-SAVE-FILE

```
SHOW-ARCHIVE (SAVE-VERSIONS)           INFORMATION = SUMMARY
ENVIRONMENT  = SF                      ARCHIVE-NAME = $SYSHSMS.MANUAL.DMS
SV-NAME      = ANY                     SV-DATE      = INTERVAL EARLIEST LATEST
USER-ID      = OWN                     EXP-DATE     = ANY
--------------------------------------------------------------------------------
M SAV-DATE SAV-TIME  EXP-DATE  SFID              SEL-F  BC IND  USER-ID  SV-NAME
  16-05-12 16:41:02  16-05-14  S.160512.164102  ALL-F  D       SYSHSMS
  16-05-13 08:59:33  16-05-13  S.160513.085933  MOD-F  D       SYSHSMS
  16-05-14 15:50:13  16-05-14  S.160514.155013  MOD-F  D















--------------------------------------------------------------------------------
NEXT-PAGE : +   (+, -, ++, --, E)
%  HSM0012 END OF OUTPUT LIST REACHED
```

Moving all save versions that have been on S1 for more than three days

The MOVE-SAVE-FILES statement was executed on 16.05.16:

```
//MSF ARCHIVE-NAME=MANUAL.DMS -
// ,FROM-STORAGE=*DISK(MINIMUM-DAYS-ON-DISK=3) -
// ,OPERATION-CONTROL=*PARAMETERS(REPORT=*FULL,OUTPUT=#1)
```

Overview of the save versions after three saves, after MOVE-SAVE-FILE

```
SHOW-ARCHIVE (SAVE-VERSIONS)           INFORMATION = SUMMARY
ENVIRONMENT  = SF                      ARCHIVE-NAME = $SYSHSMS.MANUAL.DMS
SV-NAME      = ANY                     SV-DATE      = INTERVAL EARLIEST LATEST
USER-ID      = OWN                     EXP-DATE     = ANY
------------------------------------------------------------------------------
M SAV-DATE SAV-TIME  EXP-DATE  SFID             SEL-F  BC IND  USER-ID  SV-NAME
  16-05-12 16:41:03  16-05-12  S.160516.110025  ALL-F  D       SYSHSMS
  16-05-13 08:59:34  16-05-13  S.160516.110025  MOD-F  D       SYSHSMS
  16-05-14 15:50:13  16-05-14  S.160514.155013  MOD-F  D
------------------------------------------------------------------------------
NEXT-PAGE : +   (+, -, ++, --, E)
%  HSM0012 END OF OUTPUT LIST REACHED
```

Overview of the save files after MOVE-SAVE-FILE

```
SHOW-ARCHIVE (SAVE-FILES)              INFORMATION       = SUMMARY
ENVIRONMENT     = SF                   ARCHIVE-NAME      = $SYSHSMS.MANUAL.DMS
SAVE-FILE-STATE = ANY                  SAVE-FILE-STORAGE = ANY
CREATED-BEFORE  = LATEST               EXPIRATION-BEFORE = LATEST
------------------------------------------------------------------------------
M SFID             CREA-DATE  EXP-DATE  OBS  ACCESS ST  DEVICE    #VOL #SV #RUNS
  S.160514.155013  16-05-14   16-05-14  YES  OWNER  PUB                1    1
  S.160516.110025  16-05-16   16-05-16  YES  OWNER  TAP TAPE-C4   1    2    1
------------------------------------------------------------------------------
NEXT-PAGE : +   (+, -, ++, --, E)
%  HSM0012 END OF OUTPUT LIST REACHED
```

Moving node save files from S1 to S2

The example shows the moving of all S1 save files to S2 level for node files with shadow archive.

Rather than at S2 level, all the node files should first be saved at S1 level (for example because of hardware maintenance of the tape units). Subsequently all S1 saves are moved to S2 level. The archive is assigned a shadow archive.

Three saves are performed on public disk (S1 level) using the BACKUP-NODE-FILES statement:

```
//BNF PATH-NAMES=*PATH-NAME(PATH=/manual) -
//   ,ARCHIVE-NAME=MANUAL.NF -
//   ,SELECT-FILES=*MODIFIED-FILES -
//   ,TO-STORAGE=*PUBLIC-DISK(PUBSET-ID=DUB) -
//   ,OPERATION-CONTROL=*PARAMETERS(REPORT=*FULL,OUTPUT=#1)
```

Save versions and save files after three saves

- Main archive

```
SHOW-ARCHIVE (SAVE-VERSIONS)           INFORMATION = SUMMARY
ENVIRONMENT  = SF                      ARCHIVE-NAME = $SYSHSMS.MANUAL.NF
SV-NAME      = ANY                     SV-DATE      = INTERVAL EARLIEST LATEST
USER-ID      = OWN                     EXP-DATE     = ANY
------------------------------------------------------------------------------
M SAV-DATE SAV-TIME  EXP-DATE  SFID             SEL-F  BC IND  USER-ID  SV-NAME
  16-05-19 13:13:57  16-05-19  S.160519.131355  MOD-F          SYSHSMS
  16-05-19 13:14:43  16-05-19  S.160519.131441  MOD-F          SYSHSMS
  16-05-19 13:15:24  16-05-19  S.160519.131522  MOD-F          SYSHSMS
                                                                 
                                                                 
                                                                 
                                                                 
                                                                 
                                                                 
                                                                 
                                                                 
                                                                 
                                                                 
------------------------------------------------------------------------------
NEXT-PAGE : +   (+, -, ++, --, E)
%  HSM0012 END OF OUTPUT LIST REACHED
```

```
SHOW-ARCHIVE (SAVE-FILES)             INFORMATION       = SUMMARY
ENVIRONMENT     = SF                  ARCHIVE-NAME      = $SYSHSMS.MANUAL.NF
SAVE-FILE-STATE = ANY                 SAVE-FILE-STORAGE = ANY
CREATED-BEFORE  = LATEST              EXPIRATION-BEFORE = LATEST
------------------------------------------------------------------------------
M SFID             CREA-DATE  EXP-DATE  OBS  ACCESS ST  DEVICE    #VOL #SV #RUNS
  S.160519.131355  16-05-19   16-05-19  YES  OWNER  PUB                1    1
  S.160519.131441  16-05-19   16-05-19  YES  OWNER  PUB                1    1
  S.160519.131522  16-05-19   16-05-19  YES  OWNER  PUB                1    1









------------------------------------------------------------------------------
NEXT-PAGE : +   (+, -, ++, --, E)
%  HSM0012 END OF OUTPUT LIST REACHED
```

- Shadow archive
  The shadow archive is still empty. Saves to S1 level/public disk have not yet been taken over into the shadow archive!

The saves are moved to S2 level using the MOVE-SAVE-FILES statement:

```
//MSF ARCHIVE-NAME=MANUAL.NF -
//   ,ENVIRONMENT=*NODE-STD -
//   ,OPERATION-CONTROL=*PARAMETERS(REPORT=*SUMMARY,OUTPUT=#1)
```

Save versions and save files after the move

- Main archive

```
SHOW-ARCHIVE (SAVE-VERSIONS)           INFORMATION = SUMMARY
ENVIRONMENT  = SF                      ARCHIVE-NAME = $SYSHSMS.MANUAL.NF
SV-NAME      = ANY                     SV-DATE      = INTERVAL EARLIEST LATEST
USER-ID      = OWN                     EXP-DATE     = ANY
------------------------------------------------------------------------------
M SAV-DATE SAV-TIME  EXP-DATE  SFID             SEL-F  BC IND  USER-ID  SV-NAME
  16-05-19 13:13:58  16-05-19  S.160519.131806  MOD-F          UNDEF
  16-05-19 13:14:44  16-05-19  S.160519.131806  MOD-F          UNDEF
  16-05-19 13:15:25  16-05-19  S.160519.131806  MOD-F          UNDEF









------------------------------------------------------------------------------
NEXT-PAGE : +   (+, -, ++, --, E)
%  HSM0012 END OF OUTPUT LIST REACHED
```

```
SHOW-ARCHIVE (SAVE-FILES)              INFORMATION       = SUMMARY
ENVIRONMENT     = SF                   ARCHIVE-NAME      = $SYSHSMS.MANUAL.NF
SAVE-FILE-STATE = ANY                  SAVE-FILE-STORAGE = ANY
CREATED-BEFORE  = LATEST               EXPIRATION-BEFORE = LATEST
------------------------------------------------------------------------------
M SFID             CREA-DATE  EXP-DATE  OBS  ACCESS ST  DEVICE    #VOL #SV #RUNS
  S.160519.131806  16-05-19   16-05-19  YES  OWNER  TAP TAPE-C4   1    3    1









------------------------------------------------------------------------------
NEXT-PAGE : +   (+, -, ++, --, E)
%  HSM0012 END OF OUTPUT LIST REACHED
```

- Shadow archive

```
SHOW-ARCHIVE (SAVE-FILES)             INFORMATION       = SUMMARY
ENVIRONMENT     = SF                  ARCHIVE-NAME      = $SYSHSMS.MANUAL.NF.SH
SV-NAME =       = ANY                 SAVE-FILE-STORAGE = INTERVAL EARLIEST LATEST
USER-ID         = OWN                 EXP-DATE          = ANY
------------------------------------------------------------------------------
M SAV-DATE SAV-TIME  EXP-DATE  SFID             SEL-F  BC IND  USER-ID  SV-NAME
  16-05-19 13:13:58  16-05-19  S.160519.131806  MOD-F          UNDEF
  16-05-19 13:14:44  16-05-19  S.160519.131806  MOD-F          UNDEF
  16-05-19 13:15:25  16-05-19  S.160519.131806  MOD-F          UNDEF









------------------------------------------------------------------------------
NEXT-PAGE : +   (+, -, ++, --, E)
%  HSM0012 END OF OUTPUT LIST REACHED
```

```
SHOW-ARCHIVE (SAVE-FILES)             INFORMATION       = SUMMARY
ENVIRONMENT     = SF                  ARCHIVE-NAME      = $SYSHSMS.MANUAL.NF.SH
SAVE-FILE-STATE = ANY                 SAVE-FILE-STORAGE = ANY
CREATED-BEFORE  = LATEST              EXPIRATION-BEFORE = LATEST
------------------------------------------------------------------------------
M SFID             CREA-DATE  EXP-DATE  OBS  ACCESS ST  DEVICE    #VOL #SV #RUNS
  S.160519.131806  16-05-19   16-05-19  YES  OWNER  TAP TAPE-C4   1    3    1









------------------------------------------------------------------------------
NEXT-PAGE : +   (+, -, ++, --, E)
%  HSM0012 END OF OUTPUT LIST REACHED
```

4.6 Selection of BS2000 files, job variables and node files

The basic functions of HSMS are generally used to process large numbers of files and job variables. HSMS offers various options for selecting the files and job variables for processing.

4.6.1 Selection of BS2000 files

BS2000 files are files which are managed and edited in BS2000. Unless an explicit distinction is made below, in the case of files on Net-Storage, the term "BS2000 file" refers to both the file type BS2000 and the file type node file which is also supported in BS2000 OSD/BC V10.0 and higher.

BS2000 files can be selected by means of:

- various operands in the individual HSMS statements
- the HSMS statement SELECT-FILE-NAMES
- dialog input

4.6.1.1 Selection by means of operands

The files which are to be processed by a specific HSMS statement are selected by means of its FILE-NAMES and EXCEPT-FILE-NAMES operands. In the HSMS statement COPY-SAVE-FILE, the corresponding operands are SELECT-FILES and EXCEPT-FILES. The selection can be further restricted via additional operands. Some HSMS statements offer the option of confirming the final selection of files via a screen mask.

The simplest way to select files is to process all the files of a given user ID or, in the case of the HSMS administrator, all the files in the system by means of a single HSMS statement:

```
FILE-NAMES/SELECT-FILES=*OWN/*ALL
```

This is advisable especially when making full backups, but also whenever files are to be selected according to specific criteria (see below), instead of by file name.

When *OWN/*ALL is specified, shared pubsets are in many cases ignored if the home computer is a slave. Details are provided under the various HSMS statements.

Direct specification of file names

All HSMS statements that are used to process files permit the file names to be entered directly. Up to 20 files can be specified per HSMS statement. If more than 20 files are to be specified, the input should be made by means of a file (see “Specification of file names in a file” below).
The files may be specified using fully-qualified or partially-qualified names, and with or without specification of the catalog ID and user ID.

The use of wildcards is permitted; the syntax for wildcards is described in “HSMS Vol. 2” [1] under “Metasyntax”. When wildcards are resolved in the catalog ID, shared pubsets for which the home computer is a slave are in many cases completely ignored. Details are provided under the relevant HSMS statement.

By specifying partially-qualified catalog IDs or user IDs, it is easy to process entire pubsets and user IDs.

Nonprivileged users can also process files belonging to other user IDs if they are co-owners of these files (see the “SECOS” manual [16]).

Specification of file names in a file or library elements

It is also possible to pass the file names to HSMS in a list file, in which each file record contains the name of a file to be processed. This list file must be a SAM file with variable record length and contain uppercase letters only. Alternatively, a library element of type S can also be used. To create a input file, you can use for example the BS2000 command:

```
/SHOW-FILE-ATTRIBUTES ...,OUTPUT=<file name>(FORM-NAME=*FILE-NAME)
```

or the HSMS statement SELECT-FILE-NAMES. The procedure for creating a file with the HSMS statement SELECT-FILE-NAMES is described on "Selection by means of the HSMS statement SELECT-FILE-NAMES".

The file names in the file may be fully qualified or partially qualified, and with or without specification of the catalog ID and user ID. Only uppercase letters are permitted. The wildcard syntax (see the section "Metasyntax" in "HSMS Vol. 2” [1]) may be used, with the following restriction:
The file names in the file may not be prefixed with a hyphen to exclude them from processing.

The name of this file or the library element must be specified in the FILE-NAMES, EXCEPT-FILE-NAMES, SELECT-FILES, or EXCEPT-FILES operand of the relevant HSMS statement with *FROM-FILE(...) or *FROM-LIBRARY-ELEMENT(...). e.g.:

```
FILE-NAMES/SELECT-FILES=*FROM-FILE(LIST-FILE-NAME=<filename>) or
   *FROM-LIBRARY-ELEMENT(LIBRARY=<library>,ELEMENT=<element>)
```

For a nonprivileged user, the file or the library must be under his/her user ID or access must be permitted by co-ownership.

Restriction of selection via statement operands

Many HSMS statements permit the set of files defined by means of FILE-NAMES to be further qualified through the specification of additional operands: files selected by means of FILE-NAMES are not processed unless they also fulfill the conditions defined by the other operands.

Example

```
//BACKUP-FILES FILE-NAMES=*ALL,SUPPORT=*PRIVATE-DISK(...)
```

Instead of backing up all the files, HSMS backs up only those that reside on a private disk (which may also be specified).

All the default values of these qualifying operands have been defined in such a way as to ensure that they do not interfere with file selection by means of FILE-NAMES.

Selection of BS2000 files, job variables and node files

In BS2000/OSD-BC V9.0 and higher a pubset can be provided with additional storage space which a net server makes available as Net-Storage. This enables files from pubsets to reside either on local pubset disks or remotely on a Net-Storage. BS2000 OSD/BC V10.0 and higher supports both the file type BS2000 and node file on Net-Storage.

When backup takes place using BACKUP-FILES, you can specify in the save options (SAVE-OPTIONS) that data from Net-Storage files of the type BS2000 is not to be saved. The catalog entries of the files are saved independently of this.

Net-Storage files of the type node file are always saved with data irrespective of the specification in the SAVE-NET-STOR-DATA operand.

Example

```
//BACKUP-FILES FILE-NAMES=*ALL,...,
               SAVE-OPTIONS=*PAR(SAVE-NET-STOR-DATA=*NO)
```

When the save operation is restricted in the SUPPORT operand to files from shared disks, the save can also be restricted to files which reside on Net-Storage or to files which reside on local pubset disks.
In the case of Net-Storage the backup can also be restricted to the file type BS2000 or node file in the FILE-TYPE operand. By default all Net-Storage files are saved irrespective of the file type.

Examples

```
//BACKUP-FILES FILE-NAMES=*ALL,...,
             SUPPORT=*PUBLIC-DISK(STORAGE-TYPE=*PUBLIC-SPACE),...
```

When the save operation is restricted in the SUPPORT operand to files from shared disks, the save can also be restricted to files which reside on Net-Storage or to files which reside on local pubset disks.

```
//BACKUP-FILES FILE-NAMES=*ALL,...,
             SUPPORT=*PUBLIC-DISK(
                STORAGE-TYPE=*NET-STORAGE(VOLUME=(1#VBIG,2#VBIG)))
```

Only files which reside on Net-Storage volumes `1#VBIG` and `2#VBIG` are saved.

```
//BACKUP-FILES FILE-NAMES=*ALL,...,
             SUPPORT=*PUBLIC-DISK(
                STORAGE-TYPE=*NET-STORAGE(VOLUME=(1#VBIG,2#VBIG),
                                  FILE-TYPE=*BS2000))
```

Only files of the type BS2000 which reside on Net-Storage volumes `1#VBIG` and `2#VBIG` are saved. Net-Storage files of the type node file which may exist in user-specific directories on these volumes are not saved.

This selection option is also available in the EXPORT-FILES, IMPORT-FILES, and RESTORE-FILES statements (in the ORIGINAL-SUPPORT or NEW-SUPPORT operand).

Selecting file generations

HSMS backs up file generations as well as file generation groups. File generation groups can be specified in the same way as files; all generations of the group will then be backed up. If a specific generation is to be processed, the specified file name must be fully-qualified (with generation number).

4.6.1.2 Selection by means of the HSMS statement SELECT-FILE-NAMES

The HSMS statement SELECT-FILE-NAMES creates a list of path names that can be used in the subsequent operands to select BS2000 files:

```
FILE-NAMES/SELECT-FILES=*SELECTED
```

Unless otherwise specified, this list is written to a temporary file that is deleted once the task has been concluded.

Alternatively, this list can be written to a cataloged file, which can then be specified as a list file:

```
FILE-NAMES/SELECT-FILES=*FROM-FILE(LIST-FILE-NAME=<filename>)
```

When handled in this way, the generated list can be used not just once, during a single HSMS session, but as often as desired.

SELECT-FILE-NAMES can be used to select files:

- from the catalog of a pubset in preparation for backup, archiving or migration or when exporting files
- from an archive directory when restoring or recalling files or from a directory when importing files.

In addition to the FILE-NAMES operand, a whole range of selection criteria can be used to select files in a catalog or archive directory.

Selection from a catalog

Files from a pubset catalog are selected by means of the SELECT-FROM= *CATALOG operand. The following selection criteria apply:

- the volume on which they reside
- the storage type (*PUBLIC-SPACE or *NET-STORAGE)
- the file type of Net-Storage files (*BS2000 or *NODE-FILE)
- the storage level at which they are managed
- the period of time since the last file access
- the file size
- the backup class.

Selection from an archive

Files can be selected from an archive by means of the operand SELECT-FROM= *ARCHIVE. The following selection criteria apply:

- the archive in which they are managed
- the save version in which they have been saved
- the file expiration date
- their save type.

SELECT-FILE-NAMES can also be used to select files in a dialog.

4.6.1.3 Selection in a dialog

File selection in a dialog is available only for BS2000 files and only at a block-oriented data display terminal (e.g. 9750).
In the case of both SELECT-FILE-NAMES and also some action statements, a further selection can be made in the list which was generated according to the criteria specified: the files in the list are presented in screen masks when DIALOG-FILE-SELECT=*YES. The files can be highlighted to select or exclude them individually or by the screenful.

Example

Screen mask display for DIALOG-FILE-SELECT

```
SELECT-FILE-NAME (FROM-CATALOG)                          #FILES       = 27
STORAGE-LEVEL = ANY             UNUSED-DAYS  = 7         MINIMUM-SIZE = NONE
SUPPORT       = ANY             BACKUP-CLASS = ANY       MAXIMUM-SIZE = NONE
------------------------------------------------------------------------------
M  FILE-NAME                                              UNUSED   #PAGES  ST BC
X  :T:$TESTUSER.FILE.TEST.1                                    8       31  S0  A
X  :T:$TESTUSER.FILE.TEST.3                                   12       43  S0  A
X  :T:$TESTUSER.LIB.TEST.2                                    13      523  S0  A
X  :T:$TESTUSER.LIB.TEST.5                                     7      267  S0  A
X  :T:$TESTUSER.LIB.TEST.7                                    23     1290  S0  A
X  :T:$TESTUSER.LIB.TEST.10                                    9      101  S0  A
X  :T:$TESTUSER.PROC.TEST.3                                    7        7  S0  A
X  :T:$TESTUSER.PROC.TEST.6                                   21        3  S0  A
X  :T:$TESTUSER.PROG.TEST.1                                   17      198  S0  A
X  :T:$TESTUSER.PROG.TEST.2                                   11      353  S0  A
X  :T:$TESTUSER.PROG.TEST.2                                   11      353  S0  A
X  :T:$TESTUSER.SYNTAX.TEST.GROUP                             15      333  S0  A
X  :T:$TESTUSER.SYNTAX.TEST.USER                              15      145  S0  A
X  :T:$TESTUSER.TEST.1                                        12      232  S0  A
X  :T:$TESTUSER.TEST.12                                       37       87  S0  A
X  :T:$TESTUSER.TEST.14                                       34      102  S0  A

------------------------------------------------------------------------------
NEXT-PAGE : __  (+, -, ++, --, E)    MARK :    (A: ALL, N: NONE)
```

Key:

Column – Values	**Meaning**
M	Mark column (character = file is selected, blank = file is not selected)
FILE-NAME	Path name of the file
UNUSED	Number of days since the file was last accessed
#PAGES	File size in PAM pages (FILE-SIZE)
ST – S0, S1, S2	Storage level at which the file is located – possible storage levels
BC – A, B, C, D, E	File backup class – possible backup classes

Selection by marking

A file is regarded as selected if it is marked in the mask by a character in the mark column (M column). It is regarded as exempted if it is not marked (blank in the mark column).

When calling DIALOG-FILE-SELECT, files are listed in FHS masks but not selected.

By entering “A” in the MARK field in the bottom line all files are selected, even if they are not displayed on the screen.

As an alternative, “N” may be entered in the MARK field. This excludes all files from selection, i.e. all files displayed in the mask are marked with a blank in the mark column. The desired files can then be marked with any character other than a blank (positive selection). You can reset all the entries you have made thus far by entering “N” or “A”.

The contents of the screen when you terminate the function by entering “E” in the NEXT-PAGE field determine which files are selected.

Positioning by means of a search string

Users can scroll up and down within the mask by entering “ + ”, “ – ”, etc. Alternatively, a search string of up to 10 characters enclosed in single quotes can be entered in the NEXT-PAGE field in order to position to specific files.

The search string entered as a positioning criterion may be either a fully or partially qualified path name without wildcards, or part of a path name (catalog ID and/or user ID and/or file name). Any leading parts of the path name which the user omits are adopted from the corresponding parts of the path name of the first file displayed in the current mask. HSMS appends a wildcard to the search string to ensure that longer path names will also be found. If the specified (or extended) path name does not exist, the cursor is moved to the next file name in the sorting sequence.

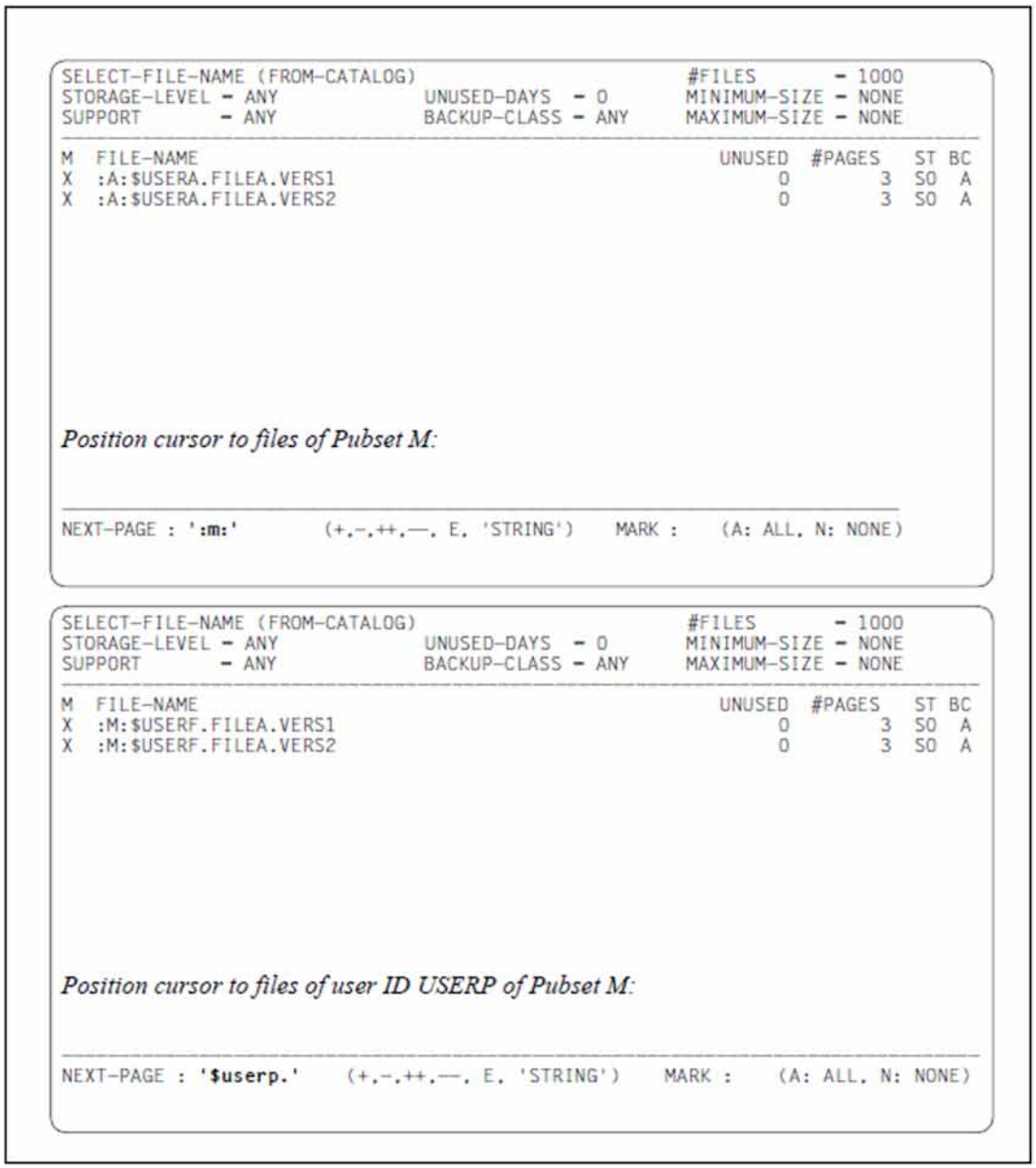
For example, specification of a catalog ID positions the cursor to the first file with that catalog ID or to the next existing catalog ID in the sorting sequence.

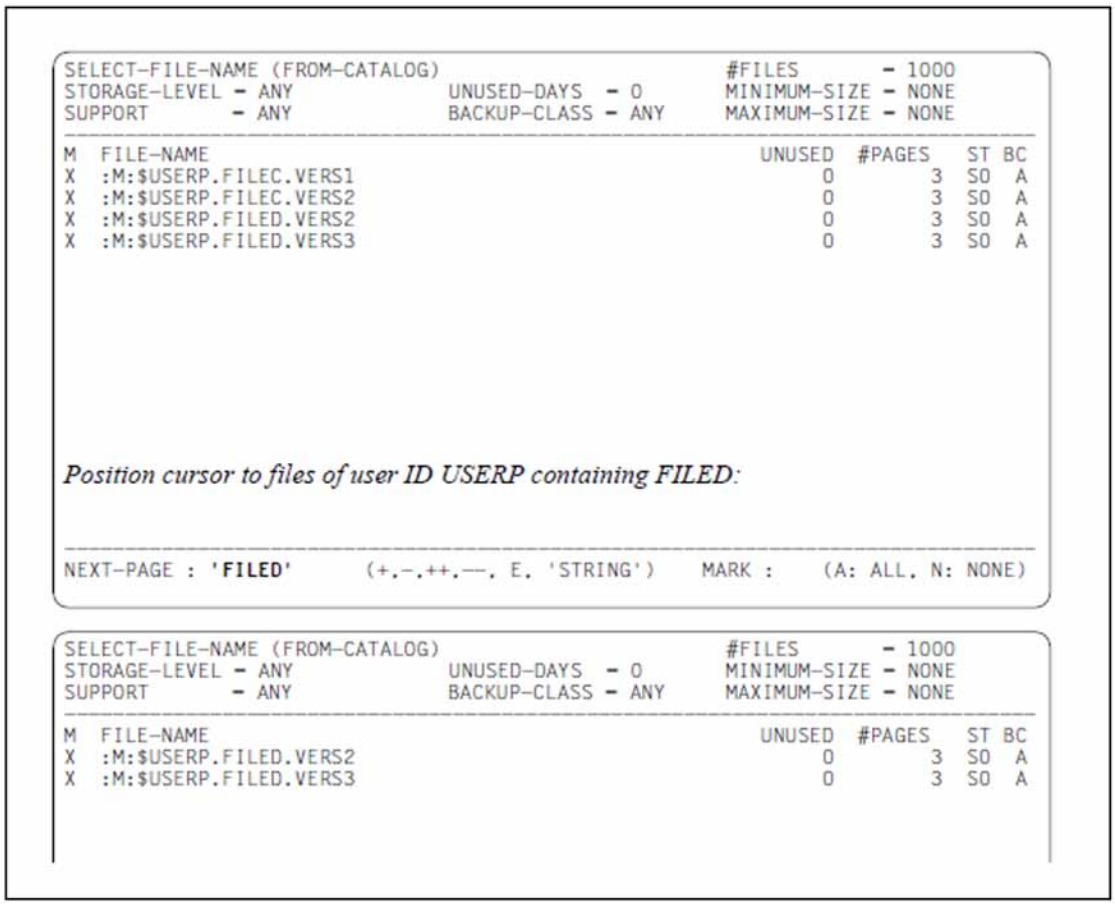
Specification of a user ID positions the cursor to a file which has that user ID and the catalog ID of the first file displayed in the current mask. The user ID must be entered with “$” followed by a period. Otherwise, HSMS will perform a secondary read operation to the default user ID.

The cursor can be positioned to individual files by entering the first characters of their name (without the period). This initiates a search for a file that has both the specified file name and the user ID and catalog ID of the first file displayed.

Example

The cursor is to be positioned to a file beginning with the string “FILED”, belonging to the user USERP and residing on pubset M. This is carried out in several stages.





Even though the search string is short, the cursor can be positioned to the desired file by successively entering the catalog ID, the user ID and parts of the name.

4.6.2 Selection of job variables

Job variables can be selected by means of:

- the operands JV-NAMES and EXCEPT-JV-NAMES in statements for backing up and exporting save files
- the operands SELECT-JV and EXCEPT-JV in statements for copying save files

It is also possible to specify either all job variables (*OWN/*ALL), no job variables (*NONE) or a list of job variables (*FROM-FILE(...) or *FROM-LIBRARY-ELEMENT(...)).

If *FROM-FILE(...) or *FROM-LIBRARY-ELEMENT(...) is used to specify a list of job variables, the user must create a SAM file or a library element of type S containing the names of job variables which are to be processed.

- Users can create this file or the library element themselves outside of HSMS, e.g. with the EDT

  One job variable is to be specified per file record. The use of wildcards is permitted, subject to the limitation that the convention for excepting files from processing, i.e. prefixing their names with a hyphen, must not be used.
- With the HSMS statement SELECT-JV-NAMES users can create a SAM file with the required job variable names.

4.6.3 Selection of node files of the BS2000-UFS

Node files of the BS2000-UFS or node S0 can be selected by:

- various operands in the individual HSMS statements
- the HSMS statement SELECT-NODE-FILES

4.6.3.1 Selection by means of operands

The files which are to be processed by a specific HSMS statement are selected by means of its PATH-NAMES and EXCEPT-PATH-NAMES operands. In the HSMS statement COPY-NODE-SAVE-FILE, the corresponding operands are SELECT-PATHS and EXCEPT-PATHS. The selection can be further restricted via additional operands. Some HSMS statements offer the option of confirming the final selection of files via a screen mask.

Direct specification of file names

In all HSMS statements that are used to process node files, users can specify the path name of the node files which are to be selected. The path name may contain wildcards. Only a single path name may be specified in an HSMS statement. If it is necessary to specify two or more node files, the input should be made by means of a file (see "Specification of file names in a file or library elements").

Users can specify the path names of node files which are to be excluded from processing in the same way.

Nonprivileged users are permitted to specify only node files residing on the local BS2000-UFS. The HSMS administrator is authorized to specify any node files which reside on the local BS2000-UFS or remote workstation.

If files and directories of a specific hierarchical level are to be selected, the user can also specify whether the scope of the selection process is to be limited to the current file system or to include the entire file tree.

In order to obtain any of the numbered results shown in the diagram below, the user must make the specifications shown in the table following the diagram for the path name and the SELECTION-BOUNDARY operand.

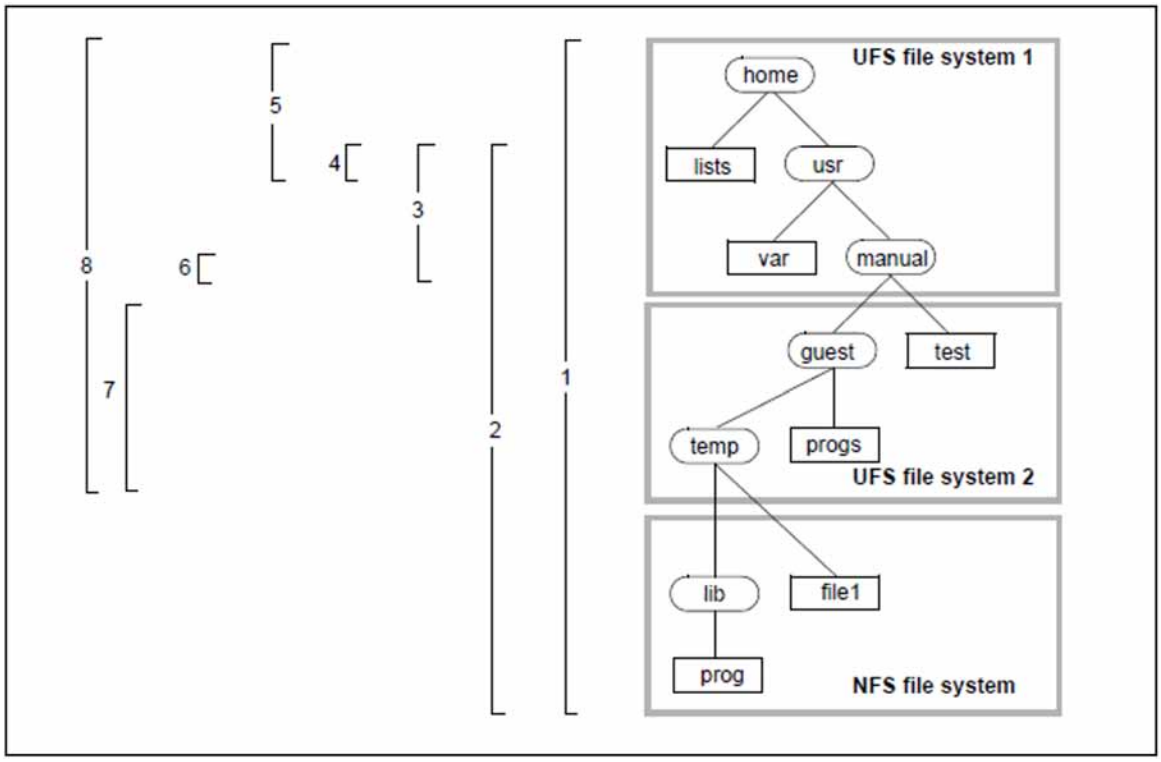


Figure 18: Scope of a node file save operation

Number	PATH-NAMES=	SELECTION-BOUNDARY=
1	/home or	*ALL-FILE-SYSTEMS
	/home*	*ALL-FILE-SYSTEMS
2	/home/*	*ALL-FILE-SYSTEMS
3	/home/*	*CURRENT-FILE-SYSTEM
4	/home/*	*SPECIFIED-PATHS
5	/home or	*ALL-FILE-SYSTEMS
	/home*	Additional specification: EXCEPT-PATH-NAMES=/home/usr/*
6	/home/usr/*	*CURRENT-FILE-SYSTEM
7	/home/usr/manual/*	*ALL-FILE-SYSTEMS
8	/home	*ALL-LOCAL-FILE-SYSTEMS

In the selection strategy presented above, the specification of *ALL-FILE-SYSTEMS is recommended for saving the files of all remote computers, in particular where more than one file system is mounted under the root directory (“/”).

The software product NFS is used to access node files on a passive workstation. Therefore, the names of remote files are extended internally in the following way:

```
/HSMS/<node-id>/<pathname>
```

The total length of any file name, including the internal extension, must not exceed 1023 characters. This restriction should be borne in mind particularly when restoring and renaming files. It may also have the effect of rendering inaccessible any remote node files whose names, including the prefix “/HSMS/<node-id>”, exceed the permissible length of 1023 characters, even though the user has specified a shorter file name (directory level, recursive selection, wildcards, ...).

Specification of file names in a file or library elements

It is also possible to pass the file names to HSMS in a BS2000 file or a library element of type S. These contain, record by record, the path names of the node files or node file groups to be processed.

This BS2000 file must be a SAM file with variable-length records, which can be created with, for example, the BS2000 editor EDT or with the HSMS statement SELECT-NODE-FILES. The procedure for creating a file with the HSMS statement SELECT-NODE-FILES is described in "Selection by means of the HSMS statement SELECT-NODE-FILES".

The path names can be specified in the file as follows

No.	Path	References	access rights	
			HSMS administrator	Nonpriviliged users
1.	/<dir>/... where <dir> != “HSMS”	local BS2000-UFS	X	X
2.	<node-id>:/...	specified workstation	X	–
3.	/ * ...	local BS2000-UFS	X	X
4.	* : /...	all workstations	X	–

The path names specified with FROM-FILE must not be prefixed with a hyphen to exclude them from processing.

The name of this file the library element must be specified in the relevant HSMS statement in the PATH-NAMES or SELECT-PATHS operand with *FROM-FILE(...) or *FROM-LIBRARY-ELEMENT(...):

```
PATH-NAMES/SELECT-PATHS=*FROM-FILE(LIST-FILE-NAME=<file name>) or
   *FROM-LIBRARY-ELEMENT(LIBRARY=<library>,ELEMENT=<element>)
```

The paths are processed in the sequence in which they are entered in the file specified in *FROM-FILE. As a result, users can improve performance when backing up node files by using parallel processing (either by activating multiplexing operation or by using multiple drives).
Here is an example:

If a network consists of a number of separate, independent subnets, it may be useful to mix workstations from the various subnets in order to separate the data paths that are used by the ARCHIVE subtasks.
In concrete terms, we could find the following situation: workstations W1 and W2 belong to subnet S1 which is connected to server S via HNC1. Workstations W3 and W4 belong to a subnet S2 which is also connected to server S but via HNC2. The four workstations are to be backed up using two ARCHIVE subtasks (with or without multiplexing operation).

If the workstations are entered in the file in the order W1, W2, W3, W4 then the two ARCHIVE subtasks will simultaneously be working with the same subnet, which is not the perfect solution. The better alternative – as far as the separation of the data paths is concerned – is to specify the sequence W1, W3, W2, W4. The same considerations are also applicable at path level (multiple paths per workstation, multiple workstations).

The sequence in which the workstations are entered in the file specified in *FROM-FILE is therefore very important. The structure of this file should therefore be carefully considered. Furthermore, the paths should be chosen in a way that balances the data volumes.

See also section "Multiplexing operation" in "Parallel and serial processing in ARCHIVE".

4.6.3.2 Selection by means of the HSMS statement SELECT-NODE-FILES

HSMS provides the statement SELECT-NODE-FILES as a means of selecting the path names of nodes. This HSMS statement creates a list of path names that can be used in the subsequent operands to select node files:

```
PATH-NAMES/SELECT-PATHS=*SELECTED
```

Unless otherwise specified, this list is written to a temporary file that is deleted once the task has been concluded. Alternatively, this list can be written to a cataloged file, which can then be specified as a list file:

```
PATH-NAMES/SELECT-PATHS=*FROM-FILE(LIST-FILE-NAME=<file name>)
```

When handled in this way, the generated list can be used not just once, during a single HSMS session, but as often as desired.

SELECT-NODE-FILES can be used to select node files from an archive directory for restoration. In addition to the PATH-NAMES operand, a whole range of selection criteria can be used to select node files from an archive directory.

Selection from an archive

Files can be selected from an archive by means of the operand SELECT-FROM= *ARCHIVE. The following following selection criteria apply:

- the archive in which they are managed
- the save version in which they have been saved
- the file expiration date
- their save type.

4.7 Request handling

This section provides information on

- Action statements, requests and HSMS server tasks
- consistency checks
- Request-time control
- operation control
- Request management
- Request management for shared pubsets
- Recovery of requests after a host crash
- assignment of priorities for job processing
- Collector requests
- Asynchronous and synchronous processing
- device reservation
- parallel and serial processing

4.7.1 Action statements, requests and HSMS server tasks

A number of HSMS statements result in I/O operations on user files and save files and these are executed in an HSMS subtask (not in the calling task) and require some action on the part of the software product ARCHIVE. These HSMS statements are referred to as action statements:

ARCHIVE-FILES (ARF)	Archive files
ARCHIVE-NODE-FILES (ANF)	Archive node files
BACKUP-FILES (BCF)	Save files and job variables
BACKUP-FILE-VERSIONS (BFV)	Save files
BACKUP-NODE-FILES (BNF)	Save node files
COPY-EXPORT-SAVE-FILE (CES)	Copy save file
COPY-NODE-SAVE-FILE (CNF)	Copy node save files
COPY-SAVE-FILE (CSF)	Copying save files
EXPORT-FILES (EXF)	Export files and job variables
IMPORT-FILES (IMF)	Import files and job variables
MIGRATE-FILES (MGF)	Migrate files
MOVE-SAVE-FILES (MSF)	Move save file
RECALL-MIGRATED-FILES (RMF)	Retrieve files
REORGANIZE-VERSION-BACKUP (RVB)	Reorganization of version backup archives
REPAIR-CATALOG-BY-RESTORE (RCR)	Repair inconsistencies in migrated files using RESTORE
REPLACE-SAVE-FILE-BY-RESTORE (RFR)	Replace save file in migration archive using RESTORE
RESTORE-FILES (RSF)	Restore files and job variables
RESTORE-LIBRARY-ELEMENTS (RLE)	Restore elements of a library file
RESTORE-NODE-FILES (RNF)	Restore node files
UPDATE-EXPORT-SAVE-FILE (UES)	Update backup

Each action statement includes an OPERATION-CONTROL operand which serves to control the processing of the requests.

HSMS generates a request for each I/O operation initiated by any of these action statements. The requests are processed by user tasks specifically designed for this purpose, the HSMS server tasks. With the exception of the data transfer function, the HSMS server tasks are given the job name “HSMSSERV” and run under the TSOS user ID; they are present throughout the HSMS session.

These HSMS server tasks call ARCHIVE, which in turn creates ARCHIVE subtasks for processing the inputs /outputs. The ARCHIVE subtasks are given the job name in accordance with the specification in the REQUEST-NAME operand (see section "Operation control").

4.7.2 Request-time control

Request-time control is concerned with how a request created by a user task is forwarded to an HSMS server task. In request-time control – apart from the forwarding of a request from one task to another – the following conditions, for example, must be taken into account:

- Serial processing of requests that were issued for the same archive:
  some requests require exclusive access to the specified archive.
- Serial processing of requests which write to the same save file of an archive:
  as this concerns the same volumes, serial processing is mandatory here.
- Optimization, e.g. tape handling via collector requests.
- Parameters for a tape processing time which control access to the tape (storage level S2).

There are requests which do not need to be processed serially. These are requests for the restoration and recall of files, as well as requests which create a save file on S1. Such requests are processed completely in parallel if they restore or recall no files from S2 or from the same SFID of the same magnetic tape cartridge. The first free HSMS server task receives the request and processes it.

Requests which affect backups or node backups can only continue the last save file of the archive concerned or create a new save file, as the sequence of the save files (SFIDs) and save versions (SVIDs) must be guaranteed. When an attempt is made to continue a save file other than the last one, a new save file is created. Such requests are processed serially. When a server task accepts the request of an archive to be processed serially, it receives all requests which concern this archive. The server task calls ARCHIVE for all these requests. ARCHIVE ensures that they are processed serially.

Requests which concern the other archive types are processed as follows:

- in parallel if they write to different save files
- serially if they write to the same save file.

If a server task accepts the request of such a “serial SFD” archive, the server task locks the relevant output save file. All other requests affecting the same archive and the same save file are processed by the same server task, which ensures the correct serial processing.

The number of output save files that can be processed in parallel is physically limited to eight different save files per archive.

If a server task accepts several requests, it can create a collector request in order to optimize the tape processing (see section "Collector requests").

The following tables show the type of processing for every possible request combination issued for the same archive:

	BCF	**RSF/RLE**	**ARF**	**BFV**	**MGF**	**RMF**	**CSF/MSF**	**RVB**
BCF	parallel	parallel	not relevant	not relevant	not relevant	not relevant	parallel	not relevant
RSF/RLE		parallel	parallel	parallel	parallel	parallel	parallel	parallel
ARF			parallel	not relevant	not relevant	not relevant	parallel	not relevant
BFV				serial (1)	not relevant	not relevant	not relevant	serial (2)
MGF					parallel	parallel	parallel	not relevant
RMF						parallel	parallel	not relevant
CSF/MSF							parallel	not relevant
RVB								serial (2)

Table 1: Processing of DMS requests on S1

BCF BACKUP-FILES

RSF RESTORE-FILES

RLE RESTORE-LIBRARY-ELEMENTS

BFV BACKUP-FILE-VERSIONS

ARF ARCHIVE-FILES

MGF MIGRATE-FILES

RMF RECALL-MIGRATED-FILES

CSF COPY-SAVE-FILE

MSF MOVE-SAVE-FILES

RVB REORGANIZE-VERSION-BACKUP

(1) With a BACKUP-FILE-VERSIONS statement the sequence of the versions of file must be guaranteed. Therefore parallel processing is not possible.

(2) During reorganization process perform of other write requests is prohibited. Therefore parallel processing is not possible.

	BCF	**RSF / RLE**	**ARF**	**BFV**	**MGF**	**RMF**	**CSF/MSF**	**RVB**
BCF	serial (1)	parallel	not relevant	not relevant	not relevant	not relevant	serial (3)	not relevant
RSF / RLE		parallel	parallel	parallel	parallel	parallel	via SFID (2)	parallel

ARF			via SFID (2)	not relevant	not relevant	not relevant	via SFID (2)	not relevant
BFV				serial (4)	not relevant	not relevant	not relevant	serial (5)
MGF					via SFID (2)	parallel	via SFID (2)	not relevant
RMF						parallel	parallel	not relevant
CSF / MSF							via SFID (2)	not relevant
RVB								serial (5)

Table 2: Processing of DMS requests on S2

BCF BACKUP-FILES

RSF RESTORE-FILES

RLE RESTORE-LIBRARY-ELEMENTS

BFV BACKUP-FILE-VERSIONS

ARF ARCHIVE-FILES

MGF MIGRATE-FILES

RMF RECALL-MIGRATED-FILES

CSF COPY-SAVE-FILE

MSF MOVE-SAVE-FILES

RVB REORGANIZE-VERSION-BACKUP

(1) With a backup or node backup the sequence of the SFIDs and SVIDs must be guaranteed for partial backups. Therefore parallel processing is not possible.

(2) Requests which affect the same save file are processed serially. Requests which affect different save files are processed in parallel.

(3) With a COPY-SAVE-FILE statement for a backup archive or node backup archive, the sequence of the SFIDs and SVIDs must be guaranteed. Therefore parallel processing is not possible.

(4) With a BACKUP-FILE-VERSIONS statement the sequence of the versions of file must be guaranteed. Therefore parallel processing is not possible.

(5) During reorganization process performing of other write requests is prohibited to guarantee the consistency of the directory. Therefore parallel processing is not possible.

	BNF	**RNF**	**ANF**	**CNF/MSF**
BNF	serial (1)	parallel	not relevant	serial (3)

RNF		parallel	parallel	parallel
ANF			via SFID (2)	via SFID (2)
CNF/MSF				via SFID (2)

Table 3: Control of node requests

BNF BACKUP-NODE-FILES

RNF RESTORE-NODE-FILES

ANF ARCHIVE-NODE-FILES

CNF COPY-NODE-SAVE-FILE

MSF MOVE-SAVE-FILES

(1) With a backup or node backup the sequence of the SFIDs and SVIDs must be guaranteed for partial backups. Therefore parallel processing is not possible.

(2) Requests which affect the same save file are processed serially. Requests which affect different save files are processed in parallel.

(3) With a COPY-NODE-FILE statement for a backup archive or node backup archive, the sequence of the SFIDs and SVIDs must be guaranteed. Therefore parallel processing is not possible.

In a COPY-SAVE-FILE request two archives are involved; sequencing takes place in accordance with the target archive. The source archive’s directory remains open during execution of the copy request and can, during this time, block other requests (for example restore or backup requests) for the same archive.

Warning

Under certain circumstances, requests to be processed in parallel can lead to unexpected results. You should therefore pay particular attention to the following:

- A restore request for files which are currently being processed by a backup request can either restore the previous status of the files from the preceding backup, or their current status from the backup that is currently running.
- After an SM pubset is imported or the HSMS subsystem is started, it may be the case that when previously accepted requests are reactivated the original order of job acceptance is not taken into consideration.
- The parallel processing of requests – mainly during the restore – allows different server tasks to use the same save file. Due to the reservation of the volumes, however, the requests are actually processed serially.
- If save files are rescued with the MODIFY-ARCHIVE statement, exclusive access to the archive directory is necessary. Such a rescue operation is therefore rejected if other requests are accessing the archive. By the same token, execution of requests is suspended as long as an archive is being rescued.

4.7.3 Operation control

The OPERATION-CONTROL operand serves to control request processing. Some of its values are preset by the archive definition (except for data transfer). It is, however, possible to modify these presettings with the action statement which generates the request.

PARALLEL-RUNS

Number of ARCHIVE subtasks for a save request running in parallel. This enables save times to be considerably reduced, especially for system save runs, as HSMS saves the files of different user IDs to different tape devices.

If preprocessing/postprocessing is activated (PRE-POST-PROCESSING operand) in the case of operations involving the BACKUP-NODE-FILES and RESTORE-NODE-FILES statements, the ARCHIVE subtasks await the results of the preprocessing action before they commence data exchange with the workstation. However, this results in degraded performance.

If multiplexing operation is not active (see section "Multiplexing operation"), there must be one tape device per task when writing to or reading from S2 in order to accelerate processing (for the conditions see the section "Parallel and serial processing in ARCHIVE" as well as in the “ARCHIVE” manual [2]).
If multiplexing operation is active, the system calculates the number of ARCHIVE subtasks. The number of tape devices is taken from the user specification (see section "Multiplexing operation").

Two tape devices per task must be available when save files are duplicated from S2 to S2 or automatically to a shadow archive.

When save files are written to Net-Storage, a corresponding number of Net-Storage volumes must be specified. This specification depends on the number of save tasks running simultaneously.

Each time a save file is read (restored, recalled, duplicated), no more ARCHIVE subtasks can be executed than were used during writing to this save file.
If during reading of a save file fewer parallel runs are used than when writing to this save file, the tapes may be rewound to the beginning-of-tape marker.

WRITE-CHECKPOINTS

This operand determines whether checkpoints are to be written to the ARCHIVE checkpoint file during processing by ARCHIVE in order to provide for a restart following an interrupt (INTERRUPTED state); see section "Restarting requests".

OPERATOR-INTERACTION

This operand regulates how messages which require an operator response are to be handled. If the messages are not to be output to the console, HSMS performs default handling (see the description of the PARAM statement in the “ARCHIVE” manual [2]).

REQUEST-NAME

The REQUEST-NAME operand serves to assign a request a symbolic name. This name can then be used in the HSMS statements listed below to refer to the request. HSMS internally expands the name to include the caller’s user ID and a time stamp. This means that the name does not have to be unique. The HSMS statements for request management may refer to various requests using the same name.

The request name can be thought of as the HSMS job name. It is used as the job name for ARCHIVE subtasks. If the user does not define any name, the requests are assigned default names formed by an invariable short code plus the task sequence number (TSN) of the user task.

Action statement	Standard name of the request
ARCHIVE-FILES	ARF#yyyy
ARCHIVE-NODE-FILES	ANF#yyyy
BACKUP-FILES	BCF#yyyy
BACKUP-FILE-VERSIONS	BFV#yyyy
BACKUP-NODE-FILES	BNF#yyyy
COPY-EXPORT-SAVE-FILE	CES#yyyy
COPY-NODE-SAVE-FILE	CNF#yyyy
COPY-SAVE-FILE	CSF#yyyy
EXPORT-FILES	EXF#yyyy
IMPORT-FILES	IMF#yyyy
MIGRATE-FILES	MGF#yyyy
MOVE-SAVE-FILES	MSF#yyyy
RECALL-MIGRATED-FILES	RMF#yyyy
REPAIR-CATALOG-BY-RESTORE	RCR#yyyy
REORGANIZE-VERSION-BACKUP	RVB#yyyy
REPLACE-SAVE-FILE-BY-RESTORE	RFR#yyyy
RESTORE-FILES	RSF#yyyy
RESTORE-LIBRARY-ELEMENTS	RLE#yyyy
RESTORE-NODE-FILES	RNF#yyyy
UPDATE-EXPORT-SAVE-FILE	UES#yyyy

Table 4: Action statements and standard request names

REQUEST-DESCRIPTOR

Request submitters can use this operand to specify a text of their choice to describe the request in more detail. This text is output at the system console when the request is started. It can be viewed using the HSMS statement SHOW-REQUESTS.

The REQUEST-DESCRIPTOR operand is a simple way for job submitters to send system administrators information about a request. The system administrator can then take suitable measures for special requests.

However, it is not possible to reference these operands as selection criteria in subsequent statements, for example in the HSMS statement DELETE-REQUESTS, RESTART-REQUESTS or SHOW-REQUESTS.

EXPRESS-REQUEST

The HSMS administrator can start requests as express requests which can be controlled independently of the requests of nonprivileged users.
Express requests are reactivated during the tape session control defined especially for them. The tape session controls for express requests are separate from the normal tape session controls for write/read requests. All urgent requests should therefore be started as express requests and tape session controls should be defined for express requests that are called earlier or more frequently than the normal tape session controls.

Moreover, express requests are accorded a higher processing priority than normal requests issued for the same archive (see section "Assigning priorities for request processing").

WAIT-FOR-COMPLETION

Users can determine whether they wish to wait until HSMS has processed the entered statement in the background or whether they can make the next entry immediately (see section "Asynchronous and synchronous processing").

4.7.4 Request management

The following HSMS statements are available for request management:

- DELETE-REQUESTS: delete requests from the request file
- RECOVER-REQUESTS: recover requests in the SM environment
- RESTART-REQUESTS: restart interrupted requests
- SHOW-REQUESTS: output requests to the request file

The management functions for SF and SM environments are the same. There are additional functions for SM environments for restart and recovery of requests. The management functions are described below.

- Requests for SF environments are managed exclusively in the global HSMS request file by the host on which the home pubset is running.

  For requests relating to SF pubsets only, the request management is purely local for the host. For requests issued to a shared SF pubset, the request management is more complex: requests issued to a slave sharer are usually sent to the master sharer for processing. For more detailed information, see section "Request management for shared pubsets".
- Requests for SM pubsets are managed in the SM pubset-specific request file. As a closed container, an SM pubset can be transported from host to host and shared between several hosts. This is why the request file of an SM pubset has its own request inhibit mechanism which guarantees request coherence while the host or the configuration is being changed.

  The two rules below always apply to SM pubset requests:

  1. A request that is already being processed, is linked by the inhibit mechanism to the host that is processing the request. A request of this type cannot be changed, started or deleted by another host.
  2. A request that has not yet been processed is not linked to any host. A request of this type can be accessed by any host or sharer that has imported the pubset.

  The effects that shared pubsets have on request management are described in section "Request management for shared pubsets".
  Problems that arise from a host or configuration change are described in section "Recovery of requests after a host crash".
  The compatibility of HSMS definitions in a heterogeneous version environment is described on "Compatibility of HSMS definitions in different HSMS environments".

4.7.4.1 Information about requests

While being processed by HSMS, requests pass through various request states:

State	Substate	Meaning
INCOMPLETE		INCOMPLETE (request state) The request has not been accepted
ACCEPTED		The request has been accepted by HSMS and written in the request file. Request is waiting to be started.
STARTED		The request is being processed:
	COLLECTED	– within a collector request
	START-ARCHIVE	– passed on to ARCHIVE
	ARCHIVE-COMPLETED	– completed by ARCHIVE
	START-REPORT	– report is being prepared
	SENT-TO-MASTER	– sent to the master (shared pubset)
	SENT-TO-BACK-SERV	– sent to the backup server (shared pubset)
	BACK-SERV-REPLIED	– response from the backup server
	BACK-SERV-NO-CONN	– no connection to the backup server
	BACK-SERV-TIMEOUT	– timeout for response (shared pubset, backup server)
	MASTER-REPLIED	– response from the master (shared pubset)
	MASTER-NO-CONNECT	– no connection to the master (shared pubset)
	MASTER-TIMEOUT	– timeout for response (shared pubset)
	IN-TRANSMIT	– report completed but not yet transmitted to the active node
	IN-DELETE	– pseudo-substatus: deletion of the started request is currently being initiated.
INTERRUPTED		The request was interrupted during processing (possibly followed by a restart)
	MASTER-NO-CONNECT	– no connection to the master (shared pubset)
	MASTER-TIMEOUT	– timeout for response (shared pubset)
	MASTER-REPLIED	– response from the master (shared pubset)
	BACK-SERV-REPLIED	– response from the backup server (shared pubset)

	BACK-SERV-NO-CONN	– no connection to the backup server (shared pubset)
	BACK-SERV-TIMEOUT	– timeout for response (shared pubset)
	START-ARCHIVE	– Request transferred to ARCHIVE
COMPLETED		The request has been processed by HSMS; the following suffixes appear depending on how the run went:
	WITH-WARNINGS	– request terminated with warnings
	WITH-ERRORS	– request terminated with errors
CANCELLED		The request was deleted before it could be started. No report available. This happens with requests with Concurrent Copy operations (which do not use any SHC-OSD mirroring functions) when an EXPORT-PUBSET is issued while the request has ACCEPTED status.

Table 5: request state

The HSMS statement SHOW-REQUESTS supplies information about the requests in the request file and their states. By default the statement provides information on all requests in the SF environment and all existing SM environments. In SM environments, additional information is provided about the request inhibit mechanism. The “HOST/TSN” field displays the host that has just inhibited the request. Only active requests (status ACCEPTED or STARTED) are locked by their processing host.

It is possible to obtain information about individual requests selected via

- an HSMS environment (request file of the SF or of an SM environment)
- their names as defined at request creation
- their states
- their creation dates
- their user IDs (restricted to the HSMS administrator)
- the archive for which the requests were created (restricted to archive owner and HSMS administrator)

4.7.4.2 Delete requests

Requests with the following statuses can be deleted from the request file using the HSMS statement DELETE-REQUESTS:

- COMPLETED (requests which have already been processed completely).
- INTERRUPTED (requests that have been started but also interrupted).
- ACCEPTED (requests waiting for reactivation (“scheduling”)).
- CANCELLED (requests that have been accepted but aborted before the start).
- STARTED (requests that have been started and are currently running).

Requests which are to be deleted can be selected by means of attributes, as in the SHOW-REQUESTS statement. *ANY selects all requests which have the COMPLETED, INTERRUPTED, ACCEPTED or CANCELLED status. They can also be deleted via the mask output of the SHOW-REQUEST statement. The requests which are to be deleted are selected with a special delete flag.

For information on deleting requests to shared pubsets, see section "Request management for shared pubsets".

Requests with the status STARTED are not deleted immediately. Before a started request is deleted, a deletion message is sent to the HSMS server task that is processing the request. The HSMS server task checks the deletion message and aborts once it has been read. However, since the deletion message does not have any priority, it cannot be read if the HSMS server task, for example, is locked in a security queue. As a result, the started request is not deleted immediately but only after a few minutes.

Requests with status COMPLETED which are older than the value set as KEEP-REQUESTS are automatically deleted by an implicit recovery at the start of each HSMS session. For more details on this, see section "Recovery of requests after a host crash".

4.7.4.3 Restarting requests

Requests that were interrupted during processing (INTERRUPTED state) can be continued using the HSMS statement RESTART-REQUESTS. They do not have to be restarted from the very beginning. A distinction has to be made between the following two cases:

- The request was interrupted before being passed to ARCHIVE for processing. In this case, the request can be continued.
- The request was interrupted while being processed by ARCHIVE. In this case, the request cannot be continued by means of RESTART-REQUESTS unless checkpoints for this request have been written to the ARCHIVE checkpoint file.

  With the HSMS statements for BS2000 files (BACKUP-FILES, ARCHIVE-FILES,...) as well as with the HSMS statement BACKUP-NODE-FILES, the writing of checkpoints is initiated by specifying WRITE-CHECKPOINTS=*YES for the OPERATION-CONTROL operand or WRITE-CHECKPOINTS=*STD if *YES is the default value for the archive concerned.

The requests cannot be continued unless the current operating mode is the same as at the time they were accepted by HSMS (either SIMULATION or OPERATION mode).

In certain coincidental conditions, HSMS generates more than one ARCHIVE run under the one request.
If the request is interrupted and restarted using RESTART-REQUESTS, only the ARCHIVE run started last is restarted, then the request is terminated. The report of the request will contain an error message to this effect.

When requests for node files are restarted, the appropriate UNIX file systems must be mounted, as otherwise, considerable wait times may occur before the UNIX file systems are active again.

Backup requests with Concurrent Copy which use SHC-OSD mirroring functions can be placed in the INTERRUPTED status if problems occur during processing. In this case they can be updated.
All other backup requests with Concurrent Copy are placed in the COMPLETED-WITH-ERRORS status when errors occur and not in the INTERRUPTED status. They can therefore not be updated.

In SM environments, requests with INTERRUPTED status are not linked to the processing host. The restart can therefore be issued from any of the SM pubset sharers irrespective of which host the request was originally generated on. However a request can only be restarted with the same HSMS version with which it was being processed before the interruption.

Normally a restart can only be issued for requests that were interrupted because of a device fault or volume error.

A restart for requests that were interrupted at any given place in execution because of a system crash cannot generally be performed successfully.

In an inhomogeneous shared SM-environment, as soon as at least one of the system has HSMS V12.0A installed, it is recommended to restart requests within the same configuration as it was started originally: processing system (master or backup server) during restart should have the same version of HSMS as when request was started initially.

Note that restarting requests concerning extended S1 level is only possible within HSMS V12.0A. Moreover, such requests interrupted in lower HSMS versions even cannot be started with HSMS V12.0A.

4.7.5 Request management for shared pubsets

The management of requests which relate to shared pubsets is fairly complex. Requests issued to a slave sharer are usually sent to the master sharer for processing. For detailed information on “master” and “local” processing types, see section "Working with shared pubsets".

A request sent to the master sharer is copied by the master. A request issued by a user therefore is dealt with by two separate requests which share the same request identification. The original request generated on the slave host is called the primary copy of the request. The duplicate copy created by the master host is called the master copy of the request.

From the user’s point of view, request management functions can only be issued for the primary copy of a request, i. e. for the request that the user actually issued. Any existing master copy of a request is automatically managed by the request management functions.

A number of differences between SF and SM pubsets also affect the request management for shared pubsets:

- In SF environments, the slave writes the primary copy of the request to its own HSMS global request file. The master writes the master copy of the request to its own HSMS global request file.
- In SM environments, both the primary and the master copy of a request are written to the same request file of the SM pubset.

The special features of the request management for shared pubsets for the SHOW, DELETE and RESTART functions are described below.

Displaying requests on shared pubsets

With the SHOW function, master copies of requests can be recognized from the value YES in the output field FROM-REMOTE.

In SF environments, only the local copy can be displayed immediately, depending on whether the display is on the slave or the master host.

In SM environments, the primary and the master copy of the same request can be displayed at the same time if both of them meet the selection criteria for the request status. The master copy of the request is displayed immediately after the primary copy. The “HOST/TSN” field shows which request is running on which host.

Deleting requests from shared pubsets

Under the DELETE function, only the primary copies of requests can be selected. The associated master copies are implicitly deleted by HSMS. Deletion of the primary copy of a request can only be initiated after the associated master copy has been deleted.

A request cannot be deleted while its master copy has the STARTED status.

Requests which were defined in an SF environment can only be deleted from the host on which they were issued. The master host is called to enable implicit deletion of the master copy.

Requests which were defined in an SM environment can basically be deleted from every host which has imported the SM pubset. However, for as long as a request is locked by a host it can only be deleted by that host.

Restarting requests on shared pubsets

The RESTART function can only be issued for the primary copy of requests which have the status INTERRUPTED. However, the associated master copy is requested for this purpose because only the master copy contains the restart points that HSMS requires for a restart. If the master copy is not available or does not have the status INTERRUPTED, the restart will fail.

In SF environments, the request must be sent to the host that has stored the master copy of the request. The restart therefore always sends the request to the host that was the master sharer when the request was first processed, even if it is no longer the master sharer. If the host is no longer available, the restart will fail.

In SM environments, the restart can be issued from any host or sharer that has imported the SM pubset. If the system detects a master copy request, the primary copy of the request is modified using the checkpoint information of the master copy. The master copy is then removed from the request file. When the request has been processed, the master/slave configuration of the SM pubset is cancelled.
If the local host has slave access to the SM pubset, the request is sent for processing to the current master.

4.7.6 Recovery of requests after a host crash

HSMS supports both the implicit and explicit recovery of requests.

Implicit recovery

During an implicit recovery, the system attempts to rescue the request file. The requests undergo the following operations:

Implicit recovery of requests is carried out at the beginning of an HSMS session.

In an SF environment, implicit recovery occurs each time the subsystem is generated. All of the requests located in the global HSMS request file are processed.

Requests with status COMPLETED are deleted based on the value of KEEP-REQUESTS. The value of KEEP-REQUESTS can be changed with the command MODIFY-HSMS-PARAMETERS. By default the value is *YES (40 days).

In an SM environment, implicit recovery occurs each time the SM pubset is started up, i.e. during import of the pubsets or generation of the subsystems if it was possible to access the SM pubset. For implicit recovery, the inhibit mechanism is taken into account: a thirdparty host cannot update any request that is inhibited by another host. This means that the implicit recovery of an SM pubset is just a “host-local” recovery.

Explicit recovery

Within the framework of the HIPLEX concept (see section "HIPLEX and HSMS"), HSMS supports an explicit mechanism for the extended recovery of requests. This mechanism is, however, only available for environments with SM pubsets. HSMS can use the explicit recovery to respond to host crashes and host reconfigurations within SM pubsets.

An explicit recovery accesses requests that are inhibited by another host. This remote host is usually no longer in the SM pubset configuration. The cause may be a normal export or a system crash. It is possible that the request file still contains requests that are inhibited by the remote host.

Explicit recovery is only available to the HSMS administrator on the master sharer of the SM pubset. The HSMS administrator initiates it based on an SM pubset and a host name if he feels this is necessary. A standard reason may be that the remote host cannot recover the SM pubset configuration within a reasonable time.

For an explicit recovery, the system checks whether the host to be recovered is an active sharer of the SM pubset. If it is, the recovery is rejected. For an exclusive SM pubset, this is achieved by excluding the host itself from the recovery process. In the case of shared SM pubsets, this is achieved by calling the MSCF monitoring task which continuously monitors all the active sharers of all shared SM pubsets. If the connection to the monitoring task fails, the recovery is again rejected.

Monitoring can be suppressed with the FORCE operand of the RECOVER-REQUESTS statement. This can be useful when the host to be recovered is still active, but shows no HSMS activities for the specified SM pubset. Generally speaking, we do not recommend forcing the recovery of a sharer because it

- eliminates processing inhibits
- may start certain requests twice
- may result in inconsistencies in the archive directory.

The recovery normalizes every request inhibited by a host. The inhibit is cancelled and the request is changed depending on its status, in the same way as during implicit recovery:

- Requests with status ACCEPTED are reactivated on the local host and a new inhibit is set.
- Requests with Concurrent Copy and which have the status ACCEPTED and do not use SHC-OSD mirroring functions are assigned status CANCELLED.
- Requests with status STARTED are assigned status INTERRUPTED.
- Requests with status STARTED which could not be restarted are assigned status COMPLETED.

After recovery, the unlocked requests can be accessed by any sharer. Requests with status INTERRUPTED can be restarted by any sharer. Requests with status INTERRUPTED, COMPLETED or CANCELLED can be deleted by any sharer.

If two hosts crash – e.g. master and a slave – an explicit recovery on the new master host can be requested for each host. The recovery sequence is irrelevant to the result.

4.7.7 Assigning priorities for request processing

For each HSMS job, the HSMS administrator can specify the priority with which it is to be processed. For example, important jobs such as system backups can be assigned a high priority and be started even if there are a large number of jobs from nonprivileged users in the system.

To do this, the HSMS administrator can use the REQUEST-PRIORITIES operand in the HSMS statement MODIFY-HSMS-PARAMETERS to define a range of default priorities in the HSMS control parameters. A default priority can be assigned to each archive type:

- Backup archive: default priority for reading and writing
- Long-term archive: default priority for reading and writing
- Migration archive: default priority for reading and writing
- Archive for node backups: default priority for reading and writing
- Archive for the long-term archival of nodes: default priority for reading and writing
- Shadow archive: default priority for reading and writing
- Default priority for import and export
- Default priority for implicit recall

The assigned default priorities can be viewed using the HSMS statement SHOW-HSMS-PARAMETERS (in the *REQUEST-PRIORITIES* information block).

The HSMS administrator can then define the read and write priorities for each individual archive. It can either define a specific priority or use the default priority for the specified archive type. This is done by using the REQUEST-PRIORITIES operand in the HSMS statements CREATE-ARCHIVE and MODIFY-ARCHIVE-ATTRIBUTES.

By default, archives belonging to nonprivileged users contain a reference to the default priorities.

Priorities for shadow archives are taken into account only for HSMS statements which are exclusively executed in the shadow archive – for example a RESTORE-FILES statement for a shadow archive.

When a request for an archive is issued, the processing priority is determined on the basis of the individual read and write priorities for this archive. If the default priority has been defined for the archive in question (REQUEST-PRIORITIES=*STD) then the default priority for this archive type is used. A distinction is made between read and write requests.

Exceptions

There are the following two exceptions to this general procedure:

- The serial processing of requests is taken into account (see section "Request-time control"). This may modify the priority of a request.
  In the case of serial processing, the requests are worked through as “request sets”. In consequence, each request has the priority of the set. New requests have the same priority as the preceding requests in the set.
- Requests which the HSMS administrator issues as express requests are always processed with the highest priority. However, if express requests are processed serially, the priority may be reduced as described above.

4.7.8 Collector requests

If there are several requests referring to the same archive, HSMS normally processes them individually and independently of one other. Under certain conditions, however, HSMS combines such requests in a collector request before processing. This reduces the number of tape access and positioning operations during processing.

Requests are combined in such a collector request if the following conditions apply:

- They cannot be processed immediately.
- They are processed on the processor on which they were issued.
- They are requests for archival or migration to storage level S2.
- They were not created by the HSMS administrator.
- They refer to a system archive (owner SYSHSMS).
- The standard save file of this archive is being used and updated.

Separate reports are generated and output for each of the requests collected.
Information about each of the requests can still be obtained via the normal request name. Its substate will be COLLECTED as long as it is being processed (STARTED state).

If a collector request is interrupted, its restart must be initiated by the HSMS administrator.

If during the migration a particular request is to be monitored with SM2, all requests of the collector request will be monitored.

4.7.9 Asynchronous and synchronous processing

When issuing an action statement, users can specify whether they wish to wait until the statement has been processed by HSMS or whether they would prefer to enter the next HSMS statement immediately.

4.7.9.1 Asynchronous processing

Unless explicitly requested otherwise by the user, HSMS merely checks action statements for syntactical errors and for the validity of certain general conditions. Following this check, the user is immediately free to make the next entry.

In order to execute the required actions, a request is generated and entered in the request file; actual processing then takes place as a background task.
Users can inquire about the progress of their requests using the HSMS statement SHOW-REQUESTS (see section “Information about requests”).

4.7.9.2 Synchronous processing

The user can choose to wait after issuing an HSMS statement until HSMS has completed the required actions. In this case WAIT-FOR-COMPLETION=*YES must be specified for the OPERATION-CONTROL=*PARAMETERS(...) operand.

If HSMS establishes that tape access is required for processing an HSMS statement and tape processing is currently prohibited, the statement is rejected for dialog requests. In batch mode, the statement is passed to a server task at the beginning of a tape processing session.

The HSMS administrator can define maximum wait times for accepting and starting request processing (MODIFY-HSMS-PARAMETERS). Any request not accepted within the defined wait time will be rejected.
If a request is not processed completely by a server task within the defined wait time, HSMS will automatically continue processing of the request asynchronously.

The HSMS administrator can define separate maximum wait times for interactive and batch tasks respectively.

Processing pattern (simplified):

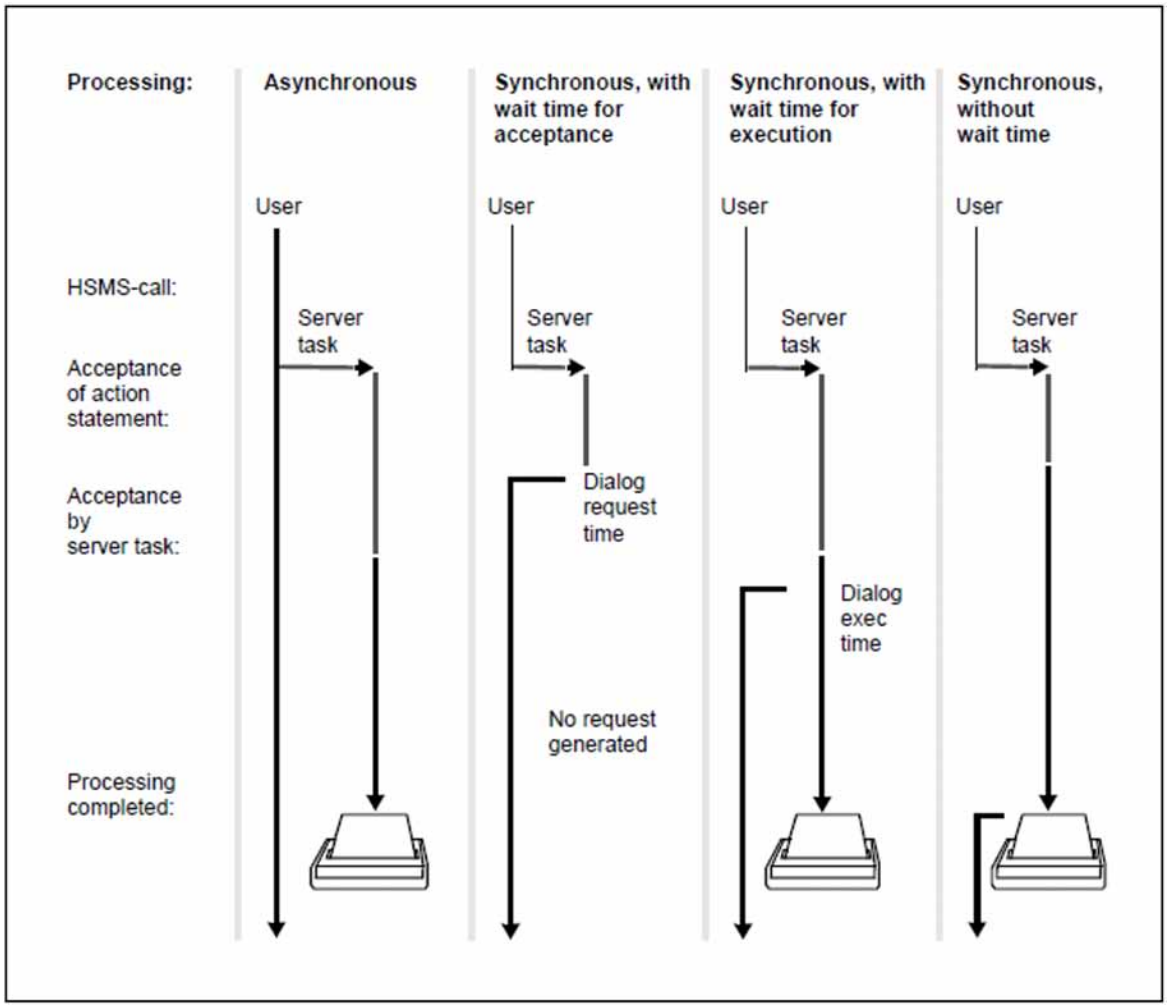


Figure 19: Asynchronous and synchronous processing

4.7.10 Parallel and serial processing in ARCHIVE

The following section describes how you can achieve better performance when working on storage level S2.

Each SAVE, RESTORE and COPY request is processed by a single HSMS server task. This HSMS server task divides the request into so-called packets. Each packet can be processed individually by an ARCHIVE subtask. ARCHIVE subtasks can run in parallel. At the beginning, each ARCHIVE subtask receives a packet. When a subtask has processed a packet it receives a new one.

To ensure good performance, the number of ARCHIVE subtasks generated should be the same as or less than the number of packets created. If multiplexing operation has not been activated, each ARCHIVE subtask requires one tape device for SAVE/RESTORE activities and two tape devices for COPY activities. The number of ARCHIVE subtasks that can successfully run in parallel is limited by the number of available tape devices.

If multiplexing operation is active, multiple ARCHIVE subtasks can share the same devices for SAVE/RESTORE activities in order to achieve better performance and device utilization. For COPY activities, twice as many devices are required for the backup.

In HSMS the division into packets is performed implicitly. The number of ARCHIVE subtasks required for RESTORE and COPY activities is calculated automatically. The user can control the number of packets created for a BACKUP-FILES request as well as the number of ARCHIVE subtasks generated for BACKUP-FILES or BACKUP-NODE-FILES (see "User control of packet generation").

If preprocessing/postprocessing is activated (PRE-POST-PROCESSING operand) in the case of operations involving the BACKUP-NODE-FILES and RESTORE-NODE-FILES statements, the ARCHIVE subtasks await the results of the preprocessing action before they commence data exchange with the workstation.

Multiplexing operation

In the multiplexing operating mode, multiple ARCHIVE subtasks can simultaneously sharethe same MTC devices and magnetic tape cartridges in parallel. This improves performance during save and restore operations and ensures optimal utilization of tape devices and streaming. In addition, the magnetic tape cartridges are optimally filled, thus reducing storage costs.

In multiplex operation the number of tape devices (or tape device pairs when copying) used in parallel for a request is preset, as is the mulitplex factor which determines the number of ARCHIVE subtasks which share a tape device.

In BS2000 save operation multiplex operation is well suited for configurations with high-performance cartridge devices (as of TAPE-C5/-C6).

In HSMS, multiplexing operation is achieved by connecting multiple ARCHIVE subtasks to a single ARCHIVE tape drive. The ARCHIVE tape drive is responsible for all accesses to the connected device. The number of subtasks for a device driver task determines its multiplexing factor. As a result, the subtasks refer to the device driver task when writing to or reading from tape.

At restore time, automatic parallel demultiplexing is performed. The number of subtasks required depends on the number of devices, the multiplexing configuration during backup and the files that are to be restored (see also section "Multiplexing operation").

Multiplexing operation is activated by specifying PARALLEL-RUNS=*MULTIPLEXING(...) in the HSMS statement BACKUP-NODE-FILES, the suboperands NUMBER-OF-DEVICES and MULTIPLEXING-FACTOR determining the number of devices and the multiplex factor. The settings for multiplex operation can also be specified at archive level using the HSMS statements CREATE-ARCHIVE and MODIFY-ARCHIVE-ATTRIBUTES. During the backup these are evaluated by specifying the PARALLEL-RUNS=*STD option.

If data is backed up to disk, no multiplexing is performed. However, the number of ARCHIVE subtasks is exactly the same as the planned number of devices (NUMBER-OF-DEVICES).

In the case of MULTIPLEXING-FACTOR=*AUTOMATIC (default setting) ARCHIVE determines a favorable multiplexing factor by itself. The result of this operation gives the multiplexing factor per drive. For further information, refer to section "Multiplexing operation".

Specifying node files via PATH-NAMES in multiplexing operation

If the path names of the node files to be processed are taken over from a file or a library element (PATH-NAMES=*FROM-FILE or *FROM-LIBRARY-ELEMENT operand), they are processed in the sequence in which the user has entered them in the specified file or library element. Consequently, BACKUP-NODE-FILES operations can be optimized by planning the sequence of files carefully. See also section "Selection of node files of the BS2000-UFS".

New format for multiplexed save files

Save files created in multiplexing mode have a slightly different format from save files which were not created in multiplexing mode. Among other things, additional control information is added to the tape security blocks.

Continuing save files in multiplexing operation

A non-multiplexed save file cannot be continued with a multiplexed save version since the tape formats are different. The entire save file must have a uniform format if restore operations are to be performed correctly.

However, it is possible to continue a multiplexed save file with a non-multiplexed save version. ARCHIVE automatically creates a multiplexing environment. In this case, the employed multiplexing factor is 1.

Selecting files via *LATEST-BACKUPS-OR-S0 in multiplexing operation

If the files for backup are selected by means of the specification FROM=*LATEST-BACKUPS-OR-S0, different save files can be used as input during the offline copying phase. This means that both multiplexed and non-multiplexed save files can be copied. When a multiplexed save version is copied, the entire new output save file must be created as a multiplexed file. As a result, non-multiplexed input save versions are multiplexed in the new output save file.

4.7.10.1 Automatic packet generation

In backup runs, packet generation depends on the type of the file being processed:

BS2000 files

In backup runs, division into packets takes place in accordance with the catalog ID and the user ID.

In the case of PARALLEL-RUNS=1, there is one packet per catalog ID and user ID. However, as soon as PARALLEL-RUNS>1 is specified, HSMS divides the files of a catalog ID and user ID into four packets according to their location on the pubset disks.

When the number of pubset disks is greater than three, all four packets contain files. When the number of pubset disks is less than three, the number of filled packets is the same as the number of pubset disks. This means that the other packets of the four remain empty.

Packet division thus also permits parallel processing when files are saved from just one catalog ID and user ID. Division into smaller packets also results in more even utilization of the parallel subtasks, and therefore reduced backup times.

Division as described in section “User control of packet generation” is only still to be recommended in exceptional cases, e.g. when saving very few but very large files.

Files on Net-Storage

For the files on Net-Storage, HSMS generates one packet per catalog ID and user ID for each Net-Storage concerned irrespective of the specification in the PARALLEL-RUNS operand.

Node files of passive nodes S0

In save runs, node files from various passive nodes are assigned to separate packets. First the specified number of ARCHIVE subtasks is started and subsequently packets created. Initially the lowest common directory level is determined from the specified node files in a node and then mutually independent subtrees are determined at this level. If there are less free subtasks than subtrees, a number of subtrees are combined. In each subtree a packet is created for each directory level; all packets are allocated to a subtreee, and with this all files from a file tree processed by the same subtask.

This structuring applies both for PARALLEL-RUNS and MULTIPLEXING.

If the file specification *ALL is being used in a save run for node files in BS2000 so that the include/exclude files are evaluated on the associated client, internal structuring into subtrees takes place beforehand as if the Include /Exclude specifications had been made in BS2000.

With RESTORE and COPY runs the structuring into packets depends on how the data was saved to the volumes.

4.7.10.2 Automatic generation of ARCHIVE subtasks

During *backup runs* the number of required ARCHIVE subtasks is automatically calculated as follows:

- Without multiplex operation as the number of devices operated in parallel, specified by PARALLEL-RUNS with the default value 1.
- With multiplex operation the multiplex factor defines the number of ARCHIVE subtasks which share the same devices. The number of devices multiplied by the multiplex factor specified numerically thus provides the number of subtasks working in parallel.
- When MULTIPLEXING-FACTOR=*AUTOMATIC is specified, the actual multiplex factor and thus the number of subtasks is calculated internally in accordance with the number of subtrees from the specification of the files to be saved.

With *COPY* runs the number of subtasks required is derived from the number of devices operated in parallel regardless of multiplexing.

With *RESTORE* runs the multiplexing factor used for the backup is also taken into account.

4.7.10.3 User control of packet generation

When backing up BS2000 files, the user can initiate a further subdivision of the automatically generated packets using the SELECT=*FROM-FILE(...) option.

> Automatic packet division by HSMS (see section "Automatic packet generation") also permits parallel operation in cases where in earlier HSMS versions the user – as described below – had to control packet generation. In addition, automatic packet division by HSMS is more favorable for parallel operation than user-controlled packet generation because the location of the files on autonomous disks which is important for parallel operation is taken into account internally.

Whenever a user has many large files in a system, it can be advantageous to use several parallel subtasks to save these files. How this is done is described below.

When saving, it is specified in the input or except file which user files are to be saved or excluded from the save and in which sequence they should be processed.

Files with different user IDs or different catalog IDs are distributed to different packets. The packets are distributed to parallel subtasks and processed in parallel.

Up to HSMS V7.0 files with the same user and catalog ID were assigned to one packet and therefore processed in one subtask. Only with the special syntax described below was it possible to divide such packets for parallel processing when BACKUP-FILES is used.

Currently HSMS divides these files into a maximum of four packets in accordance with their disk location, thus permitting parallel processing. The internal and automatic fine division is more convenient for the user and, because it is disk-oriented, more favorable for parallel processing and even subtask utilization than the previous method with special syntax.

The workaround below makes sense in exceptional cases, e.g. when few but very large files are saved, in order to achieve an even distribution of the load on the drives:

Every record in an input or except file must contain a path name. It may also optionally contain a group ID. The group ID is used to distribute files to separate packet for parallel processing by different subtasks. Specifying the group ID does not, however, allow files from different user or catalog IDs to be forced into the same packet!

```
[<gruppen-id>;] <pfadname>
```

The path name can be specified as a fully or partially qualified name with or without a catalog ID and a user ID. The use of wildcards is allowed. If a group ID is specified, it must be at least 1 but no more than 8 alphanumeric characters in length. If no group ID is specified, a blank is assumed.

If group IDs are used together with file excepts (EXCEPT-FILE-NAMES operand), the group IDs specified in the except file must be compatible with the group IDs specified in the selection file.

Examples

The following examples illustrate the conversion of an HSMS file specification to an ARCHIVE specification (NAME operand).

Example 1

Input file:	:a:file1 :a:file2 :a:file3 :b:file1	
ARCHIVE entries:	NAME 1 :a:file1 :a:file2 :a:file3	NAME 2 :b:file1

Example 2

Input file:	pkt1;:a:file1 pkt2;:a:file2 pkt1;:a:file3 pkt1;:b:file1		
ARCHIVE entries:	NAME 1 :a:file1 :a:file3	NAME 2 :a:file2	NAME 3 :b:file1

Example 3

Input file:	:a:file1 2;:a:file2 :a:file3 1;:b:file1		
ARCHIVE entries:	NAME 1 :a:file1 :a:file3	NAME 2 :a:file2	NAME 3 :b:file1

Example 4

Input file:	PKT1;:a*:$u*.file1 PKT2;:a*:$u*.file2	Pubset on the system: a1, a2
ARCHIVE entries:	NAME 1 :a1:$u*.file1	NAME 2 :a1:$u*.file2

	NAME 3 :a2:$u*.file1	NAME 4 :a2:$u*.file2

Example 5

Input file:	pkt1;:a:file1* pkt2;:a:file2* pkt1;:b:file1*
Except file:	pkt1;:a:file13 pkt2;:a:file22 pkt1;:b:file11
ARCHIVE entries:	NAME1 :a:file1* except :a:file13
	NAME2 :a:file2* except :a:file22
	NAME 3 :b:file1* except :b:file11

4.7.10.4 User control of the generated subtasks

The user can control the number of subtasks generated by means of the PARALLEL-RUNS operand. Interpretation of the value specified for PARALLEL-RUNS is subject to the following rules:

- The number of subtasks that can be used to continue, read or copy an existing backup file must not exceed the number of subtasks that were actually used when the save file was created. This also applies when an existing save file is extended without multiplexing operation.

  If multiplexing operation is active, an existing save file with a higher multiplexing factor than the predecessor save versions can be continued. Thus the number of ARCHIVE subtasks can be higher but not the number of devices.
- If the number of subtasks used for COPY activities on a save file is less than the number of subtasks that were actually used when the save file was created it can happen (with SEVERAL-SVID) that the volume is rewound to the beginning-of-tape marker once more during processing.

Notes

- If the number of packets generated is less than the number of ARCHIVE subtasks requested, some ARCHIVE subtasks will be generated which do not get used.
- The number of ARCHIVE subtasks used to create save files should be calculated such that the same number of subtasks can be used for COPY activities from these save files, although the number of tape devices available at the time of copying must be taken into account.

4.8 Volume processing

This section describes the following:

- the volumes supported on storage level S2
- the allocation of volumes
- volume management via Maren
- volume and device reservation
- the storage locations of volumes and device pools
- tape sessions
- tape processing control

4.8.1 Level S2 volumes

At level S2, HSMS and ARCHIVE support all volumes of the TAPE class that are also supported by BS2000 versions on which the current HSMS version can run.

Action statements resulting in write requests to S2 need not specify the type of volume desired. The exception being volumes that do not belong to the archive (see section "Allocating volumes").
The type of volume requested is determined by global control parameters or by the definition of the archive which is to be written to.

It is the HSMS administrator’s task to define the default volume type which is to be supplied globally for all write requests to S2 unless the archive owner has explicitly defined a different device type:

```
//MODIFY-HSMS-PARAMETERS -
// DEFAULT-HSMS-STORAGE=*PARAMETERS(S2-DEVICE-TYPE=<c-string>)
```

A different default volume type for saving data to a specific archive at storage level S2 can be defined by

```
//CREATE-ARCHIVE S2-DEVICE-TYPE=<c-string>
```

or

```
//MODIFY-ARCHIVE-ATTRIBUTES S2-DEVICE-TYPE=<c-string>
```

The definition of a default volume type can be used to introduce a new volume type for the entire system or for a specific archive without necessitating modifications to existing procedures.

4.8.2 Allocating volumes

Write requests to S2 require save volumes. Write requests always refer to specific archives. Unless otherwise specified the required volumes are fetched from the free volume pool of the archive involved (or the associated MAREN pool). See section "Volume pool management" for a description of how to create and manage such a volume pool.

Both with backup requests and when copying save files, the HSMS administrator and/or the owner of the archive involved have various options for controlling allocation of the volumes.

4.8.2.1 Volume allocation by the user or the operator

By specifying a list of volume serial numbers (VOLUMES=<vsn>) the user can define the volumes.

On the console, the operator receives a request to assign volumes in the following cases:

- VOLUMES=*FROM-OPERATOR was specified.
- The archive’s volume pool is exhausted and no further volumes can be taken from it.

Both the volumes allocated by the operator and those explicitly requested by the user but not present in the pool are listed in the directory. They are displayed with //SHOW-ARCHIVE SELECT=*VOLUMES with the specification OWNER= OPERATOR.

Once the save file stored on these volumes has been deleted, the entries for these volumes are deleted from the directory (even with FORCED-DELETE). The volumes are not included in the volume pool.

When MAREN is used these volumes must already be allocated to the user ID (STATUS=RESERVED). They are then administered by MAREN.

4.8.2.2 Volume allocation from the volume pool

When VOLUMES=*FROM-POOL is specified, the volumes are taken from the volume pool of the archive which has been created in the directory. If no volume pool has been created or this pool is exhausted, the tapes are allocated from an assigned MAREN pool using MAREN.

If both pools are exhausted, the volumes are requested from the operator as with VOLUMES=*FROM-OPERATOR.

When volumes are released these remain allocated to the ARCHIVE and are available for repeated use.

Management of the volume pool is described in section "Volume pool management".

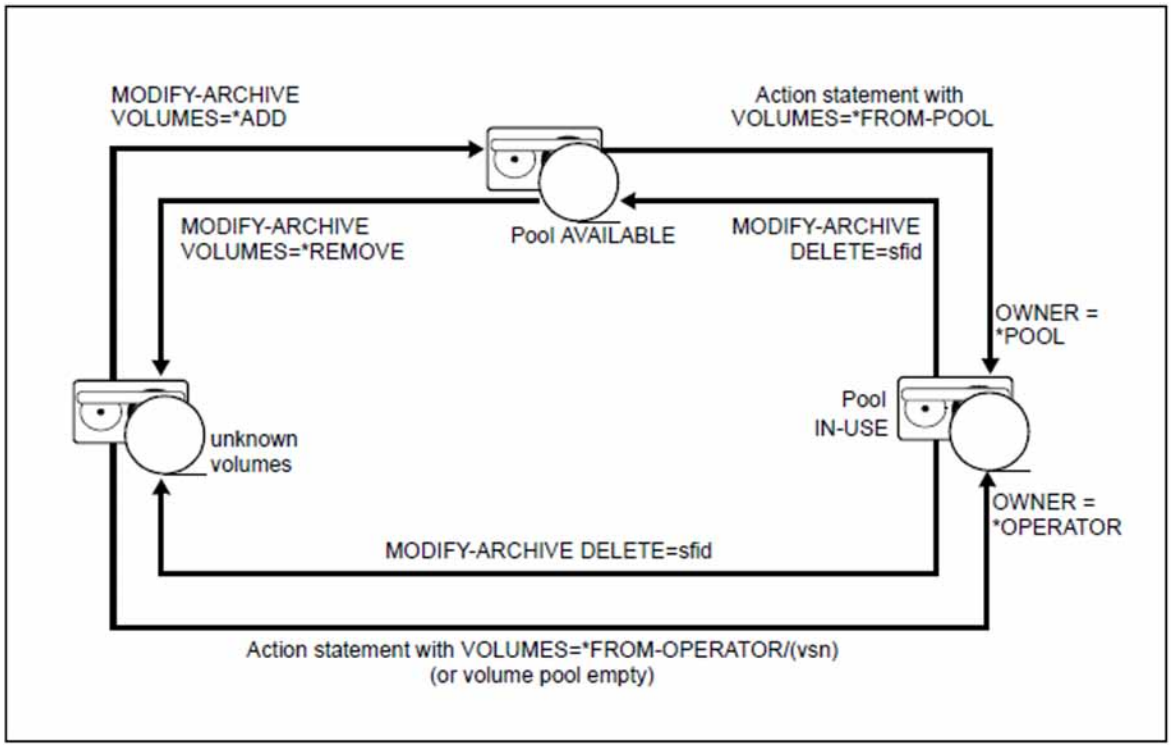


Figure 20: Volume allocation in HSMS

4.8.2.3 Volume assignment using MAREN

MAREN is a volume archiving system for BS2000 computer centers. MAREN stores all volume-related information in a specific MAREN catalog; there may be one central MAREN catalog for a number of installations. Computer center operation can be optimized by having MAREN interoperate with other BS2000 products such as ROBAR.

The free volumes are managed in free tape pools in MAREN.

Allocation of volumes in MAREN means reserving a volume either explicitly using a MAREN statement or implicitly via the free tape allocation facility. In this case the volumes are, in particular, allocated the user ID and archive directory.

The HSMS archive directory is allocated a MAREN pool in the same way as with the ARCHIVE directory file: the MAREN pool is thus not assigned to the archive name, but to the archive directory. The information on the ARCHIVE directory in the “MAREN” manual [10] thus applies analogously to HSMS archive directories.

Volumes from MAREN free tape pools can be allocated in HSMS in various ways:

- In the event of explicit allocation users themselves reserve a free volume with the MAREN statement //RESERVE-FREE-VOLUMES before calling HSMS. They can specify this volume directly as backup volume when data is backed up (VOLUMES=<vsn>).
- Implicit reservation is implemented by the automatic free tape allocation facility. The automatic free tape allocation facility is used when VOLUMES=*FROM-OPERATOR is specified. When VOLUMES=*FROM-POOL is specified it is used if the archive’s volume pool is exhausted or has not yet been created. In this case the volume is reserved for the archive directory’s user ID.
- If no tape is assigned via the automatic free tape allocation facility, a volume is requested on the console. The operator then allocates a volume that is managed by MAREN.

For detailed information on volume management using MAREN, please refer to the “MAREN” manual [10].

4.8.3 Volume and device reservation

When ARCHIVE processes a request under HSMS, it issues a SECURE-RESOURCE-ALLOCATION command either for the volume to be read or written to, or for the device which is to be used.

A SECURE-RESOURCE-ALLOCATION command is issued with the wait time defined by the HSMS administrator for the acceptance and the start of requests (HSMS parameter DIALOG-REQUEST-TIME or BATCH-REQUEST-TIME).

The minimum wait time is 200 seconds. If a wait time of 99999 seconds is specified, the secure wait time is increased to 7 days.

If device reservation is not executed within the wait time, either the operation aborts (in the case of outputs), or further attempts are made to open the input volume (in the case of inputs).

Note

ISECURE-RESOURCE-ALLOCATION commands for devices (DEVICE operand) should not be issued in the user task, since they stay reserved in the user task and are not made available to HSMS, which impedes the execution of the request.

When save versions are duplicated, a deadlock can occur if insufficient tape units are available. This is also possible during BACKUP, ARCHIVAL or EXPORT of migrated files. The deadlock ends automatically with the expiration of the wait time set by REQUEST-WAIT-LIMITS in the HSMS operands.

The user can avoid this situation by

- setting DIALOG-REQUEST-TIME resp. BATCH-REQUEST-TIME to a suitably short time in order to preclude an excessive deadlock, but long enough to permit execution to be terminated normally;
- ensuring that the number of tape units available is sufficient for all parallel runs.

The following procedure is provided to rectify deadlock situations:
When a deadlock situation is detected, the message ARC0840 resp. ARC0861 is output and removal is started automatically. The task that encountered the deadlock situation is closed and the volumes currently opened for it are released and re-requested. If the deadlock situation is eliminated correctly and the run is resumed as normal, message ARC0862 will be output.
Otherwise message ARC0707 is output and the task is aborted.

Note

Deadlock cannot occur in the case of RESTORE and IMPORT and when jobs are restarted.

4.8.4 Storage locations of data volumes and device pools

The storage locations of volumes are administered in MAREN (see the “MAREN” manual [10]). MAREN logs all the storage locations of the volumes. Whenever a volume is moved to another location, the corresponding entry in the MAREN catalog must be updated. The following input fields are concerned here:

- TEMPORARY-LOCATION: current location of the volume
- HOME-LOCATION: permanent location of the volume

A device pool management concept was developed in Nucleus Device Management (NDM) primarily in order to support robot-controlled archives in BS2000 (software product “ROBAR”). In conjunction with MAREN, this device pool management enables users to select specific devices for processing tape volumes. The ADD-DEVICE-DEPOT command is used to assign tape units to storage locations. However, the locations must be made public in MAREN beforehand by the system administrator and they must tally with those specified by the operator.

The SECURE-RESOURCE-ALLOCATION command can be used to reserve a tape unit for a certain location. If HSMS requires a magnetic tape cartridge for a read or write operation, a /SECURE-RESOURCE-ALLOCATION command is issued internally with the specified storage location. The following restrictions apply:

- The devices are requested with the specified storage location or that defined in the archive. If the volumes are also specified, these must be assigned to the specified storage location.
- If HSMS knows no storage location but only the volumes, the location of the volumes is determined using MAREN.
- /SECURE-RESOURCE-ALLOCATION is issued for every input data volume required for any of the HSMS statements RESTORE-FILES, IMPORT-FILES and RECALL-MIGRATED-FILES. The wait time corresponds to the specification defined with the HSMS parameter DIALOG-REQUEST-TIME or BATCH-REQUEST-TIME.
- The following applies to other activities (BACKUP-FILES, BACKUP-NODE-FILES, ARCHIVE-FILES, ARCHIVE-NODE-FILES, MIGRATE-FILES, EXPORT-FILES, COPY-SAVE-FILE, COPY-NODE-SAVE-FILE, COPY-EXPORT-SAVE-FILE, UPDATE-EXPORT-SAVE-FILE):
  - /SECURE-RESOURCE-ALLOCATION is issued once for the output device having the same storage location as the output data volume. The assumption is that all subsequent data volumes are available at the same location.
  - /SECURE-RESOURCE-ALLOCATION is issued every time duplication of a save version commences. The assumption is that all data volumes belonging to a save version are available at the same storage location (i. e. that the user has not changed the storage location in the MAREN catalog).
  - If no storage location is defined in the system, /SECURE-RESOURCE-ALLOCATION is issued without specification of a storage location. If at least one storage location is specified, HSMS assumes that all the storage locations entered in the input fields of the MAREN catalog correspond to a valid storage location. If this is incorrect, it causes an error and in some cases it can even lead to an abort.
  - If a task has already been assigned an output device and this task requests a new output tape from MAREN, then MAREN can take control of the output device. In this way, MAREN can – if necessary – initialize the newly requested output tape. Following initialization, the output device is returned to the task in question so that this can continue processing.

    ARCHIVE does not support tape initialization with //COPY-SAVE-FILE or //COPY-NODE-SAVE-FILE with SAVE-FILE=*NEW(...). In this case, ARCHIVE attempts to use the tape without initializing it. If, despite this, tape initialization is demanded, MAREN rejects the tape.

4.8.5 Tape sessions

HSMS permits request processing to be controlled so that input from S2 and output to S2 are restricted to specific times. Requests which require volumes of the class “TAPE” to be mounted are processed during a tape session only.
Synchronous processing of action statements which require volumes of the class “TAPE” is likewise permitted during these tape sessions only.

At the beginning of each tape session HSMS checks which of the requests currently waiting in the request file require access to S2 (for a particular archive or for a particular access type, see below). All relevant accesses are processed during the current tape session. Requests that have been received up to this point may be combined in collector requests, if appropriate, in order to accelerate request processing (see section "Collector requests").

Any requests received after the start of a tape session are assigned to the next tape session. The user cannot control the length of the tape session. It depends exclusively on the time required for processing the access requests received.
No tape session is started before the previous one has terminated.

Tape sessions can be defined for the entire system or for individual archives. Tape session control is the exclusive responsibility of the HSMS administrator. This also applies to private archives of other archive owners. The HSMS administrator sets the global system default values by means of the DEFAULT-TAPE-CONTROL operand of the HSMS statement MODIFY-HSMS-PARAMETERS. These values apply to data transfer in general; with regard to the other basic functions, the HSMS administrator can define different values for individual archives by means of the HSMS statement MODIFY-TAPE-CONTROL.

Tape sessions are defined separately for the various types of access and request.

Type of task	Operand	Access type
Read requests	READ-CONTROL	Input
Write requests	WRITE-CONTROL	Output
Express requests	EXPRESS-CONTROL	HSMS administrator I/Os

Read requests are all requests generated by

- IMPORT-FILES
- RECALL-MIGRATED-FILES FROM-STORAGE=*ANY/*S2-STORAGE-LEVEL
- RESTORE-FILES
- RESTORE-NODE-FILES
- implicit recall of migrated files

4.8.5.1 Unrestricted tape access

...-CONTROL=*PROCESS-REQUESTS permits requests of the specified access type that refer to the archive concerned to be made at any time.
In the event of a switch to a different tape control mode, processing of all requests received while *PROCESS-REQUESTS applied is first completed.

4.8.5.2 Delayed tape access

...-CONTROL=*HOLD-REQUESTS forces all requests of the specified access type to wait. It temporarily prohibits any access of the specified type to the archive involved.

All requests affected are logged in the request file until another value is entered changing tape control for this access type.

Note

If VALID-PERIOD=*SESSION was specified for the first MODIFY-TAPE-CONTROL statement, tape control changes the next time HSMS starts, which usually means that the delayed requests are started.

4.8.5.3 Defining the tape sessions

Tape sessions are defined by using ...-CONTROL=*BY-TAPE-SESSIONS(START-TIME=<time>) to specify a time at which the first tape session is to begin each day.
When START-TIME=*IMMEDIATELY, tape accesses are permitted after the HSMS statement has been entered or the HSMS session has been started. When PERIOD=0, they are permitted from the start of the tape session to the end of the day.

*BY-TAPE-SESSIONS(PERIOD=<*minutes*>) specifies how many minutes are to pass between the start of two tape sessions.

4.8.5.4 Controlling write access by quantity (only for archives for BS2000 files)

Write access may be subject to a different type of control. By specifying a value for MINIMUM-PAGES (WRITE-CONTROL operand), the start of a tape session may be made contingent on the quantity of data to be written: tape access is delayed until the total volume of all write requests has reached the specified number of 2-Kbyte blocks (PAM pages). If the minimum number of pages is not reached until the end of the HSMS session, the requests are retained for the next HSMS session.

Only the following are counted for MINIMUM-PAGES:

- write requests from S0 to S2
- write requests from nonprivileged users.

This means that the following are not counted:

- write requests from the HSMS administrator
- requests to copy save files
- requests for migration from S1 to S2.

By specifying START-TIME=*IMMEDIATELY, PERIOD=0 and MINIMUM-PAGES >= 0 it is possible to determine that write access is not time-controlled at all but always delayed until a minimum number of PAM pages can be written.

By specifying MINIMUM-PAGES in conjunction with NEW-STD-SAVE-FILE=*EACH-TAPE-SESSION, the latter defining for the archive concerned that a new standard save file is to be started for each tape session, the approximate degree of utilization of the tape save file created by the write request can be controlled; this is applicable to migration and long-term archives provided they have been defined accordingly.

Since write requests from the HSMS administrator are not counted in the number of pages to be written, i.e. would require additional space on the volume, it is advisable in this case always to start write requests from the HSMS administrator as express requests.

4.8.6 Tape processing control

Tape processing control is predetermined by the archive definition (operands OPERATION-CONTROL and TAPE-CONTROL). However, the corresponding action statement operands can be used to overrule this presetting for individual requests.

4.8.6.1 Size of tape blocks

Large tape blocks have two advantages:

- They increase the degree of tape utilization due to the smaller number of gaps.
- They improve the throughput when saving to tape, thus speeding up tape processing.

The tape block size can be selected via the BLOCKING-FACTOR operand for each statement or as an archive attribute. In addition, the tape block size can be selected globally using the ARCHIVE parameter BLOCK-SIZE-TAPE (for long tapes) and BLOCK-SIZE-T-C (for MTC devices).

*BLOCKING-FACTOR=*STD*

Specifying *STD (the default) selects the blocking factor as follows:

- The blocking factor from the archive definition applies for statements in which the archive is specified. If *STD (default when the archive is created) is also specified there, the settings of the global ARCHIVE parameters apply.
- The settings of the global ARCHIVE parameters apply for statements in which no archive is specified.

By default a block size of 256 Kbyte (BLOCK-SIZE-T-C=BIG) is set in the parameter file SYSPAR.ARCHIVE.<vers> for MTC devices. For devices which do not support this block size (emulated tape devices, T9G), ARCHIVE automatically uses a block size of 32 Kbyte. Save files are updated with the tape block size used to date.

*BLOCKING-FACTOR=*MAX*

The specification *MAX selects the maximum blocking factor possible in the current BS2000 version. Currently the maximum value is 128 (block size 256 Kbyte). A block size of 256 Kbyte also corresponds to the maximum block size for the tape devices currently released Save files which are written with the maximum value of a BS2000 version cannot be read in an earlier BS2000 version which does not support this blocking factor.

4.8.6.2 Unloading tapes

The UNLOAD-TAPE operand controls whether a magnetic tape cartridge is to be rewound to the beginning and unloaded after writing or positioned to the beginning-of-tape (BOT) marker. Tape unloading is also dependent on the system default in BS2000 which the operator can set with the UNLOAD-RELEASED-TAPE parameter in the MODIFY-MOUNT-PARAMETER command.

4.9 Handling BS2000 disks

BS2000 supports both disks with PAM keys (K disks) and disks without PAM keys (NK disks). HSMS can use both types of disks. HSMS reads and writes files with all PAM key formats (BLOCK-CONTROL-INFO=*NO/*WITHIN-DATA-BLOCK/*PAMKEY).

Files can be saved to and restored from NK4 disks. For details of the boundary conditions that apply to restoration from NK4 disks, see the section “NK4 disks” in the “ARCHIVE” manual [2].

By default, files are loaded during read operations in the same format in which they were saved, as long as this is permitted by the format of the volume to which they are written.

Partial backup is not possible for PAM files without PAM keys (BLOCK-CONTROL-INFO= *NO), except in the case of PLAM libraries.

In BS2000/OSD-BC V9.0 and higher, a pubset can be provided with additional storage space which a net server makes available as a Net-Storage volume. From the BS2000 viewpoint a Net-Storage volume is a disk with the volume type NETSTOR.
Files on Net-Storage are saved and restored like files on local pubset disks. When restored, files can be written to Net-Storage. Save files can be created on and also copied to Net-Storage.

4.9.1 Optimizing a storage space request for a save file

In order to optimize the performance when saving to disk, in the catalog entry the values for primary and secondary allocation should be adjusted to the expected volume of data to be saved. When the values are too small, a correspondingly large number of file extents is created, which can have a negative effect on the access times.

The same holds true for backups to the extended S1 level in an SM environment and to S1-SM pubsets in an SF environment. In these cases values that are too small cause the creation of save files with the subsequent number on volume sets already in use. This can cause problems with subsequent restore or recall requests.

When saving to Net-Storage, the primary allocation is automatically assigned a correspondingly high value.

4.9.2 Conversion of BS2000 files

HSMS offers a conversion option for instances in which the output volume does not permit PAM keys to be written and the saved files have a PAM key.

Conversion can be necessary particularly when importing files which originated on other BS2000 systems or when restoring files from long-term archives.

File conversion can be controlled by the FILE-CONVERSION operand in the HSMS statements RESTORE-FILES and IMPORT-FILES. This operand is ignored for job variables.

By default, files are converted by HSMS when restoring and importing. When FILE-CONVERSION=*NO is specified files located in the save file which have a PAM key are not processed when restored or imported to an NK disk. SAM files with PAM key are also not processed if they are to be restored or imported to Net-Storage with the file type node file. HSMS outputs a warning in the report for each file not processed for this reason.

Files are converted using the PAMINT subsystem.

There are two conversion methods, as outlined below.

Default format

When loaded to an NK disk, files with a PAM key are converted via PAMINT according to the following rules:

- K-ISAM files to NK-ISAM files
  (BLOCK-CONTROL-INFO=*WITHIN-DATA-BLOCK)
- K-SAM-files to NK-SAM files
  (BLOCK-CONTROL-INFO=*WITHIN-DATA-BLOCK)
- K-UPAM files to NK-UPAM files
  (BLOCK-CONTROL-INFO=*NO)
  PAM files whose structure is known (load modules, libraries) are converted accordingly. If the structure of PAM files is unknown, the PAM key information is lost. An appropriate message is displayed if the PAM key contained information that was lost. LMR libraries are converted to PLAM format.

CONV format

When loaded to an NK disk, files with a PAM key are converted to “CONV format” via PAMINT. The restored or imported file contains all the PAM keys at the end of the file in separate blocks.

Only *CONV can be specified for partially backed up files. If a file to be restored must be converted, the RELEASED-UNUSED-SPACE operand is ignored.

Converting large files (>= 32 GB)

Conversion of larger files in HSMS works in the usual way, with PAMCONV. Note that large files can only be converted on pubsets with the attributes, LARGE-OBJECTS=*YES and LARGE-FILES-ALLOWED=*YES.

Exception

The space requirements for a file in K format may increase as a result of the conversion to NK, so that the corresponding initial size and structure of the K file exceeds the 32 GB limit. Conversion is only then possible if the target file is located on a pubset with the attributes LARGE-OBJECTS=*YES and LARGE-FILES-ALLOWED=*YES.

4.10 HSMS output

After SHOW statements, HSMS outputs information for the user in screen masks and S variables, and after the processing of requests in reports.

4.10.1 Outputs after SHOW statements

- Screen masks
- S variables

4.10.1.1 Screen masks

HSMS makes use of screen masks to output the information requested by means of the SHOW statements as well as for file selection in interactive mode. All HSMS screen masks have

- a header section containing the current HSMS statement together with the selected options; in the case of the HSMS statement SHOW-HSMS-PARAMETERS it also indicates whether the parameters on display are the permanent parameters stored in the HSMS control file or the temporary, session-specific parameters;
- a data section to which the requested objects are output;
- a footer where actions such as scrolling or terminating output can be entered;
- space for displaying a message on the last line.

With regard to the data section, there are two types of HSMS screen masks: single-item screens and table screens.

- Single-item screens are used to output all attributes of a single object (archive, pubset etc.).
- Table screens display information about a number of objects of the same type, one object per line.

These screen masks serve exclusively to supply information. The objects or attributes displayed cannot be modified.

Mask output can take place as follows:

- Output to the logical system file SYSOUT and thus onto the screen.
- Output to the logical system file SYSLST or to a file which has been assigned to SYSLST. This enables the screen mask to be printed out or processed in a separate file. To save paper screen masks with 43 lines are used for outputs to SYSLST.

All screen masks include a footer (after NEXT-PAGE) in which it is possible to request scrolling up or down to preceding or following screens:

Input	Effect
+	Output of a screen displaying the next objects
++	Output of a screen displaying the last objects
–	Output of a screen displaying the preceding objects
– –	Output of a screen displaying the first objects
E	Terminates output

Note

Only the significant leading characters are taken into account.

A detailed description of the individual screen masks is provided in “HSMS Vol. 2” [1] following in each case the description of the HSMS statement which causes display of the screen mask.
For screen mask table screens, the meaning of the individual columns is in each case explained in the manual in a key following the screen mask.

Note

Only block-oriented data display terminals (e.g. 9750) support FHS masks. FHS masks cannot be displayed on line-oriented data display terminals (e.g. connection via VTSU). Consequently, always divert the output of SHOW-... statements to *SYSLST when working on a line-oriented data display terminal.

4.10.1.2 S variables

With the following statements the content of the screen masks can also be output in S variables:

SHOW-ARCHIVE
SHOW-ARCHIVE-ATTRIBUTES
SHOW-HSMS-PARAMETERS
SHOW-NODE-PARAMETERS
SHOW-PUBSET-PARAMETERS
SHOW-REQUESTS
SHOW-SM-PUBSET-PARAMETERS
SHOW-TAPE-CONTROL

The structure of the S variables corresponds to the line-by-line output of the screen masks. The S variables are better suited for machine analysis than the content of the SYSLST file.

4.10.2 Reports

After an action statement or a MODIFY-ARCHIVE or CHECK-CATALOGED-FILES statement has been processed, HSMS generates reports which supply information about the result of the request.
Report output can be controlled via suboperands of the OPERATION-CONTROL operand of the action statements. A default value can be defined for the OUTPUT suboperand with the global HSMS parameter OUTPUT. This can be *PRINTER or *MAIL.

The REPORT suboperand determines the output scope as follows:

- The operand value *SUMMARY causes a summary to be output which supplies information about the statement, the request generated and the save file created.
- *FULL causes additional information to be output about the individual files, PLAM library elements and job variables processed.
- *SAVED-FILES/*RESTORED-FILES restricts output to those files which have actually been processed in backup or restore runs.
- *NONE causes report output to be suppressed.

Note

When migration takes place from S1 or S2 (reorganization), multiple copy and delete requests are executed internally. Here logging is performed as specified in the REPORT operand. Only delete requests also create a full log (*FULL) when REPORT=*SUMMARY is specified to ensure that in all cases the report contains the volumes which have become free.

The information output for the save file and the individual files is similar to the ARCHIVE lists. The only exemption are the different page header and additional HSMS-specific messages.

The OUTPUT suboperand defines the output destination:

- OUTPUT=*STD corresponds to either OUTPUT=*PRINTER or OUTPUT=*MAIL depending on the value of the global HSMS parameter OUTPUT.
- OUTPUT=*PRINTER directs outputs to the printer. Output is in this case always to the printer and not to the *SYSLST system file.
- OUTPUT=<filename> directs outputs to a file. If the specified file is an already cataloged, non-empty SAM file, it is extended instead of overwritten (OPEN EXTEND). If this file is not a SAM file, the system displays an error message and output is re-directed to *PRINTER.
- OUTPUT=*LIBRARY-ELEMENT(LIBRARY=<library>,ELEMENT=<element>) results in a PLAM library element being output. A type P element is created with a version which contains the user ID plus the date and time. If output to the library element is not possible, an error message is issued and the output is redirected to *PRINTER.
- OUTPUT=*MAIL results in the report being sent as an email attachment. The email is sent to the address which is entered in the caller’s user entry (see the MAIL-FILE command). If the report cannot be sent by email, an error message is output and the output is redirected to *PRINTER.

Reports which are to be printed at the server-side have a different layout. The layout depends on the type of workstation on which the files are to be processed (BS2000 or UFS report).

Reports for action statements entered at an active node are formatted using predefined structures and written in response files. The contents of these response files are then sent by the communication task via the network to the inodes. The client converts the report so that it has the correct layout.

In the case of HSMS requests which are also monitored on the BS2000 backup monitor on the SE server, the reports are also stored as a PDF file under the user ID SYSHSMS. Only the system can access this file.

The structure of an HSMS report is explained below, with reference to a backup example.

1st page of the report

The *header* is the same for all pages of the report. Its title line contains

- the name of the action statement that initiated the request
- the HSMS version used
- the date and time
- the page number of the report.

The second line contains information on the request:

- the environment in which the statement was processed (SF or SM)
- the name and date of the request
- the user ID under which the statement was executed
- the request status (here COMPLETED WITHOUT ERROR).

Then the HSMS statement entered is logged. The complete HSMS statement used to create the request is output under STATEMENT LISTING.
This is followed by other operands and their values (ENVIRONMENT, ARCHIVE-NAME, SAVE-FILE ATTRIBUTES ...).

```
A***   BACKUP - FILES                HSMS V12.0       FULL         REPORT  ***2016-09-17
12:05:38     PAGE    1
     REQUEST-ENVIRONMENT=SF
     REQUEST-NAME=BCF#4362   REQUEST-DATE=2016-09-17 12:04:52   USER-ID=SYSHSM
     REQUEST-STATE=COMPLETED WITHOUT ERROR
     STATEMENT LISTING:
     BACKUP-FILES  FILE-NAMES=(¤MANUAL.FILE.0*,¤USEROLD.FILE.1*,¤USERNEW.FILE.2*),SELECT-
FILES=*ALL-FILES,
                ARCHIVE-NAME=MANUAL.BAC,SAVE-FILE=*NEW(RETENTION-PERIOD=0),
                TO-STORAGE=*S2-STORAGE-LEVEL(VOLUMES=(WORK12,WORK13,WORK14)),
                OPERATION-CONTROL=*PARAMETERS(PARALLEL-RUNS=3,REPORT=*FULL,OUTPUT=HSMS.MAN.R.
BCF.1)

                 ENVIRONMENT                          : SF
                 ARCHIVE-NAME                         : ¤SYSHSMS.MANUAL.BAC
                 SAVE-FILE ATTRIBUTES                 : NEW
                 TO-STORAGE                           : S2-STORAGE-LEVEL
                 DEVICE-TYPE                          : TAPE-C4
                 RETENTION-PERIOD                     : 0
                 SELECT-FILES PARAMETER
                 ALL-FILES
                 MAX-BACKUP-CLASS                     : STD
```

2nd page of the report

The second page lists the ARCHIVE messages relating to the acceptance of the request (ARC0002) and the generation of subtasks (ARC0003). In the case of some HSMS statements (e.g. //MODIFY-ARCHIVE), these messages appear earlier, at the bottom of the first page.

These are followed by the name of the save file which was processed and output indicating which parallel run processed which volume:

Column	Meaning
SAVE FILE IDENTIFIER	Number of the save file (SFID) which was written or read
SUBSAVE NUMBER	Number of the parallel run
VSNS	Serial numbers of the volumes containing the save file. In the case of save files on S1, only the pubset ID is output, introduced by “0:”

This is followed in turn by a table listing the files and job variables which were processed with the following meanings:

Column	Meaning
FILE/JOB VARIABLE NAME	Names of the files and job variables processed
VERS	Version number of the saved file
LASTPG/SIZE	File size in PAM pages (last-page pointer); for job variables the size in bytes
SAVE TYPE	Save type (FULL, PART, MIGF, FGGI, CNS, OPER, FNOD)
INPUT VSN	Serial number of the first input volume
DEV TYP	Device type of the input volume (D=disk, T=tape, C=catalog)
SUBSAVE	Number of the parallel run in which the volume was written/read
OUTPUT VSN(S)	Serial numbers of the output volumes

The IDs to which the listed files belong are shown in a header line, e.g.:

```
***    CATALOG - RD3A   USER - MANUAL    ***

A***  BACKUP - FILES     HSMS V12.0   FULL   REPORT  *** 2016-09-17  12:05:38   PAGE    2
     REQUEST-ENVIRONMENT=SF
     REQUEST-NAME=BCF#4362   REQUEST-DATE=2016-09-17 12:04:52   USER-ID=SYSHSMS
     REQUEST-STATE=COMPLETED WITHOUT ERROR
 %  ARC0002 STATEMENT ACCEPTED. ARCHIVE SEQUENCE NUMBER 'A.160917.120456', VERSION '12.0'
 %  ARC0033 ARCHIVE SUBTASK TSN '4487' GENERATED
 %  ARC0033 ARCHIVE SUBTASK TSN '4488' GENERATED
 %  ARC0033 ARCHIVE SUBTASK TSN '4489' GENERATED

           SAVE FILE IDENTIFIER - S.160917.120456  SAVE-VERSION-DATE=16-09-17  SAVE-VERSION-
TIME=12:04:57
            SUBSAVE
             NUMBER       VSNS
                0        WORK14
                1        WORK13
                2        WORK12

             SAVE FILE IDENTIFIER - S.160917.120456  SAVE-VERSION-DATE=16-09-17  SAVE-
VERSION-TIME=12:04:57
                       ***   CATALOG - RD3A      USER - MANUAL     ***
           FILE/JOB VARIABLE NAME                   LASTPG/  SAVE   INPUT DEV  SUB  OUTPUT
                                                VERS     SIZE  TYPE    VSN  TYP SAVE  VSN(S)
 FILE.01                                           1        5  FULL  RD3A.0  D     2  WORK12
 FILE.02                                           1        4  FULL  RD3A.0  D     2  WORK12
 FILE.03                                           1        4  FULL  RD3A.0  D     2  WORK12
 FILE.04                                           1       14  FULL  RD3A.0  D     2  WORK12
 FILE.05                                           1        9  FULL  RD3A.0  D     2  WORK12
 FILE.06                                           1        1  FULL  RD3A.0  D     2  WORK12
 FILE.07                                           1       21  FULL  RD3A.0  D     2  WORK12
```

3rd and any further pages of the report

If the list of the files and jobs variables processed extends beyond the second page, further pages are added as needed, e.g.:

```
A***   BACKUP - FILES                HSMS V12.0        FULL         REPORT  *** 2016-09-17
12:05:38    PAGE    3
     REQUEST-ENVIRONMENT=SF
     REQUEST-NAME=BCF#4362   REQUEST-DATE=2016-09-17 12:04:52   USER-ID=SYSHSMS
     REQUEST-STATE=COMPLETED WITHOUT ERROR
           SAVE FILE IDENTIFIER - S.160917.120456  SAVE-VERSION-DATE=16-09-17  SAVE-VERSION-
TIME=12:04:57
                        ***   CATALOG - RD3A      USER - USERNEW    ***
           FILE/JOB VARIABLE NAME                   LASTPG/  SAVE   INPUT DEV  SUB  OUTPUT
                                                VERS     SIZE  TYPE    VSN  TYP SAVE  VSN(S)
 FILE.21                                           1        5  FULL  RD3A.0  D     1  WORK13
 FILE.22                                           1        4  FULL  RD3A.0  D     1  WORK13
 FILE.23                                           1        4  FULL  RD3A.0  D     1  WORK13
 FILE.24                                           1       14  FULL  RD3A.0  D     1  WORK13
 FILE.25                                           1        9  FULL  RD3A.0  D     1  WORK13
 FILE.26                                           1        1  FULL  RD3A.0  D     1  WORK13
 FILE.27                                           1       21  FULL  RD3A.0  D     1  WORK13

           SAVE FILE IDENTIFIER - S.160917.120456  SAVE-VERSION-DATE=16-09-17  SAVE-VERSION-
TIME=12:04:57
                        ***   CATALOG - RD3A      USER - USEROLD    ***
           FILE/JOB VARIABLE NAME                   LASTPG/  SAVE   INPUT DEV  SUB  OUTPUT
                                                 VERS     SIZE  TYPE    VSN TYP SAVE  VSN(S)
 FILE.11                                           1        5  FULL  RD3A.0  D   0  WORK14
 FILE.12                                           1        4  FULL  RD3A.0  D   0  WORK14
 FILE.13                                           1        4  FULL  RD3A.0  D   0  WORK14
 FILE.14                                           1       14  FULL  RD3A.0  D   0  WORK14
 FILE.15                                           1        9  FULL  RD3A.0  D   0  WORK14
 FILE.16                                           1        1  FULL  RD3A.0  D   0  WORK14
 FILE.17                                           1       21  FULL  RD3A.0  D   0  WORK14
      NUMBER OF FILES=             21 GLOBAL SIZE=         174 START=2016-09-17 12:04:53
END= 2016-09-17 12:05:38
```

The penultimate line of the final page of the report contains the following information:

NUMBER OF FILES=	Number of files and job variables processed
GLOBAL SIZE=	Total size of all files and job variables
START=	Date and time at which the request was made
END=	Date and time at which the request was terminated

The last page of the HSMS report concludes with the following line:

```
***     E N D  O F                HSMS V12.0        FULL       REPORT  *** 2016-09-17  12:
05:38    ***
```

4.11 Job variable for request monitoring

With the following HSMS action statements a job variable for request monitoring can be specified. (CONTROL-JV operand within the runtime control under OPERATION-CONTROL=*PARAMETERS(...)):

- ARCHIVE-FILES
- ARCHIVE-NODE-FILES
- BACKUP-FILES
- BACKUP-FILE-VERSIONS
- BACKUP-NODE-FILES
- COPY-EXPORT-SAVE-FILE
- COPY-NODE-SAVE-FILE
- COPY-SAVE-FILE
- EXPORT-FILES
- IMPORT-FILES
- MIGRATE-FILES
- MOVE-SAVE-FILES
- RECALL-MIGRATED-FILES
- REORGANIZE-VERSION-BACKUP
- REPAIR-CATALOG-BY-RESTORE
- REPLACE-SAVE-FILE-BY-RESTORE
- RESTORE-FILES
- RESTORE-LIBRARY-ELEMENTS
- RESTORE-NODE-FILES
- UPDATE-EXPORT-SAVE-FILE

The job variable is not a monitoring job variable and is therefore processed exclusively by HSMS. In asynchronous request processing, this job variable provides the user with the result of important processing steps in HSMS /ARCHIVE. Among other things the job variable supplies information on the initialization of a CCOPY session and on automatic duplication to a shadow archive.

4.11.1 Processing of the job variable by HSMS

HSMS deletes the contents of the job variable while the request is being checked and accepted. If the job variable is not yet entered in the catalog, HSMS creates it. If at this point an error occurs during access to the job variable, the request is rejected.

If HSMS wants to update the job variable at a later stage and an access error occurs at that time, the error is logged. The request is processed normally.

4.11.2 Contents of the job variable

The job variable is composed of fixed-length fields. While the request is being checked, each field is initialized with blanks.

The job variable for monitoring HSMS requests is not governed by the conventions for monitoring job variables. In particular, the field containing the request status does not begin at position 1 (see also the “Job Variables” manual [23]).

Field	Pos.	Length	Value	Meaning
CCS INIT STATUS	1	1	Blank	Request without active CCOPY function or (yet) incomplete
			T	Terminated successfully
			A	Terminated abnormally
AUTOMATIC DUPLICATION	2	1	Blank	Request without automatic duplication to a shadow archive or not (yet) started
			S	Started
TSN SERVERTASK	3	4	txt	TSN of the server task
REQUEST DATE and TIME	7	14	dig	Date and time of the request Format: YYYYMMDDHHMMSS
SFID	21	14	dig	Save file ID Format: YYYYMMDDHHMMSS
SVID	35	14	dig	Save version ID Format: YYYYMMDDHHMMSS
(LOCAL) STATUS	49	11	txt	See STATUS in table below
(LOCAL) SUB-STATUS	60	17	txt	See SUBSTATUS in table below
FULL-FROM-FULL-STATE	77	1	Blank 2	Save of backup not yet started Save of backup running
(REMOTE) STATUS	78	11	txt	See table below
(REMOTE) SUB-STATUS	89	17	txt	See table below
SERVER-NAME	106	8	txt	BCAM name of the host that is currently processing the request
CCOPY SESSION-ID	114	8	txt	Identifier for the CCOPY save run

The following table displays the possible values for STATUS and SUBSTATUS:

STATUS	SUBSTATUS
INCOMPLETE	– – –
ACCEPTED	– – –
STARTED	COLLECTED START-ARCHIVE ARCHIVE-COMPLETED START-REPORT SENT-TO-MASTER MASTER-REPLIED MASTER-TIMEOUT MASTER-NO-CONNECT IN-TRANSMIT SENT-TO-BACK-SERV BACK-SERV-REPLIED BACK-SERV-NO-CONN BACK-SERV-TIMEOUT
COMPLETED	– – – WITH-WARNINGS WITH-ERRORS
INTERRUPTED	MASTER-TIMEOUT MASTER-NO-CONNECT START-ARCHIVE START-REPORT MASTER-REPLIED BACK-SERV-REPLIED BACK-SERV-NO-CONN BACK-SERV-TIMEOUT
CANCELLED	– – –

When HSMS halts the processing of a job, the status of the job is either COMPLETED, INTERRUPTED or CANCELLED (for more detailed information on possible request statuses, refer to section "Information about requests"). The status that is sent to the job variable can then be queried using a procedure.

If a request with the status ACCEPTED or STARTED is deleted by another task, the request monitoring job variable is updated. It is set to the status COMPLETED and the substatus WITH-ERRORS.
If requests which have a different status are deleted, the job variable is not updated.

1. **CCS INIT STATUS**
   This field provides information on the initialization status of a CCOPY session. This makes it possible to optimize CCOPY processing: The application is closed before the BACKUP-FILES statement is entered, and is immediately reopened once initialization of the CCOPY session has been terminated successfully (even before the request is accepted!).
2. **AUTOMATIC DUPLICATION**
   In the case of a backup or long-term archiving with automatic duplication, indicates that the save file of the original archive is complete and that automatic duplication is starting. This means that HSMS/ARCHIVE no longer accesses the duplicated files. Consequently, the applications that access these files can be restarted.
3. **TSN SERVERTASK**
   TSN of the server task that is processing the request.
4. **REQUEST DATE and TIME**
   This information can be helpful for HSMS statements in which a request can be selected via the date and time, e.g. SHOW-REQUESTS, DELETE-REQUESTS, etc. .
5. **SFID**
   ID of the save file that is assigned by the request. This is only present for the following HSMS statements: ARCHIVE-FILES, ARCHIVE-NODE-FILES, BACKUP-FILES, BACKUP-FILE-VERSIONS and BACKUP-NODE-FILES.
6. **SVID**
   ID of the save version that is assigned by the request. This is only present for the following HSMS statements: ARCHIVE-FILES, ARCHIVE-NODE-FILES, BACKUP-FILES, BACKUP-FILE-VERSIONS and BACKUP-NODE-FILES.
7. **(LOCAL) STATUS**
   Current status of the request. This field can be used, for example, for S procedures which monitor a request.
   See also the HSMS statement SHOW-REQUESTS: STATUS (“HSMS Vol. 2” [1])
8. **(LOCAL) SUB-STATUS**
   Provides additional information about the status of the request.
   See also the HSMS statement SHOW-REQUESTS: SUBSTATUS (“HSMS Vol. 2” [1])
9. **FULL-FROM-FULL-STATE**
   This status relates to saving with FROM=*LATEST-BACKUP-OR-S0. The field contains a 2 if backing up of S0 has been completed and saving of earlier backups has been started.
10. **(REMOTE) STATUS**
    Current status of the request on the remote server.
    For local requests, this field is of no significance.
11. **(REMOTE) SUB-STATUS**
    Supplies additional information on the status of the request on the remote server.
    For local requests, this field is of no significance.
12. **SERVER-NAME**
    BCAM name of the remote server that is processing the request.
    For local requests, this field is of no significance.

13. **CCOPY-SESSION-ID**
    This field is the identifier for a CCOPY save run with BACKUP-FILES CURRENT-COPY=YES or BACKUP-FILE-VERSIONS BACKUP-FILES CURRENT-COPY=YES.

The following table provides important information about where and when the content of the job variable is available.

Field	Location of the information	Time available
CCS INIT STATUS	User-side	During acceptance of request
AUTOMATIC DUPLICATION	Server-side	After first backup
TSN SERVERTASK	Server-side	After start of request
REQUEST DATE and TIME	User-side	After acceptance of request
SFID	Server-side	After backup
SVID	Server-side	After backup
(LOCAL) STATUS	User-side	Constantly changing
(LOCAL) SUB-STATUS	User-side	Constantly changing
FULL-FROM-FULL-STATE	Server-side	After incremental backup
(REMOTE) STATUS	Server-side	Constantly changing
(REMOTE) SUB-STATUS	Server-side	Constantly changing
SERVER-NAME	Server-side	After start of request
CCOPY-SESSION-ID	User-side	From acceptance of the request

4.11.3 Job variable for shared pubsets

Some action statements can also be processed in master mode or on the backup server (see section “Working with shared pubsets”). This means that the request is processed by a host, namely the master of the shared pubset or the backup server, other than the one that is affected by this request.

If a job variable is used to monitor such a request in master mode, some parts of the job variable are set by the slave sharer and some by the master sharer. If the backup server processes such a request, some parts of the job variable are set by the local system, some by the backup server. In such cases, you should use a job variable which is available on both sides. The best solution is to use a job variable at the pubset that is already affected by this request (the pubset at which the files are to be backed up, restored, recalled etc.).

4.12 Aliases for BS2000 files and job variables

This section describes the effects of the software product ACS (**A**lias **C**atalog **S**ervice) on HSMS. ACS is a subsystem of BS2000 and is designed for managing the aliases of BS2000 files and job variables. An alias is a substitute file or job variable name chosen virtually at will by the user.

The system administrator must start the ACS subsystem before aliases can be used. In addition, an alias catalog that unequivocally defines the correlation between aliases and file or job variable names must be generated for the current task.

HSMS accepts aliases in the FILE-NAMES or JV-NAMES operand of the HSMS statements concerned. Aliases are also accepted for the names of directory files.

Internally, HSMS uses only file or job variable names from the catalog and no aliases. For this reason, only file names are written into the HSMS request file and the ARCHIVE checkpoint file.

Any attempt to change an alias after an HSMS statement has become effective is ignored. This also applies to changes attempted before an interrupted request is resumed.

In reports, the alias appears in the user statement part; the file or job variable name is output in the result part.

For more information on aliases and administrating the alias catalog, see the section “ACS” in the “Introductory Guide to DMS” [4].

5 Management of HSMS

The HSMS administrator has a range of tasks to perform during operation. Some of these tasks have already been described in "HSMS archives" and "HSMS functions", e.g. the creation of default system archives and the definition of tape sessions.

The present chapter describes a number of additional tasks. These tasks include dividing the pubsets into a storage hierarchy according to the requirements of the system involved.

This chapter also contains a description of migration control, because the control options for this function are extremely varied. Management of the migration archives in particular requires a great deal of effort on the part of the HSMS administrator (unlike the other basic functions). Moreover, the relationship between data backup and migration is discussed.

Aspects of data privacy are addressed in a separate section.

This chapter concludes with an example. It explains how to set up a storage hierarchy and how to create and allocate system archives for a BS2000 configuration.

5.1 HSMS operating modes

HSMS can run in various operating modes; these are defined by the system administrator by means of the HSMS statement START-HSMS or by the HSMS administrator by means of the HSMS statement MODIFY-HSMS-PARAMETERS.

5.1.1 DEFINE-SHOW mode

All inquiry functions are permitted in DEFINE-SHOW mode (OPERATIONAL-MODUS=*DEFINE-SHOW). So, too, are the modification of HSMS control and pubset parameters and the creation and modification of archives. Action statements are rejected.
This mode is useful when getting to know HSMS.

5.1.2 SIMULATION mode

In SIMULATION mode (OPERATIONAL-MODUS=*SIMULATION), all HSMS statements are permitted but action statements are only simulated. This means that, although the syntax of HSMS statements is checked and the requests are generated and written to the request file.

The server tasks perform no more than a kind of pseudo processing. The requests are set to the COMPLETED state but no ARCHIVE actions are initiated, i.e. no data manipulation takes place. This mode is useful for familiarizing oneself with the functions of the various action statements without actually accessing user files.

Extreme caution should be exercised when using the HSMS statement MODIFY-ARCHIVE in simulation mode since certain activities are not just simulated but are actually performed (e.g. sharing backups with //MODIFY-ARCHIVE SAVE-FILES= *DELETE).

Action statements from active nodes are rejected when HSMS is running in simulation mode.

5.1.3 OPERATION mode

In OPERATION mode (OPERATIONAL-MODUS=*OPERATION), the full range of HSMS functions is available. This is the normal HSMS operating mode after initiation.

5.2 Managing the storage hierarchy in an environment with SF pubsets

HSMS implements its functions in a three-level storage hierarchy. In this storage hierarchy the disk and tape storages are subdivided according to their availability, access time and the costs (see section "Storage hierarchy").

Storage level S2

Storage level S2 need not be explicitly created in HSMS. The S2 level is always defined in any HSMS system. The default value for volumes requested for write requests to S2 can be defined by means of

```
//MODIFY-HSMS-PARAMETERS DEFAULT-HSMS-STORAGE=*PAR -
// (S2-DEVICE-TYPE=<c-string>)
```

HSMS supports volumes of the class “TAPE”, LTO magnetic tape cartridges and emulated tape devices of ETERNUS CS Data Protection Appliances (see section "Level S2 volumes").

5.2.1 Managing SF pubsets

The pubset is an important management unit under HSMS. A pubset is a set of public disk storages with a common catalog. It is defined by its pubset ID. HSMS offers full support of MPVS (multiple public volume set) systems.

HSMS also supports shared pubset systems: the backup, version backup, archival, and migration functions can be used to their full extent, although some restrictions in selecting files must be observed (see section "Working with shared SF pubsets").

Immediately following HSMS installation, only the home pubset is under HSMS control. All other pubsets are in the UNDEFINED state and have no entry in the HSMS control file.

A pubset in an MPVS system can either remain outside HSMS control in an undefined state or be assigned by the HSMS administrator to either of the two online storage levels S0 and S1 (see next page).

5.2.1.1 Pubsets outside HSMS control

The HSMS functions can be used even for pubsets that are not under HSMS control, i.e. not assigned to any storage level, provided that the pubsets are available in the system.

Unless otherwise specified, the backup and archival functions save to the global default system archives defined in the HSMS control file. Version backup and file migration is, however, not possible for these pubsets.

Files that reside on a pubset outside HSMS can only be saved with the Concurrent Copy function if the SYSHSMS ID can be accessed from the pubset. This is because the work file for Concurrent Copy and the metafiles are created on the SYSHSMS ID.

5.2.1.2 Assigning a pubset to storage levels S0 and S1

A pubset must be assigned to storage level S0 if files of this pubset are to be capable of migration or if pubset-specific default system archives are to be defined that differ from the globally defined default system archives.

If version backups of the files from a pubset are planned to be done, the pubset should be defined as S0. Note that only pubset-specific default system version backup archives can be defined. There is no globally defined archives for version backups.

A pubset of storage level S0 can be assigned an S1 pubset as a background level to which files can be backed up or migrated. A pubset is defined as a global pubset with:

```
//MODIFY-HSMS-PARAMETERS DEFAULT-HSMS-STORAGE=*PAR -
// (S1-PUBSET-ID=<S1-pubset-id>)
```

This global pubset is used as a background level by all S0 pubsets unless a different S1 pubset is defined by means of:

```
//MODIFY-PUBSET-PARAMETERS PUBSET-ID=<S0-pubset-id>, -
// STORAGE-LEVEL=*S0(S1-PUBSET-ID=<S1-pubset-id>)
```

Systems under HSMS management can get by without definition of an S1 pubset: all basic functions can use storage level S2 or private disks instead.

5.2.1.3 Assigning an SM pubset to storage level S1

As the size of an SF pubset is limited to 4TB, it can occur that one SF pubset is not sufficient for the backup and migration to storage level S1. In HSMS V10.0 and higher (SAVE-FILE-PROCESSING=*HSMS-V10-COMPATIBLE parameter), an SM pubset can also be assigned as storage level S1 in this case. With the exception of the control volume set and S1 volume set, all volume sets of the SM pubset are used here. Such an SM pubset is called an S1-SM pubset. The maximum possible size of an SM pubset is almost 1000TB.

To prevent the control volume set of an S1-SM pubset from being occupied with save files, use of the control volume set must be restricted beforehand as follows:

```
/MODIFY-PUBSET-RESTRICTIONS PUBSET=<cat_id of S1-SM-pubset> -
/       ,PUBSET-TYPE=*SYSTEM-MANAGED(VOLUME-SET=<control-volume-set> -
/       ,RESTRICTION=*NEW-FILE-ALLOCATION(MODE=*PHYSICAL-ONLY))
```

> It is urgently recommended that an SM pubset that is to be used as an S1-SM pubset consists exclusively of volume sets in NK2 format.
>
> S1-SM pubsets cannot be used in an SM environment, i.e. an S1-SM pubset cannot be assigned to any other SM pubset than storage level S1.

Saving to an S1-SM pubset

Saving to an S1-SM pubset is performed using the BACKUP-FILES, BACKUP-FILE-VERSIONS, ARCHIVE-FILES, COPY-SAVE-FILE, MOVE-SAVE-FILE, REORGANIZE-VERSION-BACKUP or MIGRATE-FILES statement with the specification TO-STORAGE=*S1-STORAGE-LEVEL when an SM pubset is set globally for the entire system or as storage level S1 for individual pubsets.

The save file for such a save run is configured with the following name:

`ARCHIVE.SAVE.FILE.`*`date`*`.`*`time`*`.`*`subsave#`*`.`*`seq#`*

Key:

date.time	Generation time in the format yymmdd.hhmmss
subsave#	Subsave number of the run which created this save file
seq#	Sequence number of the save file with the same time stamp (0..F)

If the capacity of the volume set on which the save file is saved, is insufficient for the backup, another save file is created on a different volume set of the S1-SM pubset.

Its name differs from the original save file by the sequence number *seq#* being incremented by 1.

A subtask can create up to 16 such save files.

Configuring an SM pubset as storage level S1

An SM pubset can be used as storage level S1 of an SF pubset only when the global HSMS parameter SAVE-FILE-PROCESSING is set to *HSMS-V10-COMPATIBLE (MODIFY-HSMS-PARAMETERS statement). If SAVE-FILE-PROCESSING=*HSMS-V9-COMPATIBLE is set, the attempt to save to an S1-SM pubset is rejected.

> Changing this setting has repercussions on the functionality of HSMS. For details, see section "Processing mode for save files".

The following steps are required to configure an SM pubset as storage level S1:

- Place the SM pubset under HSMS control
  (CREATE-SM-PUBSET-PARAMETERS statement).
- Define the SM pubset as storage level S1
  (MODIFY-PUBSET-PARAMETERS statement).
- Add the SM pubset as storage level S1 globally or on a pubset-specific basis
  (MODIFY-HSMS-PARAMETERS or MODIFY-PUBSET-PARAMETERS statement).

5.2.1.4 Placing pubsets under HSMS management and defining the parameters

The HSMS statement MODIFY-PUBSET-PARAMETERS is used by the HSMS administrator to place pubsets under HSMS management and to set the pubset parameters. All modifications made are valid from the moment of statement entry for the current HSMS session and any following sessions.

A pubset can be placed under HSMS management only under the following conditions:

- The pubset must be available locally, i.e. it must have been imported already by means of IMPORT-PUBSET.
- The user ID SYSHSMS must be created on the pubset before it can be assigned to a storage level.
- A pubset must be set up as an S1 pubset before it can be assigned as an S1 pubset to an S0 pubset.

It is necessary to take into account some special features of pubsets when placing them under HSMS management. This action should therefore only be performed together with the system administrator.

An entry relating to a particular pubset is not made in the HSMS control file unless at least one parameter differs from the default setting.

The following default values apply to pubsets being placed under HSMS management for the first time:

Operand value	**Default value for pubsets under HSMS management**
STORAGE-LEVEL	*UNDEFINED
SYSARCHIVE	global definition from the HSMS control file
SYSBACKUP	global definition from the HSMS control file
SYSVERSION	*UNDEFINED
if S0: – S1-PUBSET-ID – SYSMIGRATE – MIGRATION	 global definition from the HSMS control file global definition from the HSMS control file *ALLOWED

A pubset can be assigned to a storage level, while the pubset itself can be assigned default backup, version backup and archival system archives.

For S0 pubsets, it is also possible to define further parameters for migration (see section "Migration control and management of migration archives").

Notes

- Save files on an S1 pubset are created with the same user ID as the archive directory that manages the save file. Therefore, this user ID must also exist on the pubset that is assigned to storage level S1.
- It should be permissible to exceed the storage space limit (PUBLIC-SPACE-EXCESS).

5.2.1.5 Information about parameters and pubset utilization

The HSMS statement SHOW-PUBSET-PARAMETERS can be used by the HSMS administrator to obtain information about the parameters of all pubsets under HSMS management (see “HSMS Vol. 2” [1], HSMS statement SHOW-PUBSET-PARAMETERS).

The HSMS statement SHOW-PUBSET-USAGE can be used by the HSMS administrator to obtain information about the occupancy of all available pubsets, including those not under HSMS management (see “HSMS Vol. 2” [1], HSMS statement SHOW-PUBSET-USAGE).

The information output is particularly useful for pubset management and for preventing pubset saturation by means of timely migration and archival (see section "Migration control and management of migration archives").

5.2.1.6 Releasing pubsets from HSMS management

An entry relating to a particular pubset is not made in the HSMS control file unless at least one parameter differs from the default setting. The entry can be deleted again by resetting all parameters to the original default values. Although it is immediately deleted from the control file, the entry for the pubset is still displayed in response to pertinent HSMS statements during the current HSMS session.

5.2.2 Measures to be taken in response to special situations

- Restore following an S0 crash
- Restore following a crash on S1 or S2
- Reorganization of storage level S1
- Reorganization of storage level S2
- Moving user IDs to another pubset
- Reorganizing user or catalog IDs

5.2.2.1 Restore following an S0 crash

Here, it is necessary to differentiate between two cases:

1. Storage levels S1 and S2 are undamaged

   The following statements are required for each damaged pubset, e.g. pubset A:

   ```
   //RESTORE-FILES FILE-NAMES=:A:$*., -
   //  MIGRATED-FILES=*CATALOG-ONLY, -
   //  JV-NAMES=:A:$*., -
   //  ...
   ```

   Normal processing can be continued on all undamaged pubsets.

   If the home pubset is damaged, it is first necessary to restore and call the procedure that defines the archive. The directory file for backups must also be restored.

2. Storage levels S1 and S2 are also damaged

   First, it is necessary to restore processing level S0. Afterwards the storage levels S1 and S2 must be restored (see the following section).

5.2.2.2 Restore following a crash on S1 or S2

The following statements are required for each S0 pubset affected by the crash (e.g. A):

```
//REPAIR-CATALOG-BY-RESTORE S0-PUBSET-ID=A, -
//  TO-STORAGE=..., -
//  ARCHIVE-NAME=*SYSBACKUP, -
//  ...
```

5.2.2.3 Reorganization of storage level S1

After migrated files are deleted or recalled, the storage space in the save file is not released by ARCHIVE. This means that, after some time, a good deal of storage space on S1 is being wasted. The HSMS administrator can use the //SHOW-PUBSET-USAGE ... INFORMATION=*REUSABLE-S1-SPACE statement to obtain information about the storage space to be released at S1 level. In cases where management is decentralized, the following statements are required for each S1 pubset:

```
//MIGRATE-FILES FROM-STORAGE=*S1-STORAGE-LEVEL -
//  (S1-PUBSET-ID=..., -
//   ..., -
//  TO-STORAGE=S1-STORAGE-LEVEL, -
//  ARCHIVE-NAME=<Name des in Beziehung stehenden Archivs>)
```

5.2.2.4 Reorganization of storage level S2

After a migrated file has been recalled, the storage space on the volumes created by ARCHIVE is not released. In this way a large amount of storage space on S2 is wasted. With the next statement the files are migrated from storage level S2 to S2 and reorganized in the process. This is implemented by copying to a new save file in which only the migrated files are contained. The original save files are automatically deleted irrespective of their retention period. For each migration archive used the following statements are required:

```
//MIGRATE-FILES FROM-STORAGE=*S2-STORAGE-LEVEL -
//  (SAVE-FILE-ID=..., -
//   UNUSED-SPACE=..., -
//   ..., -
//  ARCHIVE-NAME=<Name des in Beziehung stehenden Archivs>)
```

The default save file is not taken into account here.

5.2.2.5 Moving user IDs to another pubset

If a user ID is to be moved from one pubset to another – taking due account of migrated files – we recommend the following procedure:

- make a backup to a temporary archive (including the data of the migrated files)
- assign adequate storage space on the target pubset to the user ID prior to transfer
- run a RESTORE renaming the catalog ID (migrated files with data); the files contain a new CREATION-DATE, LAST-ACCESS-DATE, etc.
- after backing up the files, re-migrate the files which were originally migrated
- delete the temporary archive
- reset the storage space limit.

5.2.2.6 Reorganizing user or catalog IDs

Changing the user ID while retaining the catalog ID:

Three variants are distinguished here:

1. Restore with renaming of the user ID from archives with unmodified directory
   This variant has the disadvantage that when restoring from backup archives you must work in a different way for save files: before changing the user ID with renaming and after changing it without renaming. Also, with SHOW-ARCHIVE the files saved before the change remain assigned to the old user ID.
2. Setting up new archives with a new directory for the new ID and copying save files from the old archive (before the ID is changed) to a new archive with renaming of the user ID
   This variant is only permitted for the HSMS administrator. When save versions are copied using this procedure from backup incremental saves it must be ensured that the basic full backup is also copied in this way.
3. In the unmodified backup archive, creating a current full backup after the change using backup with FROM-LATEST-OR-S0 renaming of the user ID
   This variant is also only permitted for the HSMS administrator.

General rule

Migrated files must be recalled to S0 before changing the user ID.

Changing the catalog ID (without modifying the pubset type)

Four variants are distinguished here:

1. Restore with renaming of the catalog ID from archives with unmodified directory
   This variant has the disadvantage that when restoring from backup archives you must work in a different way for save files: before changing the catalog ID with renaming and after changing it without renaming. Also, with SHOW-ARCHIVE the files saved before the change remain assigned to the old catalog ID.
2. Setting up new archives with a new directory for the new catalog ID and copying save files from the old archive (before the catalog ID is changed) to a new archive with renaming of the catalog ID
   This variant is only permitted for the HSMS administrator. When save versions are copied using this procedure from backup incremental saves it must be ensured that the basic full backup is also copied in this way.
3. In the unmodified backup archive, creating a current full backup after the change using backup with FROM-LATEST-OR-S0 renaming of the catalog ID
   This variant is also only permitted for the HSMS administrator.
4. Converting the directory of an archive to the new catalog ID with DIRCONV
   This variant is possible if the directory only contains entries with the old catalog ID (for SM pubset or for decentralized organization with SF pubsets).

The change from the old to the new catalog ID is possible without any problem for the save files on disk.

Procedure for migrated files

With variants 1 to 3 all migrated files must be recalled to S0 before the catalog ID is changed.

With variant 4 it is sufficient (in addition to the DIRCONV conversion to the new catalog ID for the migration directory) to restore only the catalog entry of the migrated files - not the files.

5.3 Management of the storage hierarchy in an environment with SM pubsets

HSMS implements its functions in a three-level storage hierarchy. In this storage hierarchy the disk and tape storages are subdivided according to their availability, access time and the costs (see section “Storage hierarchy”).

5.3.1 Processing SM pubsets

HSMS offers a straightforward user interface for processing SM pubsets. Because certain metadata is required on the SM pubset for backup, archival and migration, the SM pubset must be under HSMS control to allow these activities to be carried out. SM pubsets not yet under HSMS control must first be placed under HSMS control using the statement CREATE-SM-PUBSET-PARAMETERS.

The table below provides an overview of how SM pubsets are processed.

	SM pubsets not under HSMS control	**SM pubsets under HSMS control**
HSMS	No migration: no migration to S1/S2; no migration to S0; no recall	Full processing including: migration to S1/S2; migration to S0; recall
	no backup; no version backup	version backup
	EXPORT/IMPORT possible	
	ARCHIVE-FILES/RESTORE-FILES possible with a long-term archive of the host	
	A restore together with renaming of the catalog ID is possible from each archive	A restore together with renaming of the catalog ID is possible from each archive
ARCHIVE	possible	possible

5.3.2 Scope of the HSMS operations

The following table shows for each function:

- in the first and second column the location of the metadata (archive definitions and requests).
- in the third column whether the requests can be restarted after the change of host.
- in the last column the environment that can be specified for files and job variables.

	Archive definition in the control file	**Request in the request file**	**Restart after change of host?**	**Scope of the files/JV**
Backup of SF pubsets	on the host	on the host	not possible	all SF pubsets
Backup of an SM pubset	on the SM pubset	on the SM pubset	possible	this SM pubset
Version backup of an SF pubset	on the host	on the host	not possible	this SF pubset
Version backup of an SM pubset	on the SM pubset	on the SM pubset	possible	this SM pubset
Migration of SF pubsets	on the host	on the host	not possible	all SF pubsets
Migration of an SM pubset	on the SM pubset	on the SM pubset	possible	this SM pubset
Long-term archival of the host	on the host	on the host	not possible	all pubsets
Archival of SM pubsets	on the SM pubset	on the SM pubset	possible	this SM pubset
Export/import	not relevant	always on the host	not possible	all pubsets

5.3.3 Converting an SM pubsets

For HSMS to be able to process an SM pubset, it has to be converted in two steps.

1. The SM pubset must be placed under HSMS management. This is just as important in the case of the new, empty SM pubset, as in that of a SM pubset created by a conversion from other pubsets. This is done using the HSMS statement CREATE-SM-PUBSET-PARAMETERS. This statement checks the status of the SM pubset and activates the “under HSMS control” attribute for the SM pubset. The statement CREATE-SM-PUBSET-PARAMETERS includes the following actions:
   - create the control and request files of the SM pubset
   - When an S1 storage level is required:
     - explicitly define a volume set of the SM pubset as S1 volume set.
     - Or use *ALL-HSMS-CONTROLLED to define that all volume sets of the SM pubset can be used under HSMS control as extended S1 storage level (from HSMS V11.0 with parameter SAVE-FILE-PROCESSING=*HSMS-V10-COMPATIBLE and BS2000 OSD/BC V11.0A).
   - make the SM pubset’s Generic Catalog Facility
   - create the system archive of the SM pubsets for SM backup, SM migration and SF global or SM pubset archival
   - check and delete old HSMS SF definitions of the new SM pubset (if an SF pubset is being converted to an SM pubset under the same catalog ID)

   Notes

   - If the S1 volume set provided is not suitable, the CREATE-SM-PUBSET-PARAMETERS statement is rejected.
   - Creation of a system archive is not mandatory.
   - It is possible to specify just a subset of the archive parameters. The default values are set up as described in the CREATE-SM-PUBSET-PARAMETERS statement. If you do not want these default values, the system archives can be set up later with the CREATE-ARCHIVE statement and changed with the MODIFY-SM-PUBSET-PARAMETERS statement.
   - If an error occurs during the installation of the archive, a warning message will appear, but otherwise the process will continue.

   If the SM pubset contains no migrated files, the conversion is now complete.
2. If the SM pubset is created as a result of conversion of another pubset, and the SM pubset contains migrated files, or if existing archives of this SM pubset are to be used, then these archive directories must be combined and converted with DIRCONV.

A number of possible situations are explained below to illustrate the process of converting SM pubsets in various configurations.

5.3.3.1 Creating an SM pubset without migrated files

An SM pubset that contains no migrated files is to be converted. Here it is possible to create new default archives. If old archive directories are to be used again that were created before the conversion, another factor must be taken into account. The following are possible:

- Migration to a background level has not previously been used.
- All migrated files were recalled before creation of the SM pubset.

Examples

```
//CREATE-SM-PUBSET-PARAMETERS SM-PUBSET-ID=S,SYSBACKUP=*PAR,SYSMIGRATE=*PAR
```

SM pubset S is to be converted. The default archive for backup and migration and the related archive directories are created with a default name. The archive directories must be new.

```
//CREATE-SM-PUBSET-PARAMETERS SM-PUBSET-ID=S,SYSMIGRATE=*PAR, -
// SYSBACKUP=*PAR(ARCHIVE-NAME=A,DIRECTORY-NAME=D)
```

SM pubset S is to be converted. The default archive for migration and the related archive directory are created with a default name. A separate name is specified for the default archive for backup and for the related archive directory. The archive directories must be new.

5.3.3.2 Reusing an existing archive directory whose scope is restricted to an SM pubset

Existing archives are to be used as default archives for an SM pubset under HSMS control. All of the pubsets that use these archives are to be included in this SM pubset.

When creating archives (public or private) for backup or migration in an SM pubset environment, HSMS checks whether the specified archive directories contain file or job variable records for that SM pubset only. Long-term archives are not checked because they do not reside locally on an SM pubset.

The diagram below shows a typical standard use of archives by pubsets. Here, a single archive is used exclusively by pubsets M, N and O:

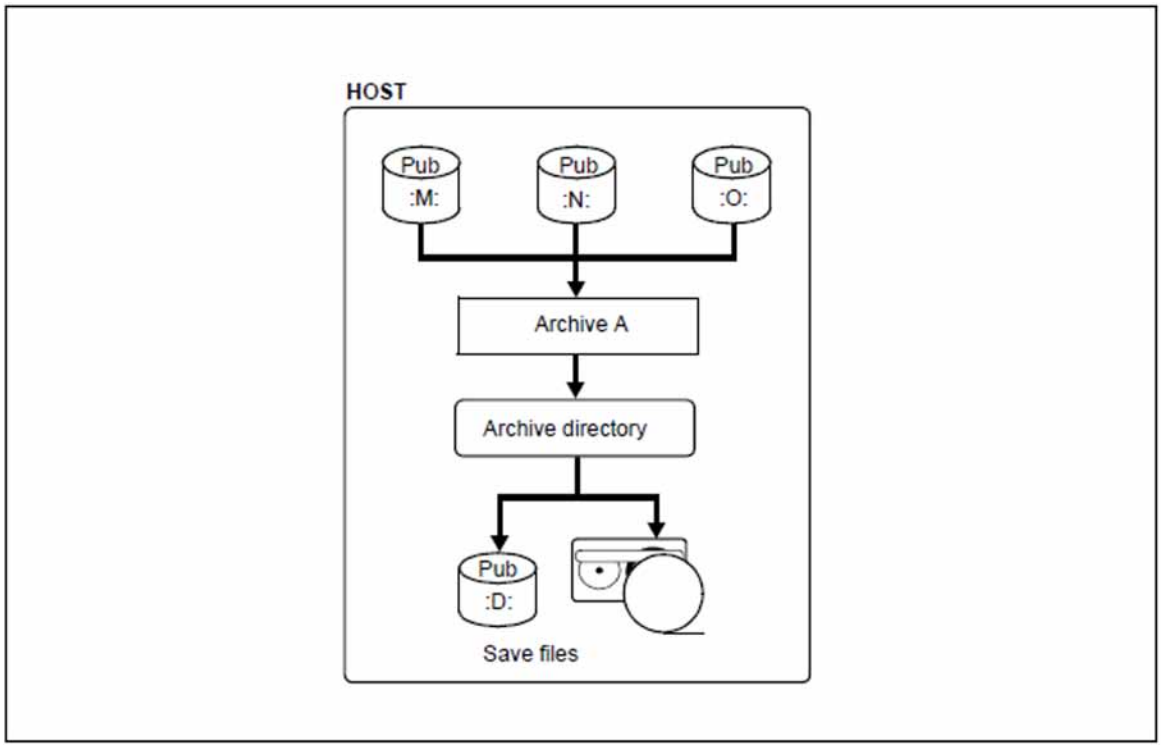


Figure 21: Archive with SF pubsets

After conversion of pubsets M, N and O to an SM pubset (with DMS statements), the archive definitions can no longer be used because HSMS cannot be used with an SM pubset which is outside HSMS control.
The archive definition is still located in the host environment. It must be deleted before the SM pubset is declared for HSMS. The diagram below illustrates this.

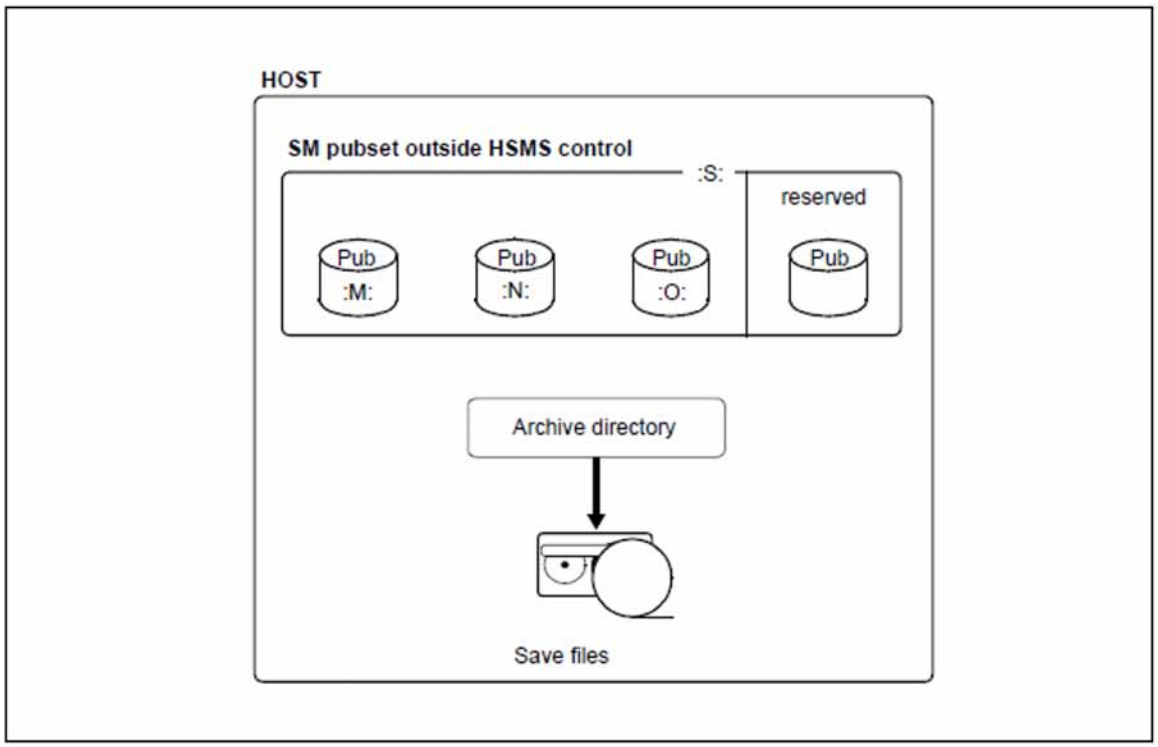


Figure 22: SM pubset outside HSMS control

A pubset must be empty so that it can be declared as the S1 volume set of an SM pubset. Save files located on disks cannot therefore be used (S1 level). Before an existing default migration archive can be used, the HSMS administrator must migrate the save files from S1 to S2 before the SM pubset is created. This is done with the following statement:

```
//MIGRATE-FILES FROM-STORAGE=*S1-STORAGE-LEVEL(..., -
   TO-STORAGE=*S2-STORAGE-LEVEL)
```

For backup archives, the HSMS statement COPY-SAVE-FILE must be used before the conversion.

The SM pubset is placed under HSMS control with the following statement:

```
//CREATE-SM-PUBSET-PARAMETERS SM-PUBSET-ID=S,SYSMIGRATE=*UNDEFINED
```

Existing archive directories can be specified with the HSMS statement MODIFY-SM-PUBSET-PARAMETERS so that they can be used within the converted SM pubset. For this, the following conditions must be met:

- The archive directory must be located within the SM pubset. It may only refer to files and job variables located on the SM pubset.
- The save files must be located in the SM pubset on the S2 level. All volumes (magnetic tape cartridges) must be in the pool of the archive directory.

  The migration to S1 is only possible after the conversion.
- The archive directory must not be used in another archive of another environment.

The diagram below shows how an existing archive is used after conversion of the SM pubset.

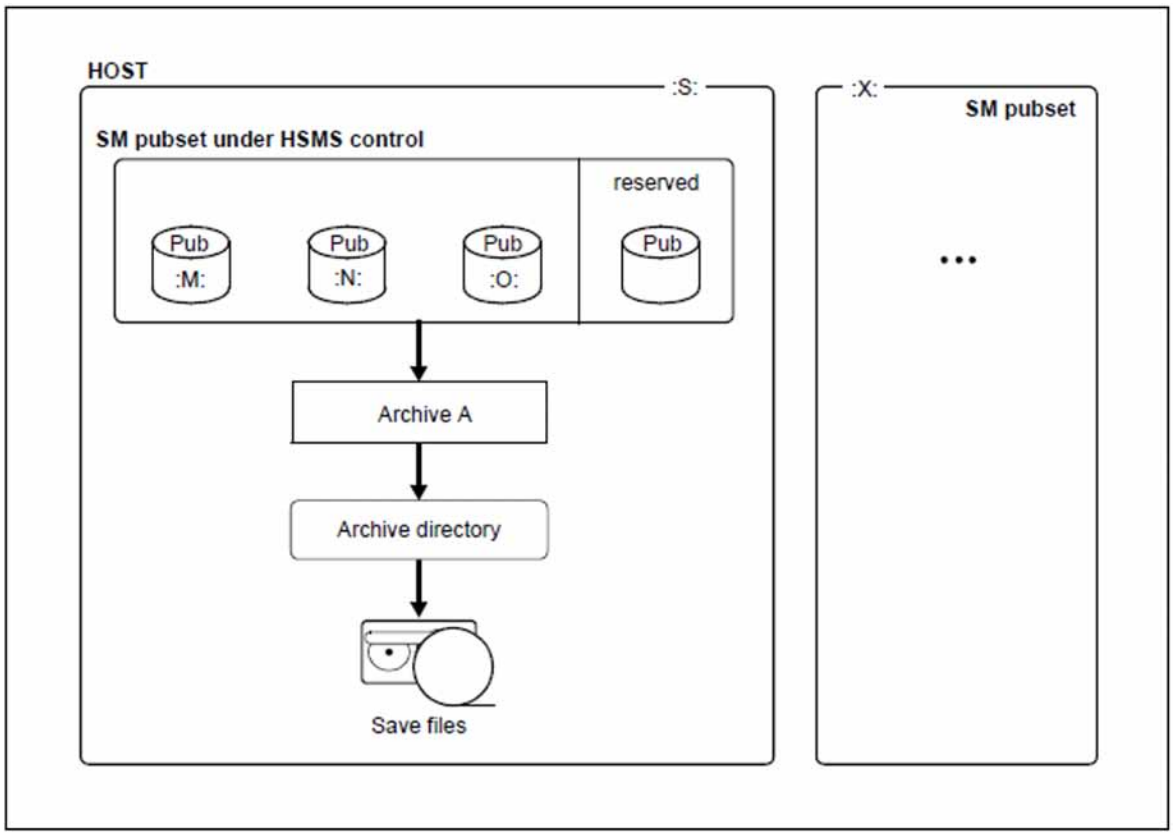


Figure 23: SM pubset under HSMS control

Example

```
/START-DIRCONV ...
//CREATE-ARCHIVE ARCHIVE-NAME=A,DIRECTORY-NAME=D(NEW-DIR=*NO), -
//               ENVIRONMENT=*SYSTEM-MANAGED(CAT-ID=*NO)
//MODIFY-SM-PUBSET-PARAMETERS SM-PUBSET-ID=S,SYSMIGRATE=A
```

If archive directory D was previously used by an archive defined in the host's control file, it must first be deleted.

The same operation can be performed for long-term and backup archives.

Converting/integrating archive directories

When an archive directory is used for an SM pubset, HSMS checks whether all of the file and job variable records refer to the catalog ID of that SM pubset only. If not, the archive directory must be converted by renaming the catalog ID in all of the file and job variable records of the archive directory.

In the diagram below, two pubsets are included in one SM pubset. These pubsets can have no system archive or their own system archive (not shareable or shareable with pubsets included in the same SM pubset) for backup and migration.
The HSMS administrator is responsible for ensuring that the archive definitions and the old archive directories are deleted. HSMS automatically updates the MAREN catalog.

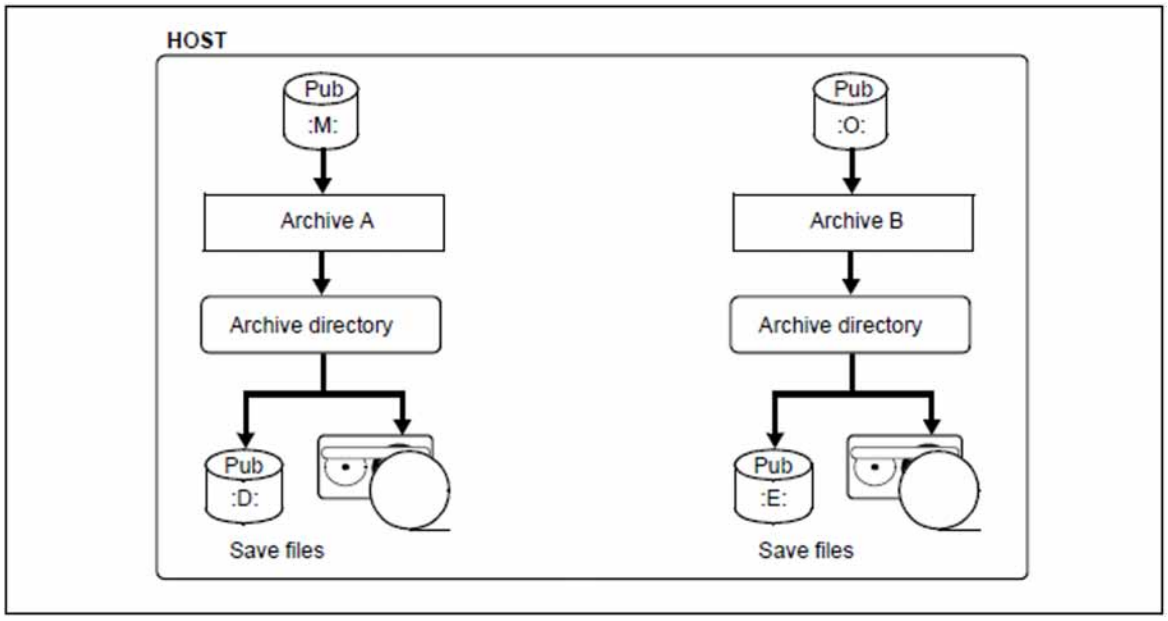


Figure 24: Different archives with separate SF pubsets

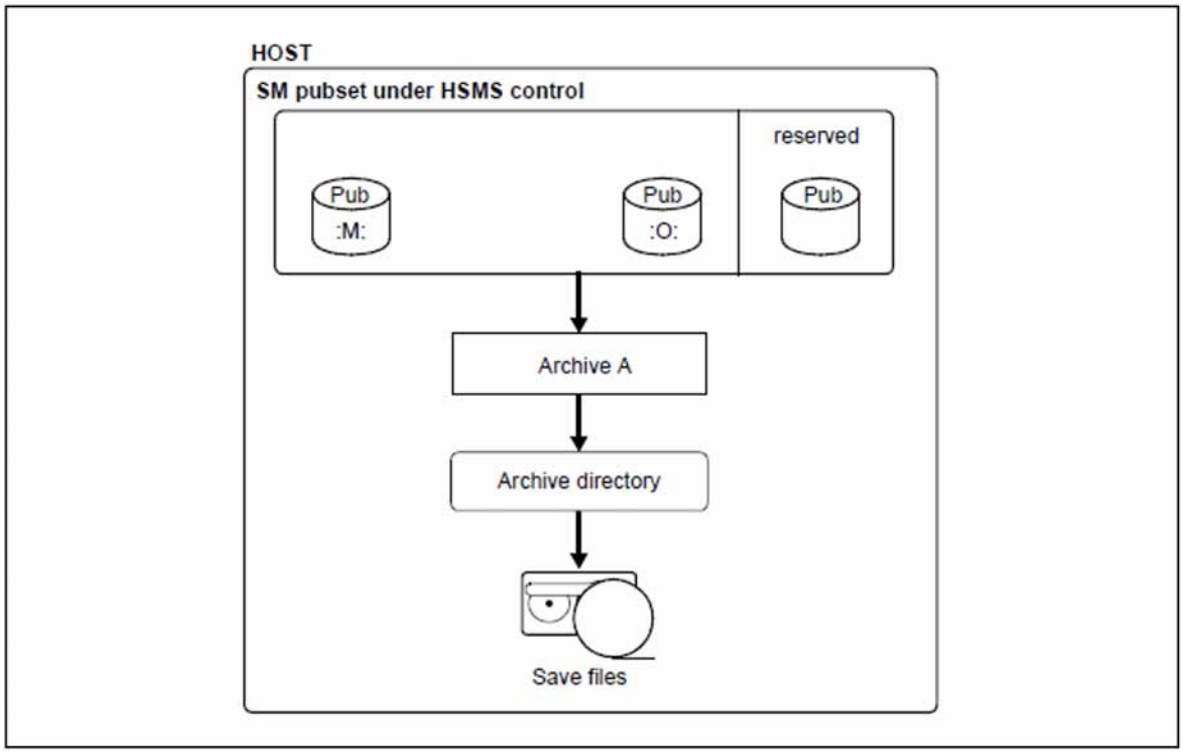


Figure 25: SM pubset under HSMS control

The DIRCONV tool is used to convert one or more archive directories to a new archive directory with a specified catalog ID and integrate them in all file and job variable records. Should a name conflict occur, the process is halted and an error message output. Each problem must then be solved individually.

If the SM pubset was created from numerous SF pubsets with shared archive directories, the statements, RENAME-CATID and UPDATE-VOLUME-CATALOG in DIRCONV must be used. This is only necessary when the new archive directory has a different name to the already existing one.

If the SM pubset was created from numerous SF pubsets, all of which had different directories, then the statements MERGE-DIRECTORIES, RENAME-CATID and UPDATE-VOLUME-CATALOG must be used in DIRCONV.

5.3.3.3 Existing archive directories not restricted to the SM pubset

If existing archive directories are not local for the future SM pubset, a number of operations are necessary to convert the environment of the SM pubset before the pubset itself is converted.

If part of the future SM pubset uses a shared backup or migration archive, that archive can continue to be used for that part of the pubset; for the rest of the SM pubsets, the following procedure must then be followed.

1st step: converting the migration archive

Before creating the SM pubset, all the migrated files concerned must be saved:

```
//SELECT-FILE-NAMES FILE-NAMES=..., -
//   SELECT-FROM=*CATALOG(SUPPORT=*PUBLIC-DISK( -
//   STORAGE-TYPE=*PUBLIC-SPACE),STORAGE-LEVEL=(*S1,*S2)), -
//   OUTPUT=*SELECT-LIST
```

Following this, a new backup archive must be created with a new archive directory:

```
//BACKUP-FILES FILE-NAMES=*SELECTED, -
//   SELECT-FILES=*ALL-FILES, -
//   NEW-FILE-NAMES=*BY-RULE(NEW-CATALOG-ID=...), -
//   SAVE-OPTIONS=*PARAMETERS(SAVE-DATA=*S2-S1-S0), -
//   ARCHIVE-NAME=...
```

Now the SM pubset can be created using DMS resources. The archive directory is used must then be transferred to the SM pubset.

The SM pubset is converted when the above-mentioned archive directory is used for the system backup archive. The migration archive is defined with a new archive directory. The new migration archive is updated with the HSMS statement REPAIR-CATALOG-BY-RESTORE.

2nd step: copying the save files to the SM pubset environment

After defining the SM pubset environment (with the new archive definitions), the user can copy the save files from the SF environment to the SM pubset environment using the HSMS statement COPY-SAVE-FILE.

For backup and long-term archives it is not always necessary to copy all the save files from the host to the SM pubset. The user can still execute a restore from the "old" archive which is defined on the host to the SM pubset, in which case the catalog ID is renamed.

5.3.3.4 Converting an SF pubset to a volume set of an SM pubset

Before an SF pubset can be integrated as a volume set in an SM pubset, it must first be prepared appropriately (see also the “System-Managed Storage” manual [18]).
If the SM pubset concerned is under HSMS control, the SF pubset to be integrated must first be removed from HSMS control. To do this, all operands in the HSMS statement MODIFY-PUBSET-PARAMETERS must be set to the default values.

HSMS does not provide for a pubset to belong to two environments. It is therefore advisable to save the central control file before the SF environment is migrated to the SM environment. This means that at a later date it is possible to return to the SF environment and recover the old SF metadata with the saved control file.

5.3.4 Modifying the SM pubset parameters

When an SM pubset has been taken under HSMS control, the environment can be modified with the various SM pubset-specific parameters using the HSMS statement MODIFY-SM-PUBSET-PARAMETERS.

5.3.5 Displaying the SM pubset parameters

The HSMS parameters that are specific to an SM pubset can be displayed with the HSMS statement SHOW-SM-PUBSET-PARAMETERS.

5.3.6 Importing or exporting an SM pubset

Importing and exporting pubsets is much more complicated for SM pubsets than for SF pubsets because SM pubsets contain specific metadata.

If the IMPORT-PUBSET command is specified for an SM pubset, the IMPORT task calls the HSMS. HSMS checks the names of the SM pubset's control and request file, reads the control file in storage tables and reactivates the requests of the request file. The general catalog facility is opened and the pubset's saturation level is checked. A check is run to remove all SF definitions of the pubset from the HSMS tables. SF definitions can occur if the SM pubset was converted under the same catalog ID or if the SM pubset is imported to another host. (Removal of the SF definitions is not final because this is not done in the control file but in the common memory pool.)

When the startup processing is complete, the SM pubset is available to HSMS. The IMPORT task waits for up to about 2 minutes for completion of the startup processing; then it hands back control.

If an error occurs during the HSMS startup processing, it is aborted even though importing of the pubset is continued. This results in an SM pubset which the DMS can access but which is not "HSMS-started". HSMS recognizes any such situation and automatically triggers a new startup as soon as the next action is carried out in the environment. For example, when the SHOW-SM-PUBSET-PARAMETERS statement is issued, HSMS attempts to restart the environment. This gives the HSMS administrator the opportunity of remedying a startup processing problem without having to fully export/import the SM pubset. The task that requests the environment start waits again for up to about 2 minutes for completion of the startup processing. The HSMS statement requesting the startup fails if an error or a timeout occurs (although in this case the startup is continued asynchronously).

When the subsystem is created, this HSMS import processing is also automatically called by the HSMS main task for each currently accessible SM pubset under HSMS control.

With the EXPORT-PUBSET command for an SM pubset the entire HSMS stop processing is implicitly carried out for an SM environment. The stop processing, as its name suggests, cancels all requests with status ACCEPTED. In addition, it waits until all requests that have been started are completed. A message is output to the operator terminal for each HSMS server task currently working in the environment. When the synchronization point is reached, the EXPORT task deletes the archive definitions of the SM environment from the memory and hands back control.

Requests with status ACCEPTED that were cancelled by the export processing, are not deleted. They remain in the request file and are reactivated the next time the SM pubset is imported.
This does not apply to cancelled Concurrent Copy requests which do not use SHC-OSD mirroring functions and whose session is deinitialized by the EXPORT task. Such Concurrent Copy requests are given the CANCELLED status the next time the SM pubset is imported because they cannot be restarted. No processing occurs and no report is available. If a request like this is prematurely deinitialized by an export, the EXPORT task sends a message to the operator terminal.

5.3.7 Adding/removing a volume set to/from an SM pubset

If a volume set is to be added to an SM pubset, the corresponding user task calls the HSMS interface DHSEMPV. DHSEMPV checks whether the volume set is declared as an SF pubset for HSMS:

- If the volume set is declared as an SF pubset for HSMS, it cannot be added to an SM pubset (see also section "Converting an SF pubset to a volume set of an SM pubset").
- If the volume set is not declared as an SF pubset for HSMS, it can be added to an SM pubset. The saturation level of the new volume set is checked asynchronously.

If a volume set is to be removed from an SM pubset, again the corresponding user task calls the HSMS interface DHSEMPV. DHSEMPV which checks whether the volume set to be removed belongs to the S1 level of the SM pubset. The following cases are possible:

- The volume set to be removed is explicitly set as S1 level. In this case HSMS implicitly executes a MODIFY-SM-PUBSET-PARAMETERS with S1-VOLUME-SET= *UNDEFINED. This means that an S1 level is no longer defined for the SM pubset.
- The volume set to be removed is under HSMS control, and because of the *ALL-HSMS-CONTROLLED setting belongs to the extended S1 level. In this case HSMS checks whether further volume sets under HSMS control are still available for the S1 level. Only if no further volume set is available does HSMS explicitly execute a MODIFY-SM-PUBSET-PARAMETERS with S1-VOLUME-SET= *UNDEFINED. This means that an S1 level is no longer defined for the SM pubset.

5.4 Working with shared pubsets

HSMS statements can be used to process files and job variables residing on a shared pubset.

If no backup server is specified and the shared pubset is imported on the slave host, it is necessary to differentiate between the following two modes:

- “Master mode”
  The HSMS statements are processed on the master sharer. This is the normal mode.
- “Local mode”
  The HSMS statements are processed on the sharer where they were entered.

If a backup server is specified, the HSMS statements are processed on this server. In this case the backup server can be either the master or slave host of the shared pubset (see section "Backup server").

Backup requests created using the “Concurrent Copy” function form an exception. Such requests are handled in the usual slave/master environment. Any backup server settings are ignored. For details, see section "Backup using the function "Concurrent Copy".

If in your computer center several processors on which various HSMS versions are used share pubsets, you should also read section "Compatibility of HSMS definitions in different HSMS environments".

In the SF environments, HSMS save parallel runs on different systems creating save files on shared S1 levels or shared SF pubsets are serialized using the ARCHIVE.ASN, see section "Work files". In the SM-environment, the serialization is guaranteed by the local request file of the SM pubset and the ARCHIVE checkpoint file.

5.4.1 Working with shared SF pubsets

- Master mode
- Local mode

5.4.1.1 Master mode

- **Interpretation of wildcards**

  Objects residing on more than one pubset can be specified in HSMS statements by means of specifications such as *ALL, *OWN or using wildcards combined with the pubset ID. Entries of this kind are always interpreted as applying to all imported pubsets. If the imported pubsets include shared pubsets imported in slave mode, however, the latter are ignored when the wildcards are interpreted. The system displays a message to this effect.

  Examples

  - FILE-NAMES = *ALL / *OWN / wildcards via catalog ID
    JV-NAMES = *ALL / *OWN / wildcards via catalog ID
    ADD-FILE-NAMES = *ALL / *OWN / wildcards via catalog ID

    in the HSMS statements

    ARCHIVE-FILES, BACKUP-FILES, EXPORT-FILES, MIGRATE-FILES, RECALL-MIGRATED-FILES, SELECT-FILE-NAMES and UPDATE-EXPORT-SAVE-FILE.
  - Specification of PUBSET-ID = *ALL in the HSMS statement SHOW-PUBSET-USAGE

- **Processing HSMS requests**

  When reference is made in HSMS statements to shared pubsets that are imported in slave mode, the statements are processed as described in the overview table in section "Overview".

- **Recommendations concerning HSMS support for a shared pubset**
  - For each shared pubset, the HSMS administrator should create a separate system archive for backup, version backup, archival, and migration.
  - The backup system archive of a shared pubset should be created with the same operands on all the sharers of the pubset, i.e. with the same name, the same archive directory, the same operands for tape or disk processing, etc.
    The same applies to the system archives for archival, version backup and migration.
  - The backup system archive of a shared pubset should be assigned the same operands for TAPE-CONTROL (i.e. the same tape sessions, etc.) on all the sharers of the pubset.
    The same applies to the system archives for archival, version backup and migration.
  - In shared pubsets, the archive directory of the system archive for backup, version backup, archival and migration should reside on the pubset.
  - The S1 pubset assigned to a shared S0 pubset must be the same for all sharers of the S0 pubset and be shareable.
  - The tape pool assigned to a shared pubset’s system archive for backup, version backup, archival and migration must be accessible at least for the master of the pubset (via MAREN).
  - For HSMS processing, the master of a shared pubset must have access to a sufficient number of tape devices.
  - When using a BACKUP server (in HSMS-V10.0 or higher with the SAVE-FILE-PROCESSING=*HSMS-V10-COMPATIBLE parameter) that does not match the master, the BACKUP server must have access to a sufficient number of tape devices. The master only needs a few tape devices (for //RESTORE-FILES, //RECALL-MIGRATED-FILES, ...)

- **Restrictions and drawbacks**
  - An archive’s default save file can only be continued as long as the master is not changed between two runs. Since the default save file is part of the archive definition and the archive definition is local for each host. This restriction does not apply to shared SM pubsets in which the archive definition is local to each SM pubset.
  - Private disk files cataloged on a shared pubset can be saved in the backup system archive only if the files are assigned to a shared private disk which the master can access. Private disk files that do not meet this requirement must be saved in a particular storage archive, which the sharers concerned can process.
  - The HSMS statements ARCHIVE-FILES, BACKUP-FILES, BACKUP-FILE-VERSIONS and MIGRATE-FILES need not be restarted after a change of master.
  - BACKUP-FILES and BACKUP-FILE-VERSIONS statements relating to files on a shared pubset cannot be issued to a slave if any of the files are being processed in a Concurrent Copy session.
  - If a large number of shared pubsets have the same master/backup server, HSMS processing may overload this host.
  - If a master/backup server crashes during processing of an HSMS statement or during an implicit recall triggered by a /SECURE-RESOURCE-ALLOCATION for migrated files, the HSMS statement or /SECURE-RESOURCE-ALLOCATION command must be issued again.
    Only an implicit recall caused by an OPEN will be repeated automatically on the backup master following a change of master.
  - If a higher HSMS version is running on the master/backup server than on the other sharers, certain messages that are written to the output file may be undefined. In this case you can only read the message number and the inserts in the output file. You can use the /HELP-MSG command to view the explanatory and help texts on the master or on any sharer which is running an HSMS version in which this message is defined.
- **Advantages of master mode**

  Master mode offers better performance than local mode since little communication is required between the slave sharer and the master.

5.4.1.2 Local mode

- **Interpretation of wildcards**

  Shared pubsets are handled like exclusively imported pubsets, irrespective of the master/slave mode.

- **Processing HSMS requests**

  All HSMS statements referencing shared pubsets are processed on the host where the HSMS statements were entered.

- **Recommendations concerning HSMS support for a shared pubset**
  - For each shared pubset, the HSMS administrator should create a separate system archive for backup, version backup, archival, and migration.
  - The backup system archive of a shared pubset should be created with the same name and the same archive directory on all the sharers of the pubset. The other archive operands can differ from each other.
    The same applies to the system archives for archival, version backup and migration.
  - The backup system archive of this pubset can be assigned either the same or different operands for TAPE-CONTROL on all the sharers of a shared pubset. Please bear in mind one exception, however: the tape session intervals of the pubset sharers may not overlap.
    The same applies to the system archives for archival, version backup and migration, as well as to private backup and archival archives whose directories are shared by two or more pubset sharers.
  - In shared pubsets, the archive directory of the system archive for backup, version backup, archival and migration should reside on the pubset.
  - The S1 pubset assigned to a shared S0 pubset must be the same for all sharers of the S0 pubset and be shareable.
  - The tape pool assigned to a shared pubset’s system archive for backup, version backup, archival and migration must be accessible for all the sharers of the pubset (via MAREN).
  - For HSMS processing, all the sharers of a shared pubset must have access to a sufficient number of tape devices.
- **Restrictions and drawbacks**
  - If tape sessions overlap, it must be ensured that different sharers do not work with the same archive directory at the same time.
  - Processing the metadata of the pubset requires extensive communication between the slave and the master.
- **Advantages of local mode**
    - Local mode is not subject to the restrictions and drawbacks of master mode.
    - If a master crashes while processing an HSMS statement, processing of the statement is resumed automatically following a change of master.

5.4.2 Working with shared SM pubsets

When working with shared SM pubsets – as opposed to shared SF pubsets – HSMS itself guarantees the coherence of the migration, backup and version backup archive definitions on all sharers, i.e. the administrator no longer has to worry about it.

Request management is simpler for shared SM pubsets than for shared SF pubsets because only one request file – namely the local request file of the SM pubset – is involved. Requests can be displayed, restarted and deleted by any sharer. When requests are restarted, the new master sharer is dynamically separated for each restart. The master sharer does not have to be called when requests are deleted.

Furthermore, in an SM environment, only one catalog ID is involved so there are no wildcards to be interpreted.

5.4.3 Backup server

A backup server can be defined explicitly for processing HSMS statements (in HSMS V10.0 or higher with the SAVE-FILE-PROCESSING=*HSMS-V10-COMPATIBLE parameter). Processing takes place on the backup server regardless of whether the server is the master or slave host of the shared pubset.

Backup and version backup requests created using the “Concurrent Copy” function form an exception. Such requests are handled in the usual slave/master environment. Any backup server settings are ignored. For details, see section "Backup using the function "Concurrent Copy"".

The archive attribute BACKUP-SERVER-USAGE determines where HSMS statements which concern this archive are processed:

- When *NO is specified, the statements are in general processed on the pubset master, as was the case in HSMS < V10.0 (for details see section "Master mode" and section "Overview".
- When *STD is specified, the statements are processed in accordance with the current setting in the HSMS parameter BACKUP-SERVER:
  - When *NONE is specified, the statements are processed on the pubset master as was the case in HSMS < V10.0.
    When a backup server is used, *LOCALHOST must be entered on this server.
  - When a backup server is entered, the statements are processed on this server.
  - When *LOCALHOST or the name of the local system is entered, the statements are processed on the local server. This setting is obligatory on a system which is to be used as a backup server.

Effect of the various settings

Archive definition BACKUP-SERVER-USAGE=	**Value of the BACKUP-SERVER operand on the local system:**		
	***NONE**	**<alphanum-name 1..8> <bcam-name>**	***LOCALHOST**
*STD	Processing without using the backup server functionality[1]	Processing on the backup server[2 3]	Processing on the local system The current system is the backup server
*NO	Processing without using the backup server functionality[1]		

1 If the local system is master sharer of a shared pubset, the request is executed locally. If the local system is the slave sharer of a shared pubset, the request is sent to the master.

2 On the backup server a check is made to see whether BACKUP-SERVER=*LOCALHOST is set. If this is not the case, the request is rejected and message HSM0329 is output to the log file.

3 Specifying the name of the local system has the same effect as *LOCALHOST.

Regardless of the BACKUP-SERVER setting, the following requests are always executed locally:

- Requests under CCOPY
- Data transfer statements
  IMPORT-FILES
  EXPORT-FILES
  COPY-EXPORT-SAVE-FILE
  UPDATE-EXPORT-SAVE-FILE
- Node statements
- COPY-SAVE-FILE
- MOVE-SAVE-FILE

Error handling

When a statement for processing is sent to a backup server, the HSMS parameter BACKUP-SERVER is checked to determine whether this system is currently a backup system (i.e. whether *LOCALHOST or the system’s host name is entered). If the system is currently not a backup server, the request is rejected and is assigned the INTERRUPTED status with the BACK-SERV-REPLIED substatus.

In the HSMS report, the message HSM0329 also displays the cause of the error.

5.4.4 Overview

The table below shows where statements are processed when shared pubsets are concerned depending on whether or not a backup server is specified.

Statement	Processing	
	No backup server specified	**Backup server specified**
ARCHIVE-FILES	(1)	(2)
ARCHIVE-NODE-FILES	local (node)	local (node)
BACKUP-FILES	(1)	(2)
BACKUP-FILE-VERSIONS	(1)	(2)
BACKUP-NODE-FILES	local (node)	local (node)
CHECK-CATALOGED-FILES	local + SM-dynamic	local + SM-dynamic
COPY-EXPORT-SAVE-FILE	local (data transfer)	local (data transfer)
COPY-NODE-SAVE-FILE	local (node)	local (node)
COPY-SAVE-FILE	local (how many masters?)	local
CREATE-ARCHIVE	local + SM-dynamic	local
CREATE-MANAGEMENT-CLASS	local + SM-dynamic	local + SM-dynamic
CREATE-SM-PUBSET-PARAMETERS	local + SM-dynamic	local + SM-dynamic
DELETE-ARCHIVE	local + SM-dynamic	local + SM-dynamic
DELETE-MANAGEMENT-CLASS	local + SM-dynamic	local + SM-dynamic
DELETE-REQUESTS	(3)	(3)
EXPORT-FILES	local (data transfer)	local (data transfer)
IMPORT-FILES	local (data transfer)	local (data transfer)
LIST-VOLUMES	local	local
MIGRATE-FILES	(1)	(2)
MODIFY-ARCHIVE	local + SM-dynamic	local + SM-dynamic
MODIFY-ARCHIVE-ATTRIBUTES	local + SM-dynamic	local + SM-dynamic
MODIFY-HSMS-PARAMETERS	local	local

MODIFY-PUBSET-PARAMETERS	local	local
MODIFY-SM-PUBSET-PARAMETERS	local + SM-dynamic	local + SM-dynamic
MODIFY-TAPE-CONTROL	local + SM-dynamic	local + SM-dynamic
MOVE-SAVE-FILES	local	local
RECALL-MIGRATED-FILES	(1)	(2)
RECALL implicit (by /SECURE-RESOURCE-ALLOCATION or OPEN)	(1)	(2)
RECOVER-REQUESTS	local	local
REORGANIZE-VERSION-BACKUP	local (how many masters?)	(2)
REPAIR-CATALOG-BY-EXCHANGE	local	local
REPAIR-CATALOG-BY-RESTORE	(1)	(2)
REPLACE-SAVE-FILE-BY-EXCHANGE	local	local
REPLACE-SAVE-FILE-BY-RESTORE	local (how many masters?)	local
RESTART-REQUESTS	(3)	(3)
RESTORE-FILES	(1) and (4)	(2) and (4)
RESTORE-LIBRARY-ELEMENTS	(1) and (4)	(2) and (4)
RESTORE-NODE-FILES	local (node)	local (node)
SELECT-FILE-NAMES	local	local
SELECT-JV-NAMES	local	local
SELECT-NODE-FILES	local (node)	local (node)
SHOW-ARCHIVE	local	local
SHOW-ARCHIVE-ATTRIBUTES	local	local
SHOW-HSMS-PARAMETERS	local	local
SHOW-MANAGEMENT-CLASS	local	local
SHOW-NODE-PARAMETERS	local (node)	local (node)
SHOW-PUBSET-PARAMETERS	local	local
SHOW-PUBSET-USAGE	local	local

SHOW-REQUESTS	local	local
SHOW-SM-PUBSET-PARAMETERS	local	local
SHOW-TAPE-CONTROL	local	local
UPDATE-EXPORT-SAVE-FILE	local (data transfer)	local (data transfer)

Key

(1)
The rules listed below are observed:

- Shared pubsets imported in master mode are treated just like local pubsets.
- If local pubsets are specified together with shared pubsets, the local pubsets are processed locally and the shared pubsets that were imported in slave mode are rejected with a warning.
- Statements that deal with shared pubsets imported in slave mode are processed on the master, if the master is unequivocal; otherwise the statement is rejected.

(2)
The rules listed below are observed:

- Requests are processed on the backup server when the archive definition BACKUP-SERVER-USAGE has the value *STD and at the same time the global HSMS parameter BACKUP-SERVER has a value other than *NONE.
  1. If a host name is specified for BACKUP-SERVER, a check is made on the backup server to see whether the *LOCALHOST value or the name of the local system is set for BACKUP-SERVER. If this is not the case, the request is rejected and message HSM0329 is output to the log file.
  2. If BACKUP-SERVER has the value *LOCALHOST, the request is executed locally as the local system is the backup server.
  3. If the host name of the local system is specified for BACKUP-SERVER, the requests are handled as if *LOCALHOST had been specified.
- If local pubsets are specified together with shared pubsets, the local pubsets are ignored as the backup server functionality is only supported for shared pubsets.

(3)
Request management
See "Request management" for information on deleting/restarting requests.

(4)
Restore requests are processed locally when the catalog ID is renamed.

local (how many masters?)
Requests that very probably have more than one master are always processed locally. This applies in particular when a statement processes the entire save file.

local (data transfer)
Statements for data transfer are always processed locally.

local (node)
Statements for nodes are always processed locally.

local + SM-dynamic
Statements are processed locally. However, in an SM environment, the results of the statement processing can be viewed on other sharers of the SM pubset concerned.

5.5 Working with SM pubsets and different HSMS versions

SM pubsets can be exported and imported from system to system. Different HSMS versions could be in use on these systems. To ensure that the various HSMS control and request files are compatible, new compatibility features have been defined for the HSMS definitions on the SM pubsets: each archive or request within an SM pubset has a so-called version attribute. This attribute describes the HSMS version required for processing the archive or request. The following general rules apply:

- When a definition is created for the first time, HSMS determines the oldest SMS version that can be processed by the definition.
- Before a definition is used or modified, HSMS checks whether the requested HSMS version is sufficient for the definition.
- When a definition is modified, HSMS redetermines the oldest HSMS version that can be processed by the definition.

If a definition is displayed with a SHOW statement, HSMS always tries to show as many correct operands as possible. If an operand is outside the permitted range, its value is represented by a question mark.

If an HSMS statement forbids an operand which affects an archive, the version attribute of the affected request is not decreased (for example, if an archive is assigned a shadow archive and the request was issued with SHADOW-COPY=*INHIBITED).

In the case of shared SM pubsets the slave sharers usually send their requests to the master sharer for processing. When a slave sharer creates a request, it determines the functional HSMS version of the request by consulting the list of conditions described above.
When the master sharer receives the request, it first uses the functional HSMS version to check whether or not it can process the request. If the functional HSMS version of the request is higher than the HSMS version on the master sharer, the master sharer rejects the request.

This functional version mechanism enables sharers on which higher HSMS versions are running to transfer requests which do not contain any new HSMS functions to sharers with lower HSMS versions.

5.6 Management of the file attributes in an environment with SM pubsets

The file attributes can be divided into two groups:

- Protection attributes
  - passwords
  - standard access control with USER-ACCESS and ACCESS
  - GUARDS, BASIC-ACL

  These protection attributes are not specific to an SF or SM pubset. They are described in the “ARCHIVE” manual [2].
- Assignment attributes
  - STORAGE-CLASS
  - AVAILABILITY
  - WORKFILE
  - IOPERF
  - IOUSAGE
  - BLOCK-CONTROL-INFO

  These attributes have a special use in an SM pubset. The table below shows how the attributes are processed in the case of a RESTORE / IMPORT. If a file that was backed up by an SM pubset is restored to an SF pubset, these attributes are reset.

Attribute	Recover	Restore without changing environment	Restore and change environment
STORAGE-CLASS	from disk	from tape	reset to *NONE
AVAILABILITY	from disk	from tape	from tape
WORKFILE	from disk	from tape	from tape
IOPERF	from disk	from tape	from tape
IOUSAGE	from disk	from tape	from tape

Unlike SF pubsets where the complete pubset has either key format or nonkey format, an SM pubset can contain volume sets with key disks and volume sets with nonkey disks. That is why, in an environment with SM pubsets, there is no automatic conversion from key format to nonkey format. A file is restored/recalled with the same BLKCTRL and PAMKEY values as for the backup.

A file is restored to the best possible volume set in terms of its assignment attributes and user standards, but with the following exceptions.

- A file with the attribute S0-MIGRATION-FORBIDDEN is always restored/recalled to the volume set where it was originally located.
- A volume set was specified in the RESTORE/RECALL statement (this does not apply to a file with the attribute S0-MIGRATION-FORBIDDEN).

If a file cannot be assigned because the specified volume set does not exist, it is restored/ recalled to the best possible volume set (this also applies to a file with the attribute S0-MIGRATION-FORBIDDEN).

5.7 Managing the workstation

This section addresses the HSMS administrator. It describes how the connection between HSMS in BS2000 and a workstation is established and managed if the latter is mounted via the BS2000-UFS (POSIX) and its files are to be saved or archived. The BACKUP-NODE-FILES or ARCHIVE-NODE-FILES statements are used for this purpose.

For UNIX workstations and PCs, the connection can only be established via NFS.

For any details of the actions to be performed which are not covered in the following description please see the “NFS” manual [11].

5.7.1 Actions to be performed at a workstation in the event of access via NFS

If access to a UNIX workstation is to carried out using NFS, the HSMS administrator must first perform certain actions at the workstation. The following description refers to a UNIX workstation.

All file systems that are to be processed by HSMS must be configured as follows so that they are shareable:

- The file systems must be entered in the file /etc/dfs/sharetab.
- The files systems must be made shareable by means of the UNIX command *share*.

Note, however, that, due to the utilization of NFS, only those file systems can be defined as shareable which do not yet include any shareable files.

The file systems on the UNIX workstation must be assigned the “root” privilege for the BS2000 system on which HSMS is running.

The administrator must specify the list of shareable file systems which he/she wishes to save in the relevant save command for node files. A list of the shareable file systems can be determined using the UNIX command *dfshares* .

5.7.2 Actions to be performed in BS2000 (server)

To ensure that workstations can be processed, the HSMS administrator must perform the following actions in BS2000:

Creating archives

In order to enable files of the local BS2000-UFS to be processed by HSMS, all users may create private archives for their own files. The HSMS administrator can create standard system archives for backup and for archival. These can be accessed via the symbolic names SYSNODEBACKUP (public backup archive) and SYSNODEARCHIVE (public long-term archive).

Defining the path and mounting file systems

If HSMS is to be used as backup server for node files, the system administrator has to make these node files available:

- POSIX must be available for processing by NFS. The POSIX file system is the local BS2000-UFS. The files of the POSIX file system are automatically available.
- In order to be able to save data from a workstation or restore data to a workstation using NFS, each remote file system that is to be processed by HSMS must be mounted on the appropriate node “HSMS/<node-id>” of the local BS2000-UFS using NFS. If the file system is a subordinate file system within a workstation, it must be mounted at a lower directory level.
- Only root should have all three access rights (read, write and execute) for the “HSMS” directory. All other protection attributes should be set to the highest protection level. Otherwise other UFS users have read access to the files of a workstation.
- If, when NFS is used, a file system contains file systems of a lower hierarchical level, the BS2000 system administrator has to mount each of these explicitly at the appropriate node of the local BS2000-UFS, following the hierarchical structure existing at the workstation. This is illustrated by the diagram below:

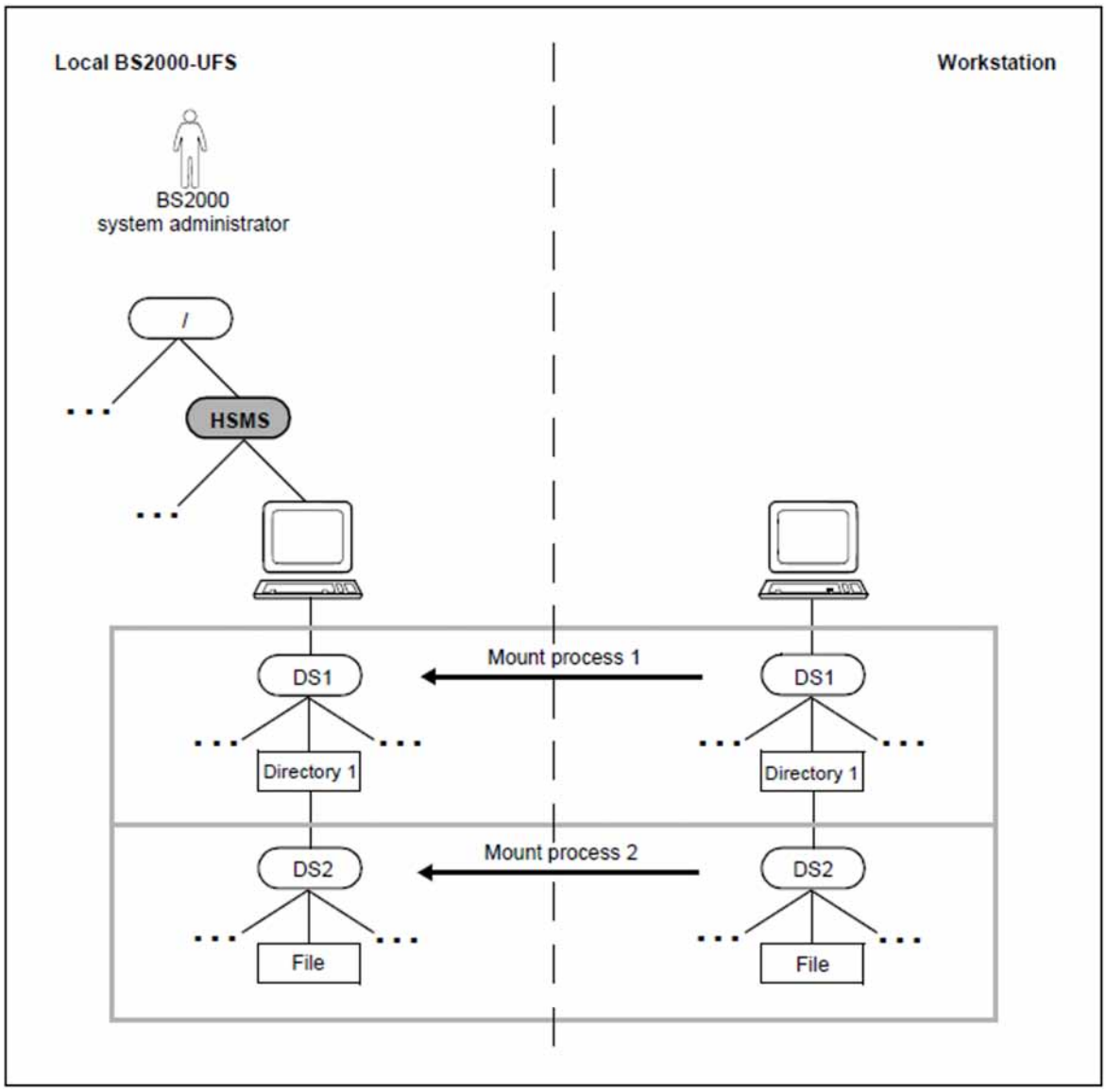


Figure 26: Correct mounting of file systems

We recommend that you write a standard procedure to facilitate the mounting of file systems on the local BS2000-UFS following their hierarchical order at the workstation.

Example

NFS command *share* <WS-address>

Possible specifications:

RESOURCE	SERVER	ACCESS	TRANSPORT
<WS-address>:/	<WS-address>	– – –	– – –
<WS-address>:/opt	<WS-address>	– – –	– – –
<WS-address>:/usr	<WS-address>	– – –	– – –
<WS-address>:/var	<WS-address>	– – –	– – –
<WS-address>:/home	<WS-address>	– – –	– – –

The mount operation must start with the highest level of the tree structure.
In the above example, the root directory “ / ” has to be mounted under a unique mounting point first.

It is not necessary to specify all file systems of a file tree, but each entry must refer to a different file system.
The *share* command does not permit multiple declarations of any one file system.
BS2000 cannot verify whether all file systems declared actually exist.

The file systems of the above example would have to be mounted in the following order:

mount -F nfs <WS-address>:/	/HSMS/<WS-subdir>/
mount -F nfs <WS-address>:/opt	/HSMS/<WS-subdir>/opt
mount -F nfs <WS-address>:/usr	/HSMS/<WS-subdir>/usr
mount -F nfs <WS-address>:/var	/HSMS/<WS-subdir>/var
mount -F nfs <WS-address>:/home	/HSMS/<WS-subdir>/home

5.7.3 Relationship between node S0s and archives

HSMS archives can be assigned to one or more or all node S0s. The following criteria may be used to decide whether to implement central or decentralized archive management:

- Concurrent execution of requests:
  Multiple archives permit concurrent execution of multiple requests.
- Number of files to be processed:
  A large number of files managed in a single archive reduces performance.

5.7.3.1 Central archive and central node S0 declaration

1. In the following example, a central backup archive and a central long-term archive are created.

```
/START-HSMS
//CREATE-ARCHIVE ARCHIVE-NAME=CENTRAL.BAC, -
//  ALLOWED-USAGE=*NODEBACKUP,..., -
//  DIRECTORY-NAME=$SYSHSMS.CENTRAL.BAC.DIR, -
//  USER-ACCESS=*ALL-USERS(ACCESS=*WRITE)
//CREATE-ARCHIVE ARCHIVE-NAME=CENTRAL.ARC, -
//  ALLOWED-USAGE=*NODEARCHIVAL,..., -
//  DIRECTORY-NAME=$SYSHSMS.CENTRAL.ARC.DIR, -
//  USER-ACCESS=*ALL-USERS(ACCESS=*WRITE)
```

   Subsequently, the archives can be used for all node S0s defined.

2. If a single archive has been created for all node S0s, all passive nodes can be backed up with one request:

```
//BACKUP-NODE-FILES PATH-NAMES=*PATH-NAME( -
//                  NODE-ID=*,PATH=*), ...
```

5.7.3.2 Archive for a single node S0 and node S0 declaration

1. The following example shows a procedure that creates a backup archive and a long-term archive for a specific node S0. Subsequently, the node S0 is defined for HSMS.

```
/BEGIN-PROCEDURE PARAMETERS=*YES(PROCEDURE-PARAMETERS=(&NODEID), -
/  ESCAPE-CHARACTER=C'&')
/START-HSMS
//CREATE-ARCHIVE ARCHIVE-NAME=BAC.&NODEID, -
//  ALLOWED-USAGE=*NODEBACKUP,..., -
//  DIRECTORY-NAME=$SYSHSMS.BAC.&NODEID..DIR, -
//  USER-ACCESS=*ALL-USERS(ACCESS=*WRITE)
//CREATE-ARCHIVE ARCHIVE-NAME=ARC.&NODEID, -
//  ALLOWED-USAGE=*NODEARCHIVAL,..., -
//  DIRECTORY-NAME=$SYSHSMS.ARC.&NODEID..DIR, -
//  USER-ACCESS=*ALL-USERS(ACCESS=*WRITE)
//END
/END-PROCEDURE
```

2. If an archive is created for each node S0, each node must be backed up using a separate HSMS statement:

```
//BACKUP-NODE-FILES PATH-NAMES=*PATH-NAME( -
//                  NNODE-ID=<node-id>,PATH=*), ...
```

3. File restoration must likewise be performed separately for each node S0.

5.7.3.3 Multiplexing operation

In the multiplexing operating mode, multiple ARCHIVE subtasks can simultaneously sharethe same MTC devices and magnetic tape cartridges in parallel. This ensures enhanced performance and the optimal utilization of tape devices. In addition, the magnetic tape cartridges are optimally filled.

Multiplexing is activated by specifying PARALLEL-RUNS=*MULTIPLEXING(..) in the HSMS statement BACKUP-NODE-FILES. Multiplexing can also be defined at archive level using the HSMS statements CREATE-ARCHIVE and MODIFY-ARCHIVE-ATTRIBUTES.

When restoring node files (//RESTORE-NODE-FILES), you do not need to specify an operand in order to activate multiplexing operation. ARCHIVE automatically calculates the number of required subtasks.

For more detailed information on multiplexing operation, see "Parallel and serial processing in ARCHIVE".

Examples

Below we illustrate the various ways in which users can activate multiplexing operation and how HSMS constructs its multiplexing configuration in response to these:

1st example: Backing up a single path

```
// BACKUP-NODE-FILES PATH-NAMES=*PATH-NAME(
   PATH=/<dir>,NODE-ID=<node-id>)
   OPERATION-CONTROL=*PAR(PARALLEL-RUNS=*MULTIPLEXING(NUMBER-OF-DEVICES=
   <integer 1..16>,MULTIPLEXING-FACTOR=<integer 2..14> oder *AUTOMATIC))
```

Multiplexing operation does not occur in this example since parallel processing is not possible. A multiplexing environment is nevertheless set up (one subtask for a tape drive) since the save file must always have “multiplexed” format. This makes it possible to continue a multiplexed tape later.

2nd example: Restoring one file or multiple files backed up using the same subtask

If only one file is to be backed up, only a single subtask can run. Consequently, it is not necessary to set up a multiplexing environment. Only one subtask is generated and this works directly with the device (as in the case of runs without multiplexing operation).

The same applies if all the files which are to be restored were backed up by the same subtask. Such files cannot be restored in parallel since they were stored sequentially on the tape.

3rd example: Restoring multiple multiplexed files (i.e. files which were backed up by multiple subtasks in parallel)

If multiple files which all belong to the same save version have been backed up by different subtasks, all the files can be restored in parallel (the files are all intermixed on the tape and the tape is not rewound during the restore operation).

ARCHIVE automatically calculates the number of required subtasks and sets up the environment necessary for a parallel restore. If 2 files are to be restored, a maximum of only 2 subtasks is required even if the save version was created with 14 subtasks. ARCHIVE always uses the smallest possible number of subtasks.

To illustrate:

- File F1 is backed up by subtask T1, F2 by T2, F3 by T3 and F4 by T4
  on device G1 in save file SF1, save version SV1.
- File F5 is backed up by T1, F6 by T2
  on device G1 in the save file SF2, save version SV2.

To restore all the files, 4 subtasks are required for save version SV1 and 2 subtasks for SV2.
If PARALLEL-RUNS=1 applies, ARCHIVE generates 4 subtasks (=max(4,2)) and 1 tape drive task.
If PARALLEL-RUNS=2 applies, ARCHIVE generates 8 subtasks and 2 tape drive tasks; the same multiplexing factor is used for each of the devices since ARCHIVE does not know which tape drive will process save file SF1 or SF2. The restoration of F1, F2, F3 and F4 requires 2 subtasks (=max(2,2)). The same rule applies to the calculation of the configuration: the result is PARALLEL-RUNS x (2 subtasks + 1 tape drive).

5.7.4 Examples

This section provides examples dealing with the following subjects:

- The HSMS administrator performs a central full backup of the local BS2000-UFS and several passive workstations
- The HSMS administrator restores a node file to a passive workstation
- A nonprivileged user restores selected file trees to the local BS2000-UFS
- The HSMS administrator archives the file tree of a passive workstation

The HSMS administrator performs a central full backup of the local BS2000-UFS and several passive workstations

Perform preparatory activities:

```
/START-HSMS
//MODIFY-NODE-PARAMETERS -  ———————————————————————————————————  (1)
//   NODE-ID=*BS2000-UFS, -
//   HSMS-CONTROL=*YES(SYSNODEBACKUP=*STD)
//MODIFY-NODE-PARAMETERS -  ———————————————————————————————————  (2)
//   NODE-ID=WORKSTA1, -
//   HSMS-CONTROL=*YES(SYSNODEBACKUP=*STD)
//MODIFY-NODE-PARAMETERS -
//   NODE-ID=WORKSTA2, -
//   HSMS-CONTROL=*YES(SYSNODEBACKUP=*STD)
//MODIFY-NODE-PARAMETERS -
//   NODE-ID=WORKSTA3, -
//   HSMS-CONTROL=*YES(SYSNODEBACKUP=*STD)
//CREATE-ARCHIVE -  ———————————————————————————————————————————  (3)
//   ARCHIVE-NAME=CENTRAL, -
//   ALLOWED-USAGE=*NODEBACKUP, -
//   USER-ACCESS=*ALL-USERS, -
//   DIRECTORY-NAME=NODEBACKUP.CENTRAL.DIR, -
//   RETENTION-PERIOD=150, -
//   S2-DEVICE-TYPE='TAPE-C4', -
//   OPERATION-CONTROL=*PARAMETERS(PARALLEL-RUNS=2)
//MODIFY-ARCHIVE -  ———————————————————————————————————————————  (4)
//   ARCHIVE-NAME=CENTRAL, -
//   VOLUMES=*ADD(VOLUMES=(VOLUM1,VOLUM2,...))
//MODIFY-HSMS-PARAMETERS -  ———————————————————————————————————  (5)
//   VALID-PERIOD=*PERMANENT, -
//   DEFAULT-HSMS-STORAGE=*PARAMETERS(SYSNODEBACKUP=CENTRAL)
//END
```

(1) The local BS2000-UFS is placed under HSMS management. The default system archive SYSNODEBACKUP is assigned to this local UFS.

(2) The workstations WORKSTA1, WORKSTA2 and WORKSTA3 are placed under HSMS management. The default system archive SYSNODEBACKUP is assigned to them. Backup takes place using NFS.

(3) The archive CENTRAL is set up as a public archive for backing up node files with read privilege.
The node files which are backed up cannot be deleted for 150 days.
Magnetic tape cartridges are specified as the target volumes in order to use a robotic archive connected to the system.
Two backup tasks are to execute simultaneously.

Note

In order for nonprivileged users to use the directory file, it must be shareable (USER-ACCESS=*ALL-USERS or *SPECIAL).

(4) Magnetic tape cartridges are added to the pool of free volumes of the CENTRAL archive.

(5) The CENTRAL archive is defined as the global default system archive for backing up node files.

Create a list of the path names which are to be backed up:

Contents of the SAM file "PATH.LIST":

```
 ————————————————————————————————————————————————————————————————————————  (6)
<bs2000-ufs path-name1>
<bs2000-ufs path-name2>
<bs2000-ufs ...>
WORKSTA1:/<path-name2>
WORKSTA1:/...
WORKSTA2:/<path-name1>
WORKSTA2:/<path-name2>
WORKSTA2:/...
WORKSTA3:/<path-name1>
WORKSTA3:/<path-name2>
WORKSTA3:/...
```

(6) Since only a single path name can be specified in the BACKUP-NODE-FILES statement, the HSMS administrator creates the SAM file PATH.LIST, which contains the path names of the node files to be backed up.

Start full backup:

```
/START-HSMS
//BACKUP-NODE-FILES -  ————————————————————————————————————————————————  (7)
//   PATH-NAMES=*FROM-FILE(LIST-FILE-NAME=PATH.LIST), -
//   ARCHIVE-NAME=*SYSNODEBACKUP, -
//   SELECT-FILES=*ALL-FILES(FROM=LATEST-BACKUPS-OR-S0), -  ————————————  (8)
//   OPERATION-CONTROL=*PARAMETERS(REPORT=*FULL,OUTPUT=FULL.BACKUP)
//END
```

(7) The path names of the node files which are to be backed up are read from the PATH.LIST file. A full backup is made to the default system archive *SYSNODEBACKUP.

(8) For performance reasons, the operand FROM=*LATEST-BACKUPS-OR-S0 is specified for the backup. Node files which have been modified since the last backup are backed up from the node S0; unmodified node files are copied from their last backup in the same archive.
The operand FROM=*LATEST-BACKUPS-OR-S0 must not be specified for the very first full backup since it is first necessary to create the backup environment.

The HSMS administrator performs a decentralized full backup of a workstation

The workstation is mounted in POSIX under /HSMS/WORKSTA3 and can be accessed via NFS.

Perform preparatory activities:

```
/START-HSMS
//MODIFY-NODE-PARAMETERS -  ————————————————————————————————————  (1)
//   NODE-ID=WORKSTA3
//CREATE-ARCHIVE -  ————————————————————————————————————————————  (2)
//   ARCHIVE-NAME=WORKSTA3, -
//   ALLOWED-USAGE=*NODEBACKUP, -
//   USER-ACCESS=*ALL-USERS, -
//   DIRECTORY-NAME=NODEBACKUP.WORKSTA3.DIR, -
//   RETENTION-PERIOD=150, -
//   S2-DEVICE-TYPE='TAPE-C4'
//CREATE-ARCHIVE -  ————————————————————————————————————————————  (3)
//   ARCHIVE-NAME=WS3DIR, -
//   ALLOWED-USAGE=*BACKUP(SAVE-FILE-STRUCTURE=*SEVERAL-SVID), -
//   DIRECTORY-NAME=BACKUP.WS3DIR.DIR, -
//   RETENTION-PERIOD=150, -
//   S2-DEVICE-TYPE='TAPE-C4'
//MODIFY-ARCHIVE -  ————————————————————————————————————————————  (4)
//   ARCHIVE-NAME=WORKSTA3, -
//   VOLUMES=*ADD(VOLUMES=(VOLUM1,VOLUM2,...))
//END
```

(1) The workstation WORKSTA3 is assigned.

(2) The archive WORKSTA3 is defined as the backup archive for node files. All users have read privilege.

Note

In order for nonprivileged users to use the directory file, it must be shareable (USER-ACCESS=*ALL-USERS or *SPECIAL).

(3) The archive WS3DIR is defined as the destination for the backup of the directory file NODEBACKUP.WORKSTA3.DIR since a directory file (BS2000 file) cannot be stored in the same save file as the node files (UNIX files).
Each of the archive’s save files may contain multiple save versions.
It is expedient to specify the same retention period for the directory-file archive (WS3DIR) as for the node-file archive (WORKSTA3).

(4) Magnetic tape cartridges are added to the pool of free volumes of the WORKSTA3 archive.

Create a list of the path names which are to be backed up:

Contents of the SAM file "WORKSTA3.PATH.LIST":

```
 ————————————————————————————————————————————————————————————————————————  (5)
WORKSTA3:<path-name1>
WORKSTA3:<path-name2>
WORKSTA3:...
```

(5) The HSMS administrator creates the SAM file WORKSTA3.PATH.LIST, which contains the path names of the node files to be backed up.

Start full backup:

```
/START-HSMS
//BACKUP-NODE-FILES -   ————————————————————————————————————————————————————  (1)
//   PATH-NAMES=*FROM-FILE(LIST-FILE-NAME=WORKSTA3.PATH.LIST), -
//   SELECT-FILES=*ALL-FILES, -
//   ARCHIVE-NAME=WORKSTA3, -
//   OPERATION-CONTROL=*PARAMETERS(REPORT=*FULL,OUTPUT=FULL.BACKUP, -
//                         WAIT-FOR-COMPLETION=*YES)
//BACKUP-FILES -   ————————————————————————————————————————————————————————  (2)
//   FILE-NAMES=NODEBACKUP.WORKSTA3.DIR, -
//   ARCHIVE-NAME=WS3DIR, -
//   TO-STORAGE=*S2-STORAGE-LEVEL(VOLUMES=DIRTAPE), -
//   OPERATION-CONTROL=*PARAMETERS(REPORT=*FULL,OUTPUT=NODEBACKUP.DIR)
//END
```

(1) The path names of the node files which are to be backed up are read from the file WORKSTA3.PATH.LIST. All node files are backed up in the archive WORKSTA3.

(2) The directory file NODEBACKUP.WORKSTA3.DIR is backed up in the archive WS3DIR on storage level S2.

The HSMS administrator restores a node file to a passive workstation

The node file was backed up in a central backup by means of the BACKUP-NODE-FILES statement.

```
/START-HSMS
//RESTORE-NODE-FILES -  ————————————————————————————————————————————————————  (1)
//   PATH-NAMES=*PATH-NAME(PATH=/DIR/FILE, -
//      NODE-ID=WORKSTA2), -
//   SELECTION-BOUNDARY=*SPECIFIED-PATHS, -  ——————————————————————————————————  (2)
//   REPLACE-FILES=*YES, -  ———————————————————————————————————————————————————  (3)
//   SELECT-SAVE-VERSIONS=*BY-ATTRIBUTES -  ———————————————————————————————————  (4)
//      (SAVE-VERSION-DATE=00-08-27(TIME=22:19:15)), -
//   OPERATION-CONTROL=*PARAMETERS(WAIT-FOR-COMPLETION=*YES, -
//      REPORT=*FULL,OUTPUT=FULL.RESTORE)
//END
```

(1) The node file with the path name /DIR/FILE is restored to the passive workstation WORKSTA2.

(2) If the node file is a directory, only the inode will be restored. The files contained in this directory will not be restored.

(3) Any existing node file with the same path name will be overwritten during the restore.

(4) This specification is required only if the node file backup which is to be restored is not the latest one.

A nonprivileged user restores selected file trees to the local BS2000-UFS

The file trees were backed up centrally by means of the BACKUP-NODE-FILES statement.

```
/START-HSMS
//SELECT-NODE-FILES -   ————————————————————————————————————————  (1)
//   PATH-NAMES=*PATH-NAME(PATH=*), -
//   OUTPUT=PATHNAMES.LIST
//END
```

```
/START-EDT ————————————————————————————————————————————————————————  (2)
@READ ‘PATHNAMES.LIST‘          Deletion of all path names which are not to be restored
@WRITE ‘PATHNAMES.LIST‘
@HALT
```

```
/START-HSMS
//RESTORE-NODE-FILES -   ————————————————————————————————————————  (3)
//   PATH-NAMES=*FROM-FILE(LIST-FILE-NAME=PATHNAMES.LIST), -
//   SELECTION-BOUNDARY=*SPECIFIED-PATHS, -
//   REPLACE-FILES=*YES(PROTECTION-RESPECTED=*NONE), -
//   OPERATION-CONTROL=*PARAMETERS(WAIT-FOR-COMPLETION=*YES)
//END
```

(1) First, all the files of the local BS2000-UFS are selected to obtain a complete list of file names. The list of file names is output to the file PATHNAMES.LIST.

(2) An editor such as EDT is then called to delete the path names which should not be restored from the file PATHNAMES.LIST.

(3) All the path names still present in the file PATHNAMES.LIST are restored. Any existing files with the same path names are overwritten during the restore.

The HSMS administratot transfers files from a workstation to an “empty” workstation

The workstation is mounted in POSIX under /HSMS/WORKSTA3 and can be accessed via NFS.

To do this, the decentralized full backup is used.

```
/START-HSMS
//RESTORE-NODE-FILES -  ————————————————————————————————————————  (1)
//   PATH-NAMES=*FROM-FILE(LIST-FILE-NAME=WORKSTA3.PATH.LIST), -
//   SELECTION-BOUNDARY=*ALL-FILE-SYSTEMS, -  ————————————————————  (2)
//   NEW-PATH-NAMES=*BY-RULE(NEW-NODE-ID=WORKSTA9), -  ——————————  (3)
//   OPERATION-CONTROL=*PARAMETERS(REPORT=*FULL,OUTPUT=WORKSTA9.LIST)
//END
```

(1) All files listed in the file WORKSTA3.PATH-LIST are transferred to the workstation WORKSTA9.

(2) In addition, all node files and directories on the subordinate levels of the directories listed in WORKSTA3. PATH.LIST are transferred.

(3) This attempt to transfer files succeeds only if the workstation WORKSTA9 was placed under HSMS management beforehand, using the HSMS statement MODIFY-NODE-PARAMETERS.

The HSMS administrator archives the file tree of a passive workstation

Perform preparatory activities:

```
//START-HSMS
//CREATE-ARCHIVE -  ————————————————————————————————————————  (1)
//   ARCHIVE-NAME=ARCHIVAL, -
//   ALLOWED-USAGE=*NODEARCHIVAL, -
//   DIRECTORY-NAME=ARCHIVAL.DIR, -
//   RETENTION-PERIOD=30
//MODIFY-ARCHIVE -  ————————————————————————————————————————  (2)
//   ARCHIVE-NAME=ARCHIVAL, -
//   VOLUMES=*ADD(VOLUMES=(VOLUM1,VOLUM2,...))
//MODIFY-NODE-PARAMETERS -
//   NODE-ID=WORKSTA1, -
//   HSMS-CONTROL=*YES
//END
```

(1) Sets up the ARCHIVAL archive for the long-term archiving of node files. By default, only the owner of the archive has access to it. The directory file is named ARCHIVAL.DIR. The archived node files cannot be deleted for 30 days.

Note

In order for nonprivileged users to use the directory file, it must be shareable (USER-ACCESS=*ALL-USERS or *SPECIAL).

(2) Enters volumes in the pool of free volumes of the ARCHIVAL archive.

Start archiving:

```
//START-HSMS
//ARCHIVE-NODE-FILES -   ————————————————————————————————————————————  (3)
//   PATH-NAMES=*PATH-NAME(PATH=/DIR,NODE-ID=WORKSTA1), -
//   SELECTION-BOUNDARY=*ALL-FILE-SYSTEMS, -
//   ARCHIVE-NAME=ARCHIVAL, -
//   OPERATION-CONTROL=*PARAMETERS(REPORT=*FULL,OUTPUT=ARCHLIST)
//END
```

(3) Archives the directory /DIR of the workstation WORKSTA1. Also archives all the node files and directories located on the subordinate levels of the directory /DIR (SELECTION-BOUNDARY=*ALL-FILE-SYSTEMS). The files and directories are stored in the ARCHIVAL archive.
A full report is output, including a list of all archived node files. The report is written to the output file ARCHLIST.

5.8 HIPLEX and HSMS

HIPLEX is the concept which allows several hosts to be grouped together in a computer network. This means that users can address their requests to a computer network and not have to send them to one of the hosts. The advantage of this concept is that continuous request processing is guaranteed even if one or more hosts of the network are unavailable. Files that have been backed up or archived on one host, can be restored on another host in the network if that host crashes. When the crashed host is up and running again, the situation can be returned to the pre-crash status while the continuity of HSMS activities is guaranteed.

To be able to move HSMS activities from one host in the network to another without problems, the environment has to be adjusted. The data and metadata available on the first host must be made available on the substitute host. General availability within the network is achieved by means of SM pubsets. If the data and metadata is located on an SM pubset, the HSMS activities can be continued after a host crashes by importing the SM pubset on a substitute host (see also section "HSMS in an SM pubset environment").

When the HSMS administrator issues the HSMS statement RECOVER-REQUESTS, all requests that were issued on the crashed host are integrated in the local requests. The SHOW-REQUESTS statement can be used to output the status of the requests at the moment the host crashed. Requests that had not been started on the crashed host can be processed on the substitute host. Requests that were running just as the host crashed and had not been interrupted can be restarted on the substitute host.

For the user, the host changeover is transparent: the user starts off working with an SM pubset and a number of archives on that SM pubset. After a host crash, the user’s requests are processed by another, the host itself not needing to perform any actions.

5.8.1 Recovery of DMS files

The recovery function is available for all DMS files located on an SM pubset. Recovery is based on the reconfigurability of SM pubsets and on the extended HSMS request management within an SM pubset.

Manual recovery

If a host on the computer network is no longer available, the HSMS administrator must ensure that the DMS can access the SM pubset:

- If the SM pubset was imported as an exclusive pubset, the HSMS administrator must import it to the substitute host. The control file, which is on the SM pubset, is read. All metadata (archives, declarations for the SM pubset, ...) are added to the local metadata. The SM pubset is then available again for new HSMS activities.
- If the pubset is a shared SM pubset, HSMS activities are still possible for the remaining sharers provided that the SM pubset can actually be accessed. If the master crashes, the sharers must of necessity be reconfigured. The MSCF subsystem provides a control task for the automatic reconfiguration by a backup master.

Once the SM pubset can be accessed again, the HSMS administrator should decide whether the requests that were being processed by the crashed host need to be available again for the restart. The requests of the SM pubset can be output with the HSMS statement SHOW-REQUESTS. The “inhibiting” host is displayed for each request. For more detailed information on request inhibits in SM pubsets, see section "Request management".

Finally the HSMS administrator must issue the HSMS statement RECOVER-REQUESTS to normalize the status of all requests linked to a host and selected for recovery. The description of the RECOVER-REQUESTS statement explains the results that are obtained by a recovery operation.
For shared pubsets, the recovery can only be issued on the master sharer.

Note on MSCF reconfiguration

> If an error occurs in the MSCF reconfiguration, the pubset will probably be switched to QUIET mode. From the HSMS point of view, the pubset is, however, still in operation (because it has not been exported). But the pubset cannot be accessed any more. The system administrator should then always remedy the situation by either calling a new master for the SM pubset or exporting it to each slave.

Automatic recovery

The automatic recovery of HSMS requests is only possible for configurations with shared SM pubsets. Backup sharers must be in operation to be able to detect the crash of one of the other sharers. HSMS requests can only be recovered after a successful MSCF recovery.

HSMS requests are recovered automatically by using the product PROP-XT. Events can be defined with PROP-XT that are based on console messages. These events are defined in an S procedure that runs on each host. If certain MSCF messages are registered which refer to a successful MSCF recovery, the recovery of HSMS requests is initiated.

For further information on PROP-XT, see the “PROP-XT” manual [19].

Example of an S procedure for automatic recovery

The following S procedure describes an HSMS watchdog task which automatically recovers HSMS requests after a slave crash is detected. For further information on watchdogs, see the “HIPLEX MSCF” manual [20].

Structure of the S procedure

```
Start PROP-XT processing;
Request operator role ROLEALL which must first have been defined with SYSPRIV;
Create PROP-XT event with the name DMS03B0 which is based on the console message
DMS03B0;
Enter observation loop which is controlled by a job variable;
    Wait for event DMS03B0;
    Output command /SHOW-SHARED-PUBSET <pubset-ID> in variable PUBSET-LIST;
    Search each sharer registered by /SHOW-SHARED-PUBSET to establish whether the
    local host has master access to the pubset and to determine the name of the
    crashed partner;
        If master access and at the same time the slave crashes:
            output HSMS statement RECOVER-REQUESTS;
    Delete variable PUBSET-LIST;

Disable PROP-XT;
Exit
```

S procedure

The explanations of the run are documented in the procedure.

```
/SET-PROCEDURE-OPTIONS DATA-ESCAPE-CHAR=*STD
/
/ "-------------------------------------------------------------------------"
/ "-- Example of an HSMS watchdog task                                    --"
/ "--                                                                     --"
/ "-- This SDF-P procedure sets up a watchdog task which was designed     --"
/ "-- for the automatic recovery of HSMS requests after the crash of      --"
/ "-- a slave sharer.                                                     --"
/ "--                                                                     --"
/ "-- The following environment is necessary:                             --"
/ "-- * Product PROP-XT must be installed and the subsystem created;      --"
/ "-- * Operator role ROLEALL must be defined with SYSPRIV:               --"
/ "--   /CREATE-OPER-ROLE ROLEALL, ROUT=*ALL                             --"
/ "--   /MODIFY-OPER-ATTR USER=TSOS,ADD=ROLEALL                          --"
/ "-- * The job variables must be available, in particular $SYSJV.HOST    --"
/ "--                                                                     --"
/ "-- The batch task uses:                                                --"
/ "-- * the job variable HSMS.WATCHDOG.MON to monitor the task            --"
/ "-- * the file HSMS.WATCHDOG.LST to output the task                     --"
/ "--                                                                     --"
/ "-- Start the watchdog task with the SYSHSMS privilege:                 --"
/ "-- /ENTER-PROC <filename-containing-this-proc>, JOB-NAME=HSMSWDOG      --"
/ "--                                                                     --"
/ "-- Stop the watchdog task with:                                        --"
/ "-- /SETJV HSMS.WATCHDOG.MON,'STOP'                                     --"
/ "--                                                                     --"
/ "-- The watchdog task should be created on every host that has          --"
/ "-- master access to at least 1 SM pubset.                              --"
/ "-- During the operation, the watchdog task waits for console message   --"
/ "-- DMS03B0 and outputs the HSMS statement RECOVER-REQUESTS when the    --"
/ "-- local master access and slave crash conditions occur.               --"
/ "--                                                                     --"
/ "-- Note:                                                               --"
/ "-- This example only supports the recovery of requests for master      --"
/ "-- sharers after slave crashes. It does not support the recovery of    --"
/ "-- requests after a master crash.                                      --"
/ "-------------------------------------------------------------------------"
/
/
/ "-------------------------------------------------------------------------"
/ "-- Declaration of the local variables                                  --"
/ "--                                                                     --"
/ "-------------------------------------------------------------------------"
/

/ " Create variable SYSPOP for the dialog with PROP-XT "
/DECLARE-VARIABLE NAME=SYSPOP(TYPE=STRUCTURE)
/
/ " Variable PUBSET-LIST to obtain the output of /SHOW-SHARE-PUBSET "
/DECLARE-VARIABLE VARIABLE-NAME=PUBSET-LIST(TYPE=*STRUCTURE(DEFINITION=-
/*DYNAMIC)),MULTIPLE-ELEMENTS=*LIST
/
/ " Variables PUBSET and SHARER to be able to access the elements of "
```

```
/ " PUBSET-LIST "
/DECLARE-VARIABLE VARIABLE-NAME=PUBSET(TYPE=*STRUCTURE(DEFINITION=*DYNAMIC))
/DECLARE-VARIABLE VARIABLE-NAME=SHARER(TYPE=*STRUCTURE(DEFINITION=*DYNAMIC))
/
/ " Variable LOCAL-HOST, initializes with the local host name "
/DECLARE-VARIABLE VARIABLE-NAME=LOCAL-HOST(TYPE=*STRING)
/
/ " Variable CRASH-HOST for the name of the crashed host "
/DECLARE-VARIABLE VARIABLE-NAME=CRASH-HOST(TYPE=*STRING)
/
/ " 2 Boolean indicators to determine master and crash events "
/DECLARE-VARIABLE VARIABLE-NAME=MASTER-IND(TYPE=*BOOLEAN)
/DECLARE-VARIABLE VARIABLE-NAME=CRASH-IND(TYPE=*BOOLEAN)
/
/ "-------------------------------------------------------------------------"
/ "-- Assign SYSOUT to file                                                --"
/ "--                                                                      --"
/ "-------------------------------------------------------------------------"
/
/ASSIGN-SYSOUT  TO=HSMS.WATCHDOG.LST
/
/ "-------------------------------------------------------------------------"
/ "-- Resolve local host name with job variable $SYSJV.HOST               --"
/ "--                                                                      --"
/ "-------------------------------------------------------------------------"
/
/LOCAL-HOST = (JV('$SYSJV.HOST'))
/INFORM-OPERATOR MSG=-
/     '*** HSMS watchdog started on host &(LOCAL-HOST) ***'
/
/ "-------------------------------------------------------------------------"
/ "-- Prepare PROP-XT environment                                          --"
/ "--                                                                      --"
/ "-------------------------------------------------------------------------"
/
/BEGIN-BLOCK
/
/  " Start PROP-XT preparation "
/  BEGIN-PROP-PROCESS            -
/       PROCESS-NAME=SMCHECK
/  START-PROP-OBJECT-MONITORING  -
/       OBJECT-NAME=OBJ-MONITOR
/
/  " Request operator role ROLEALL "
/  SEND-TO-PROP-OBJECT OBJECT-NAME=OBJ-MONITOR     -
/      ,TYPE=CMD(IMPLICIT-EVENTS=CMD-TERMINATION)  -
/      ,DATA='REQ-OPER-ROLE ROLEALL'               -
/      ,REFERENCE-NAME=REQOPERROLE
/  WAIT-FOR-PROP-EVENTS -
/       EVENT-NAME=REQOPERROLE, TIME-LIM=10
/  IF CONDITION=(SYSPOP.MAINCODE <> '0000')
/     GOTO LABEL=PROPERR
/  ELSE-IF CONDITION=(SYSPOP.MES-ID <> 'NBR0740')  -
/                 OR (SYSPOP.I3 <> 'CMD0001')
/     MAINCODE=SYSPOP.I3
/     MESSAGE=MSG(MSG-ID='&MAINCODE')
/     SHOW-VARIABLE (MAINCODE,MESSAGE)
/     GOTO LABEL=PROPEND
/  END-IF
```

```
/
/  " Create event DMS03B0 to wait for console message DMS03B0 "
/  START-PROP-EVENT-MONITORING   -
/        EVENT-NAME=DMS03B0      -
/       ,SELECT-EVENT=FROM-OBJECT(EVENT-DATA=*SYSTEM-MSG(MSG-ID=DMS03B0))
/
/  "-------------------------------------------------------------------------"
/  "-- Watchdog loop to trace console message DMS03B0                     --"
/  "--                                                                     --"
/  "-------------------------------------------------------------------------"
/
/  CREATE-JV HSMS.WATCHDOG.MON
/  SET-JOB-STEP
/  MODIFY-JV  JV-CONTENTS=(JV-NAME=HSMS.WATCHDOG.MON),SET-VALUE='CHECK'
/
/  INFORM-OPERATOR MSG=-
/       '*** HSMS watchdog enters observation loop ***'
/
/  WHILE  ( JV('HSMS.WATCHDOG.MON') = 'CHECK' )
/
/    INFORM-OPERATOR MSG=-
/         '*** HSMS watchdog waits for message DMS03B0 ***'
/
/    " Wait for event DMS03B0, timeout after 5 minutes "
/    WAIT-FOR-PROP-EVENTS  EVENT-NAME=DMS03B0,TIME-LIMIT=300
/
/    IF ( SYSPOP.MAINCODE <> '0B18' )
/
/      IF CONDITION=(SYSPOP.MAINCODE <> '0000')
/        GOTO LABEL=PROPERR
/      END-IF
/
/      INFORM-OPERATOR MSG=-
/      '*** HSMS watchdog checks system &(SYSPOP.I0) on pubset ***'//-
/      '*** &(SYSPOP.I1) ***'
/
/      "-------------------------------------------------------------------"
/      "-- Output command /SHOW-SHARE-PUBSET redirected to               --"
/      "-- OPS variable                                                  --"
/      "-------------------------------------------------------------------"
/
/     ASSIGN-STREAM STREAM-NAME=SYSINF,TO=*VARIABLE(VARIABLE-NAME=PUBSETLIST)
/      SHOW-SHARED-PUBSET &(SYSPOP.I1)
/      SET-JOB-STEP
/      ASSIGN-STREAM STREAM-NAME=SYSINF,TO=*DUMMY
/
/      "-------------------------------------------------------------------"
/      "-- Check status of pubset for partner                            --"
/      "--                                                               --"
/      "-------------------------------------------------------------------"
/
/      FOR PUBSET=*LIST(PUBSET-LIST)
/
/        " Search sharer list to determine master and crashed hosts "
/        MASTER-IND = FALSE
/        CRASH-IND  = FALSE
/        FOR SHARER=*LIST(PUBSET.LIST)
/          " Check whether the local host is the master host "
/          IF (    (SHARER.PARTNER-NAME=LOCAL-HOST)  -
```

```
/              AND (SHARER.SHARER-TYPE='*MASTER') )
/            MASTER-IND = TRUE
/          END-IF
/          " Check whether partner has crashed "
/          IF (SHARER.SYS-ID='&(SYSPOP.I0)')
/           CRASH-HOST = SHARER.PARTNER-NAME
/           IF ((SHARER.SHARER-STA='*CRASH') OR (SHARER.SHARER-STA='*CHECK'))
/              CRASH-IND = TRUE
/            END-IF
/          END-IF
/        END-FOR
/
/        " If not master, output console message "
/        IF NOT MASTER-IND
/          INFORM-OPERATOR MSG=-
/          '*** HSMS watchdog - local host is not master sharer ***'//-
/          '*** for pubset &(SYSPOP.I1) ***'
/        END-IF
/
/        " If no crash detected, send console message "
/        IF NOT CRASH-IND
/          INFORM-OPERATOR MSG=-
/          '*** HSMS watchdog - no crash for partner &(CRASH-HOST) ***'//-
/          '*** detected ***'
/        END-IF
/
/        " If host=master and crash detected, recovery is "
/        " required "
/        IF (MASTER-IND) AND (CRASH-IND)
/          " Recover crashed slave sharer "
/          INFORM-OPERATOR MSG=-
/          '*** HSMS watchdog - recovery for partner &(CRASH-HOST) ***'//-
/          '*** on pubset &(SYSPOP.I1) called ***'
/          START-HSMS
//           RECOVER-REQUESTS ENV=*SYS-MAN(&(SYSPOP.I1)) -
//                           ,HOST-NAME=&(CRASH-HOST)
//           STEP
//         END
/        END-IF
/
/      END-FOR
/      SET-JOB-STEP
/
/      FREE-VAR PUBSET-LIST
/      SET-JOB-STEP
/
/    END-IF
/
/  END-WHILE
/
/  INFORM-OPERATOR MSG=-
/       '*** HSMS watchdog - observation loop is exited ***'
/
/END-BLOCK
/
/IF-BLOCK-ERROR
/  GOTO LABEL=PROPERR
/ELSE
/  GOTO LABEL=PROPEND
```

```
/END-IF
/
/ "-------------------------------------------------------------------------"
/ "-- Handling of PROP-XT errors                                          --"
/ "--                                                                     --"
/ "-------------------------------------------------------------------------"
/
/PROPERR:
/  MAINCODE=MAINCODE(); SUBCODE1=SUBCODE1(); SUBCODE2=SUBCODE2()
/  MESSAGE=MSG(MSG-ID='&MAINCODE')
/  SH-VARIABLE (MAINCODE,SUBCODE1,SUBCODE2,MESSAGE)
/  SH-VARIABLE (SYSPOP)
/
/ "-------------------------------------------------------------------------"
/ "-- Exit                                                                 --"
/ "--                                                                     --"
/ "-------------------------------------------------------------------------"
/
/PROPEND:
/  END-PROP-PROCESS
/  INFORM-OPERATOR MSG=
/       '*** HSMS watchdog deinitialized ***'
/  ASSIGN-SYSOUT  TO=*PRIMARY
/  EXIT-JOB MODE=NORMAL,SYSTEM-OUTPUT=NONE
/
/EXIT-PROCEDURE
```

5.8.2 HIPLEX support for node files

BS2000 provides HIPLEX support for node files which are managed by HSMS. The node files must belong either to BS2000-UFS or to remote passive nodes.

HIPLEX support for node files and for DMS files is very similar. SM pubsets must also be used for HIPLEX support for node files.

HSMS offers the following support if the data and the metadata are saved on one SM pubset:

- After a host crashes HSMS continues backup/restore for both BS2000-UFS and for remote workstations on a substitute host. The archive is made available by importing the SM pubset. The archive is made available by importing the SM pubset. Once the crashed host is up and running again, HSMS makes it possible to return to this host without problems.
- Requests which were interrupted by a system failure are started again by HSMS on the substitute host.
- HSMS recognizes node reconfiguration. Reconfiguration is understood to be the reorganization of nodes, where the affected activities are transferred from one host to another without causing a system failure. The reconfiguration can be carried out for all of the nodes or only for certain POSIX-HIPLEX users.

5.8.2.1 HSMS and the UNIX file system (UFS)

When working with node files the term “file system” has a particular meaning – and in particular in the UNIX-/POSIX world. In this context, the following points always apply to file systems:

- The BS2000-UFS includes the root file system and the client file systems which are mounted on the root file system or another client file system.
- A passive node which is accessed using NFS includes a remote passive client file system which is mounted using NFS under the /HSMS directory of BS2000-UFS.

5.8.2.2 General prerequisites

The following prerequisites must be fulfilled, before you can use HIPLEX support for node files:

- For each POSIX user, the POSIX information entered in the user catalog must be identical on all hosts.
- A POSIX client file system for which the container is located on an SM pubset, must be mounted in the file system under the agreed directory “pubset”. For example, the file system for SM pubset A must be mounted under /SM-PUB/A. HSMS uses this convention to help it find where the POSIX data to be backed up is located and where the corresponding metadata is to be saved.

  If a POSIX file system is to be mounted which has a path name containing /SM-PUB/A, then the POSIX installation program checks to see whether the file system container is really located on SM pubset A.
- For this concept the mount points must be identical on all hosts.
- The system administrator must avoid contradictory mount points within the file structure on the various hosts by making sure that all path names are identical.

  Each individual environment must be taken into consideration when creating HSMS statements. Each environment requires its own HSMS statement.
  The operand value *ALL is now limited to a single environment and can no longer cover all nodes if you are using several different environments.
- In the event of a system failure the system administrator must perform several actions to make sure that the activities can be continued on a substitute host.After the cause of the failure has been eliminated, the activities can again be taken up by the original host (see section "Usage models").
- HIPLEX support requires local archiving of each individual SM pubset. Global archiving is not possible. The system administrator cannot, for example, carry out central archiving in a global archive on an SF pubset.

5.8.2.3 Restrictions

Because of POSIX restrictions HIPLEX support is not offered for the root file system of the POSIX file system. HIPLEX support is only available for the client file system of the POSIX file system.

5.8.2.4 Usage models

The following illustrate all supported usage models for BS2000-UFS and remote workstations.

BS2000-UFS

Environment description

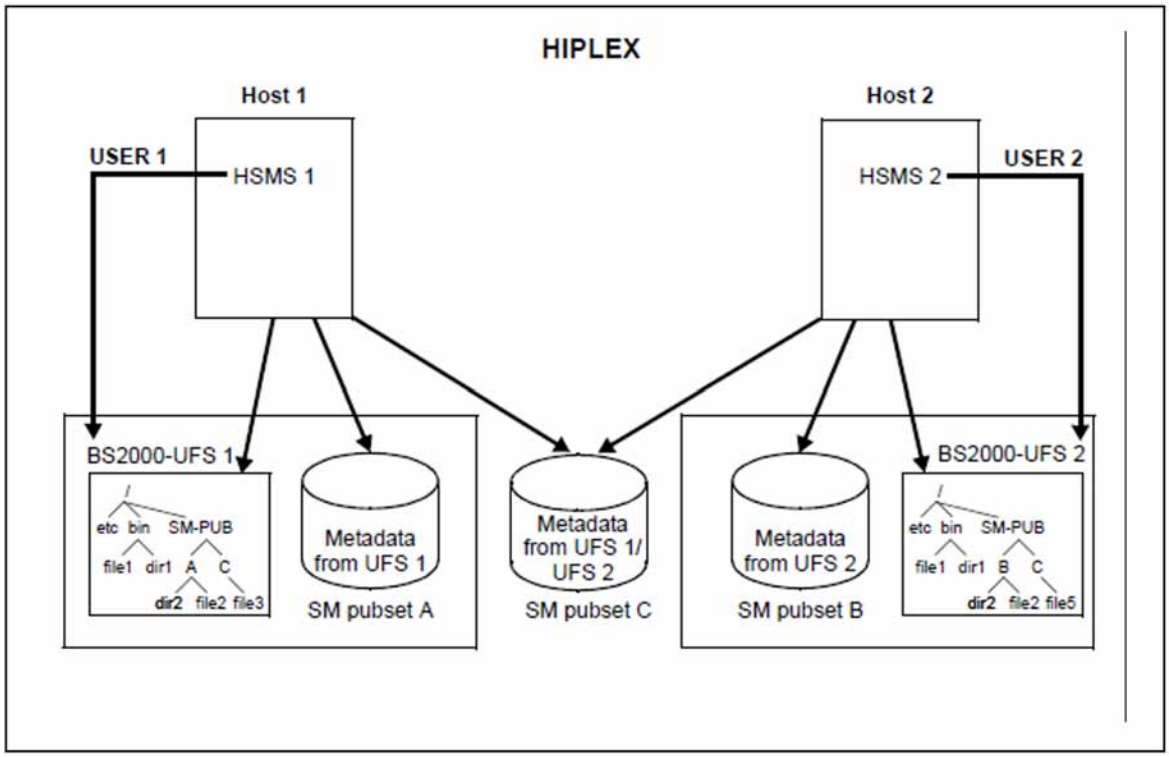


Figure 27: HSMS in a HIPLEX for BS2000-UFS: accessibility of POSIX files and HSMS metadata

- A HIPLEX system has several hosts, in this example they are Host 1 and Host 2.
- Each host supports the HSMS functions for its BS2000-UFS.
- The POSIX administrator must store all file system containers which are to be used for HIPLEX support on one of the pubsets (shared or exclusive). In the diagram shown above the file system containers are stored on SM pubset C.
- The affected SM pubsets must be brought under HSMS control (CREATE-SM-PUBSET-PARAMETERS, HSMS 1 for A and C, HSMS 2 for B and C).
- Several archives must be stored on SM pubsets A, B and C.
- On Host 1 the BS2000-UFS must be brought under HSMS control in each environment according to BS2000-UFS (SF pubset for the root file system, SM pubset A for the branch /SM-PUB/A, SM pubset C for the branch /SM-PUB/C):

```
//MODIFY-NODE-PARAMETERS NODE-ID=*BS2000-UFS,ENVIRONMENT=*S-F
//MODIFY-NODE-PARAMETERS NODE-ID=*BS2000-UFS,ENVIRONMENT=*S-M(CAT-ID=A)
//MODIFY-NODE-PARAMETERS NODE-ID=*BS2000-UFS,ENVIRONMENT=*S-M(CAT-ID=C)
```

- On Host 2 the BS2000-UFS must be brought under HSMS control in each environment according to BS2000-UFS (SF pubset for the root file system, SM pubset B for the branch /SM-PUB/B, SM pubset C for the branch /SM-PUB/C):

```
//MODIFY-NODE-PARAMETERS NODE-ID=*BS2000-UFS,ENVIRONMENT=*S-F
//MODIFY-NODE-PARAMETERS NODE-ID=*BS2000-UFS,ENVIRONMENT=*S-M(CAT-ID=B)
//MODIFY-NODE-PARAMETERS NODE-ID=*BS2000-UFS,ENVIRONMENT=*S-M(CAT-ID=C)
```

During normal operation HSMS saves the corresponding metadata in the environment in which the POSIX data is also saved:

- When you back up a file system which is located under the branch /SM-PUB/A, HSMS stores the metadata in an archive on SM pubset A.
- When you back up a file system which is located under the branch /SM-Pub/C, HSMS stores the metadata in an archive on SM pubset C.

Measures to be taken after a system failure

After a system failure on Host 1, Host 2 is the designated substitute host. Once the following measures have been taken HSMS 2 can continue the work of HSMS 1.

Measures to be taken on Host 1	**Measures to be taken on substitute host i.e Host 2**
System failure -> Host 1 is no longer available	
	The *system administrator* must take the following measures:
	1. Import SM pubset A (/IMPORT-PUBSET) 2. Start a POSIX session, if one does not already exist. 3. “Mount” file systems from Host 1 using the POSIX installation program (use SYSDAT.POSIX-BC.FSLIST file); Host 1 users now have the same configuration and access as they did before the system failure. 4. Load and start HSMS, if HSMS is not yet running 5. BS2000-UFS in the new imported environment must be brought under HSMS control.
	The *HSMS administrator* must take the following measures:
	Enter the HSMS statement RECOVER-REQUESTS REQUEST-DATA-ORIGIN=*NODE for SM pubset A, this allows you to process the requests, which were initiated on Host 1 before the system failure.

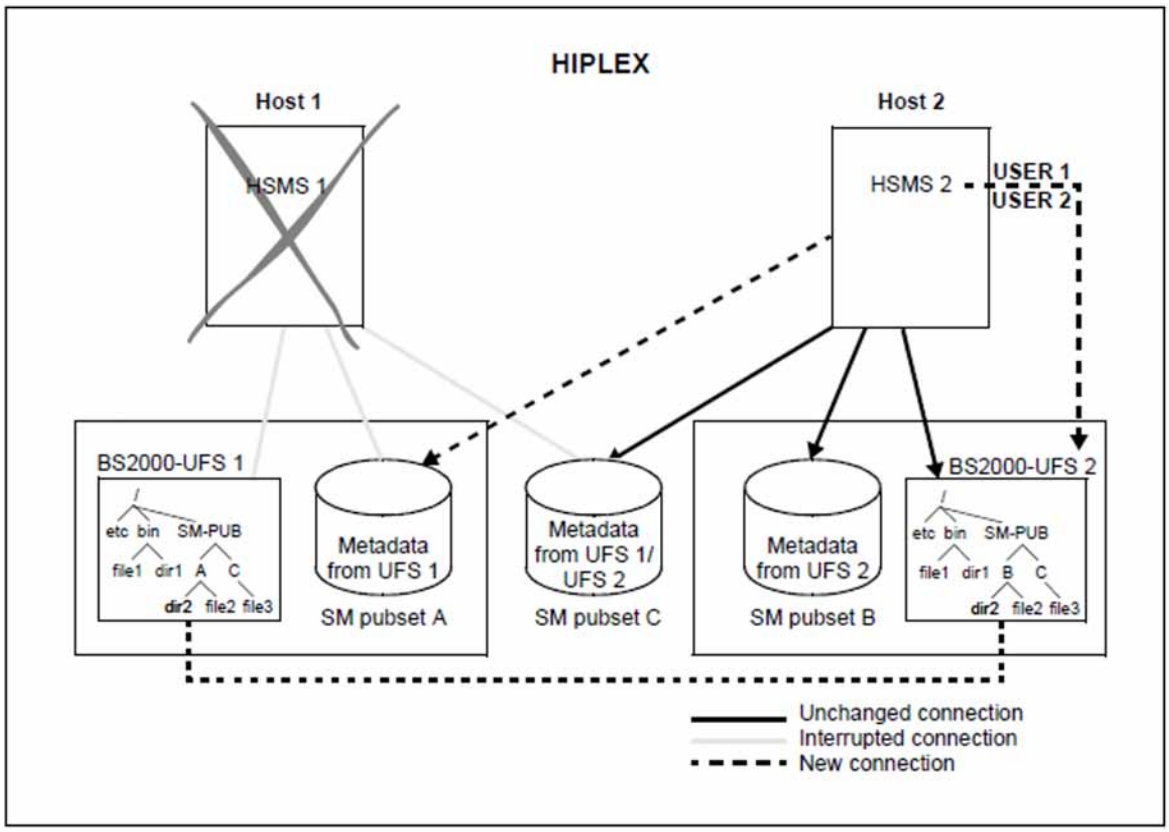


Figure 28: HSMS in a HIPLEX for BS2000-UFS: accessibility of POSIX files and HSMS metadata after a system failure

Measures to be taken after recovery from a system failure

Once Host 1 is up and running again, the node files can be brought onto the original Host and the HSMS activities can be transferred back again. To do this the following measures must be taken:

Measures to be taken on Host 1	**Measures to be taken on substitute host i.e Host 2**
Reason for system failure eliminated -> Host 1 is available again	
	The *system administrator* must take the following measures:
	1. “Delete” the file system using the POSIX installation program 2. Remove BS2000-UFS in SM pubset A from HSMS control 3. Export SM pubset A (/EXPORT-PUBSET)
The *system administrator* must take the following measures:	
1. Import SM pubset A (/IMPORT-PUBSET) 2. Update user catalog: Add users which were added to Host 1 during the crash procedures being taken on Host 2; evaluate information in the file SYSDAT.HIPLEX.USERS on the SM pubset A 3. “Mount” file system from Host 1 using the POSIX installation program (use SYSDAT.POSIX-BC.FSLIST file) 4. Bring BS2000-UFS in the environment SM pubset A and C under HSMS control, if this hasn’t already happened.	
The *HSMS administrator* must take the following measures:	
Enter HSMS statement RECOVER-REQUESTS REQUEST-DATA-ORIGIN=*NODE for SM pubset A to be able to process the requests initiated on Host 1 before the system failure.	

Organization of the file system

The following illustrates the various ways in which the POSIX file system can be organized, to guarantee HIPLEX support for a part of the file system tree.

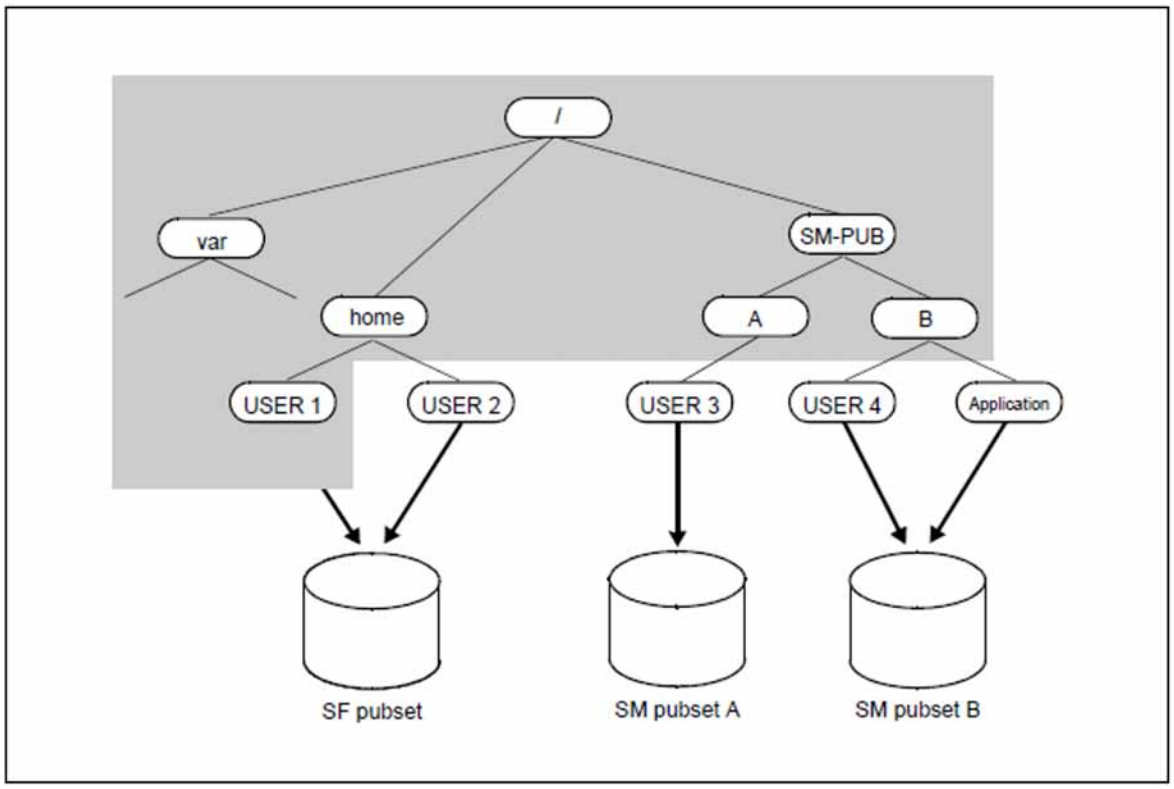


Figure 29: POSIX-/HSMS mechanism: Client file systems on SM pubsets

The SF pubset contains the root file system and the file system container for USER 1 and USER 2, since these are “non-HIPLEX clients”.

The file system container for USER 3 is located on SM pubset A. The file system container for the normal users are stored on SM pubset B. To do this you must keep to the following conventions:

- If the file system container is located on SM pubset A, the corresponding file system must be mounted under the branch /SM-PUB/A.
- If the file system container is located on SM pubset B, the corresponding file system must be mounted under the branch /SM-PUB/B.

The following diagram shows how the POSIX data and HSMS metadata can be accessed during normal system operation.

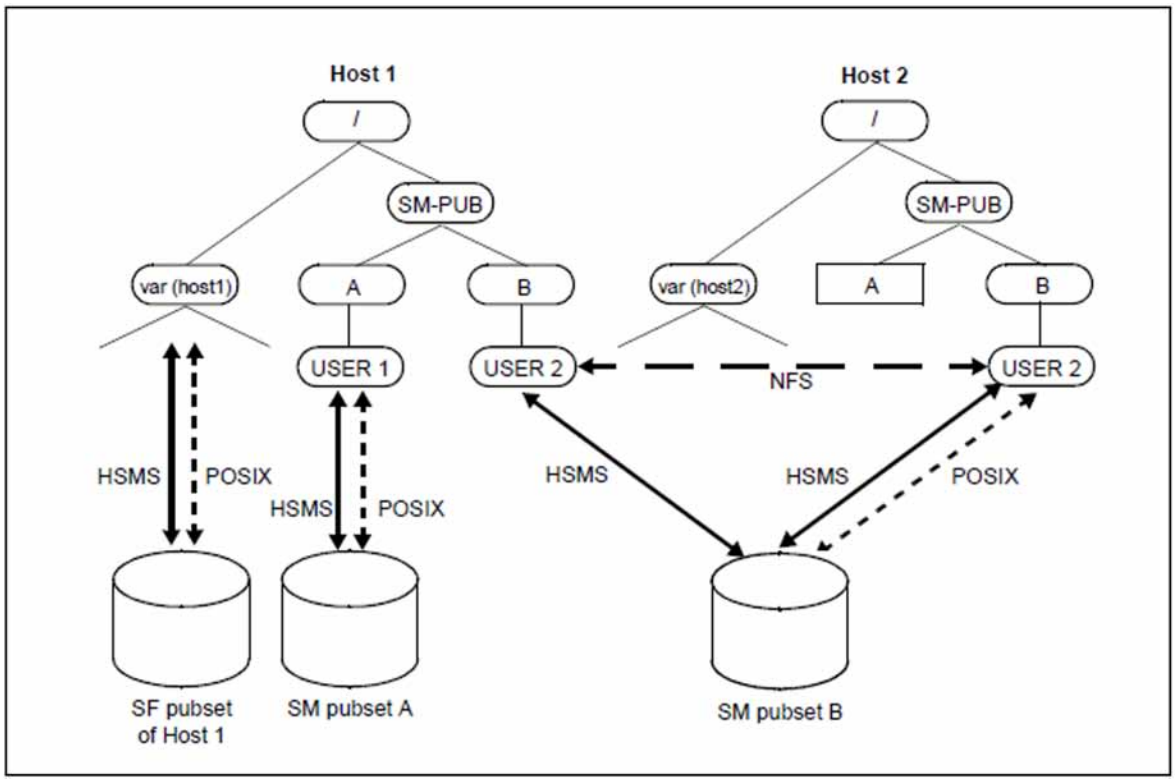


Figure 30: Access to POSIX files and HSMS metadata before a system failure

The UNIX file system on SM pubset A is mounted on Host 1. The UNIX file system on SM pubset B is mounted on Host 2.

After a system failure the system administrator must carry out the measures described in section "BS2000-UFS: Measures to be taken after a system failure". This allows activities to be transferred to a substitute host.
POSIX data and metadata can also be accessed in the way shown in the following diagram.

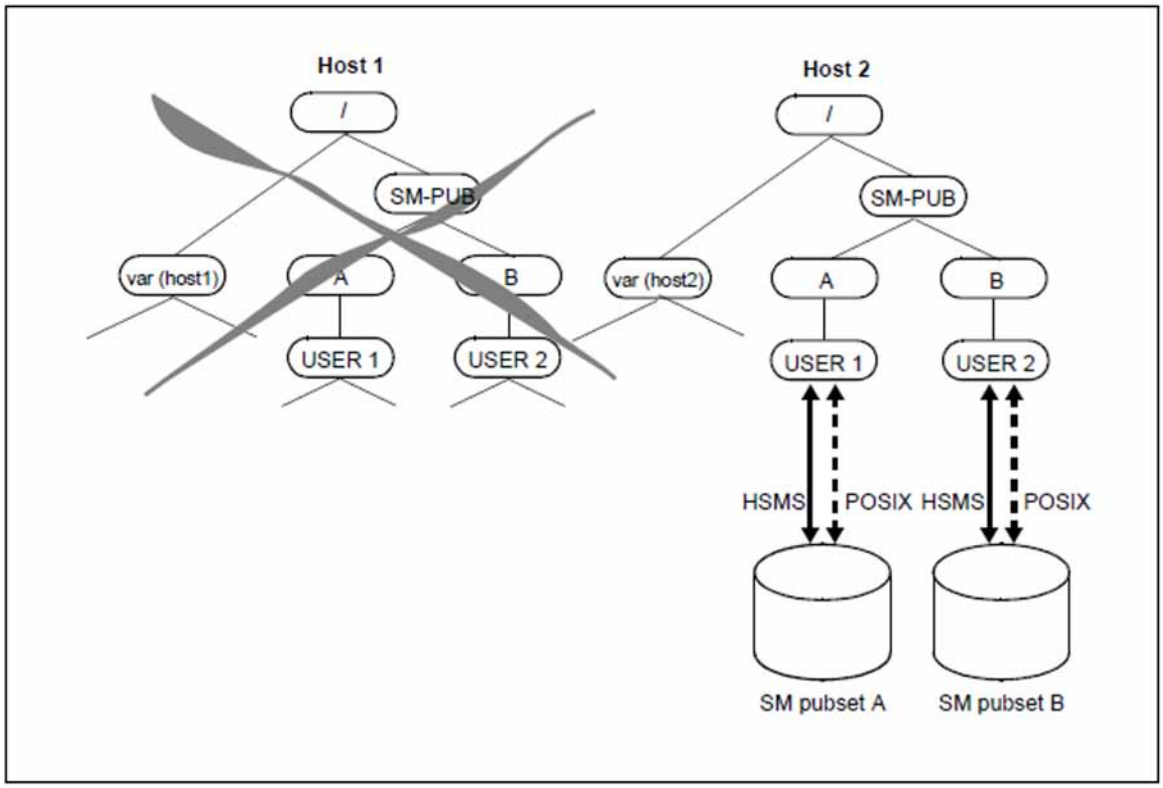


Figure 31: Access to POSIX files and HSMS metadata after a system failure

The UNIX file system on SM pubset A is now no longer mounted on Host 1, it is instead mounted on Host 2.

Remote workstations

Environment description

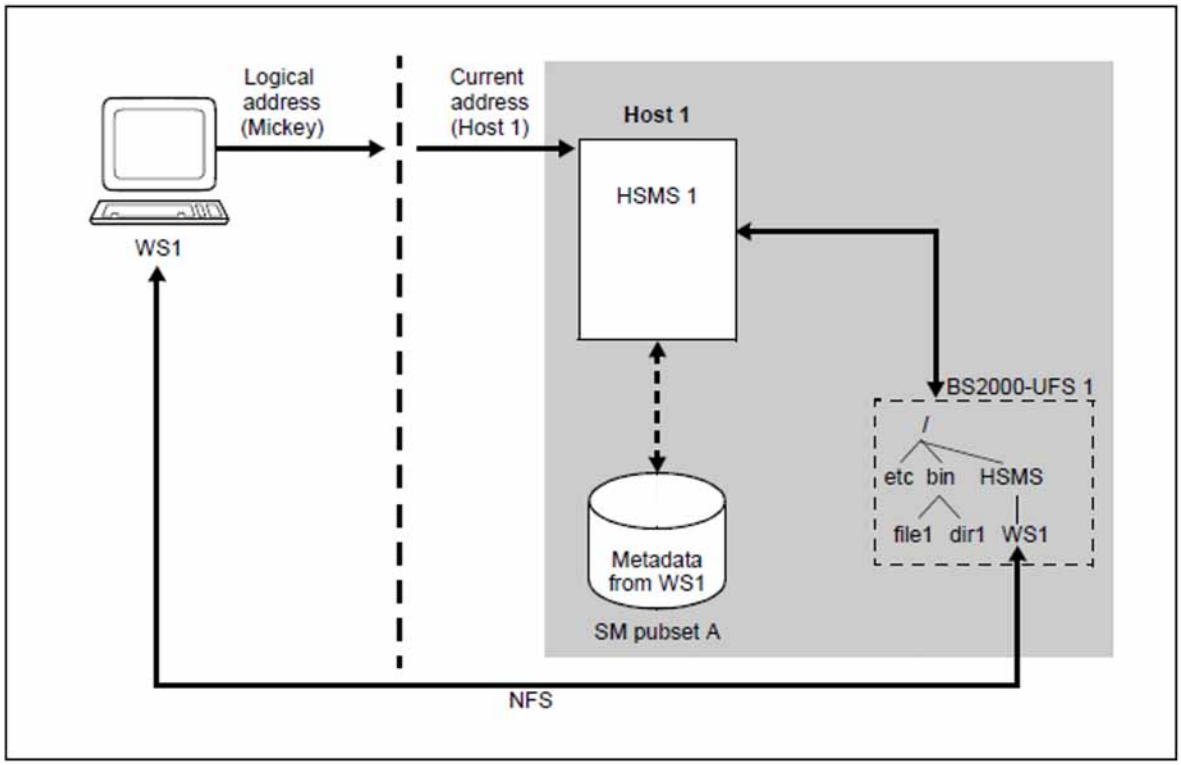


Figure 32: HSMS in a HIPLEX for workstations **with** NFS access: data/metadata stream

- The HIPLEX system includes several hosts.
- Host 1 makes available HSMS functions for the remote workstation WS1.
- The system administrator must save the data for all file systems of the workstations on a single SM pubset (shared or exclusive), in this example on SM pubset A.
- The affected SM pubset (here A) must be brought under HSMS control (//CREATE-SM-PUBSET-PARAMETERS, HSMS 1 for A).
- Several archives must be stored on SM pubset A.
- WS1 in the SM pubset environment must be brought under the control of HSMS 1 (//MODIFY-NODE-PARAMETERS).
- You access the WS1 data with NFS - depending on the type of workstation (see figure 32).
- If you want to be able to access WS1 via NFS (see figure 32), the file system of WS1 must be mounted under /HSMS in the POSIX root file system. In addition, BS2000-UFS must also be brought under HSMS control (//MODIFY-NODE-PARAMETERS).

During normal operation HSMS saves the associated metadata in the environment in which the WS1 data is also saved.

Measures to be taken after a system failure

After a system failure on Host 1, Host 2 is the designated substitute host. Once the following measures have been taken HSMS 2 can continue the work of HSMS 1.

Measures to be taken on Host 1	**Measures to be taken on substitute host i.e Host 2**
System failure -> Host 1 is no longer available	
	The *system administrator* must take the following measures:
	1. /CREATE-VIRTUAL-HOST Define the virtual host. (This BCAM command can be entered when starting HIPLEX – or in other words before the system failure.) 2. /BCACT HOST=Host 2 Activates the virtual host which must now replace Host 1. After the system failure the system administrator calls this procedure.) In a TCP/IP network the address of the recently activated host is distributed transparently. 3. Import SM pubset A (/IMPORT-PUBSET) 4. For workstations with access via NFS: Using NFS “mount” remote workstations, which are managed by Host 1 under /HSMS. 5. Load and start HSMS, if HSMS is not yet running
	The *HSMS administrator* must take the following measures:
	Enter the HSMS statement RECOVER-REQUESTS REQUEST-DATA-ORIGIN=*NODE for SM pubset A, this allows you to process the requests, which were initiated on Host 1 before the system failure.

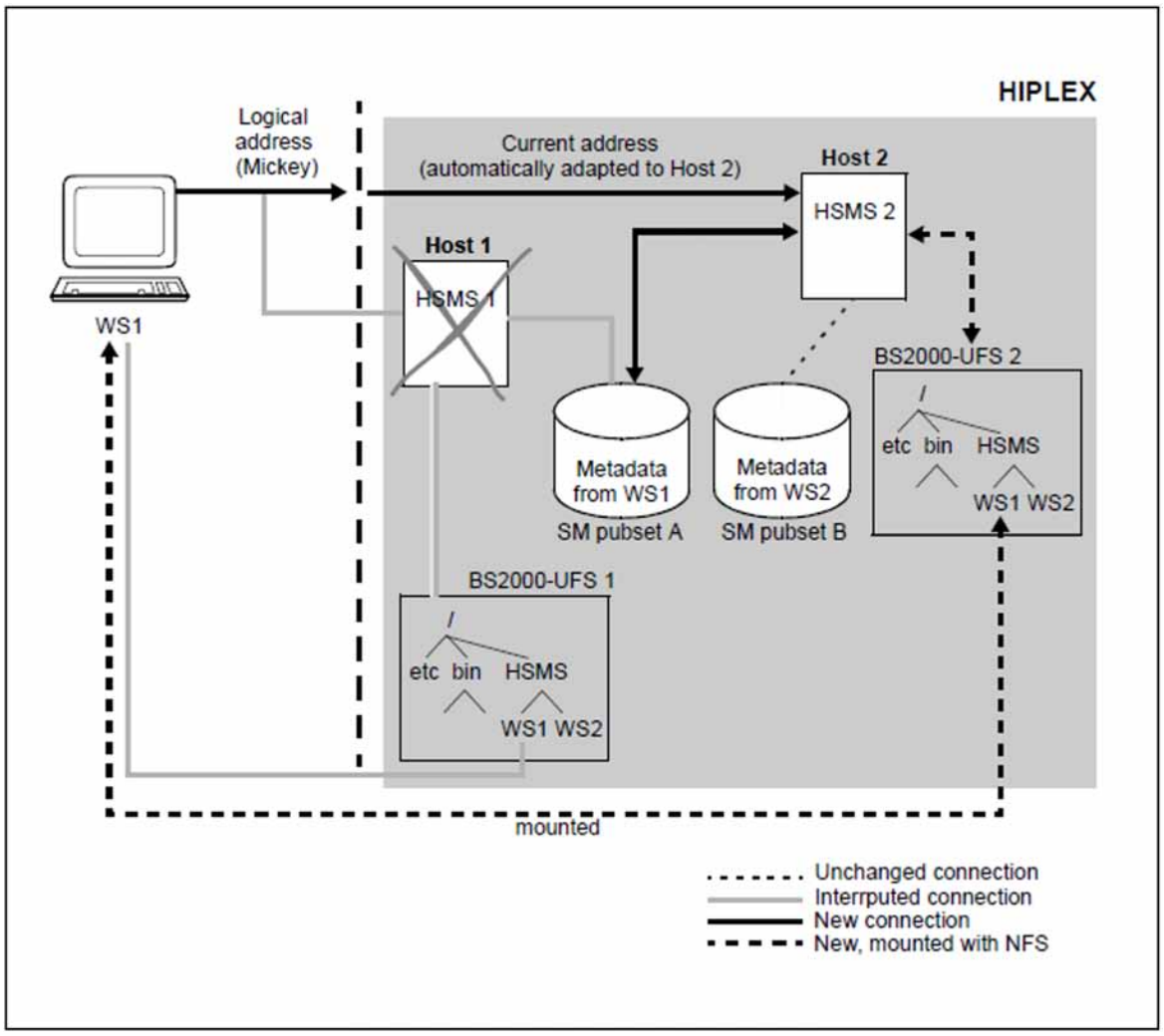


Figure 33: HIPLEX for workstations **with** NFS access: Accessibility of data/metadata after a system failure

Measures to be taken after recovery from a system failure

After Host 1 is back up and running, the HSMS activities for the workstation node files can be transferred back to the original host. To do this the following measures must be taken:

Measures to be taken on Host 1	**Measures to be taken on substitute host i.e Host 2**
Reason for system failure eliminated -> Host 1 is available again	
	The *system administrator* must take the following measures:
	1. Only for workstations with access via NFS: Using NFS “unmount” remote file systems which are saved on SM pubset A. 2. /CREATE-VIRTUAL-HOST Define the virtual host. (This BCAM command can be entered when starting HIPLEX – or in other words before the system failure.) 3. /BCACT HOST=Host 2 Activates the virtual host which must now replace Host 1. After the system failure the system administrator calls this procedure.) In a TCP/IP network the address of the recently activated host is distributed transparently. 4. /BCDAC HOST=Host 2 Deactivate the virtual host and return to the original host, Host 1. 5. Export SM pubset A (/EXPORT-PUBSET)
The *system administrator* must take the following measures:	
1. Import SM pubset A (/IMPORT-PUBSET) 2. Only for workstations with access via NFS: Using NFS “mount” remote workstation which are to be managed by Host 1 under /HSMS.	
The *HSMS administrator* must take the following measures:	
1. Enter HSMS statement RECOVER-REQUESTS REQUEST-DATA-ORIGIN=*NODE for SM pubset A to be able to process the requests initiated on Host 1 before the system failure.	

Organization of the file system

There are no special prerequisites to be taken into account when organizing workstation file systems which are accessed without the use of NFS.

However, for workstations that are accessed using NFS, the workstation file systems must be mounted under /HSMS in the POSIX root file system.

The following diagram shows how the file systems for the workstations WS1 and WS2 must be organized so that HIPLEX support can be guaranteed.

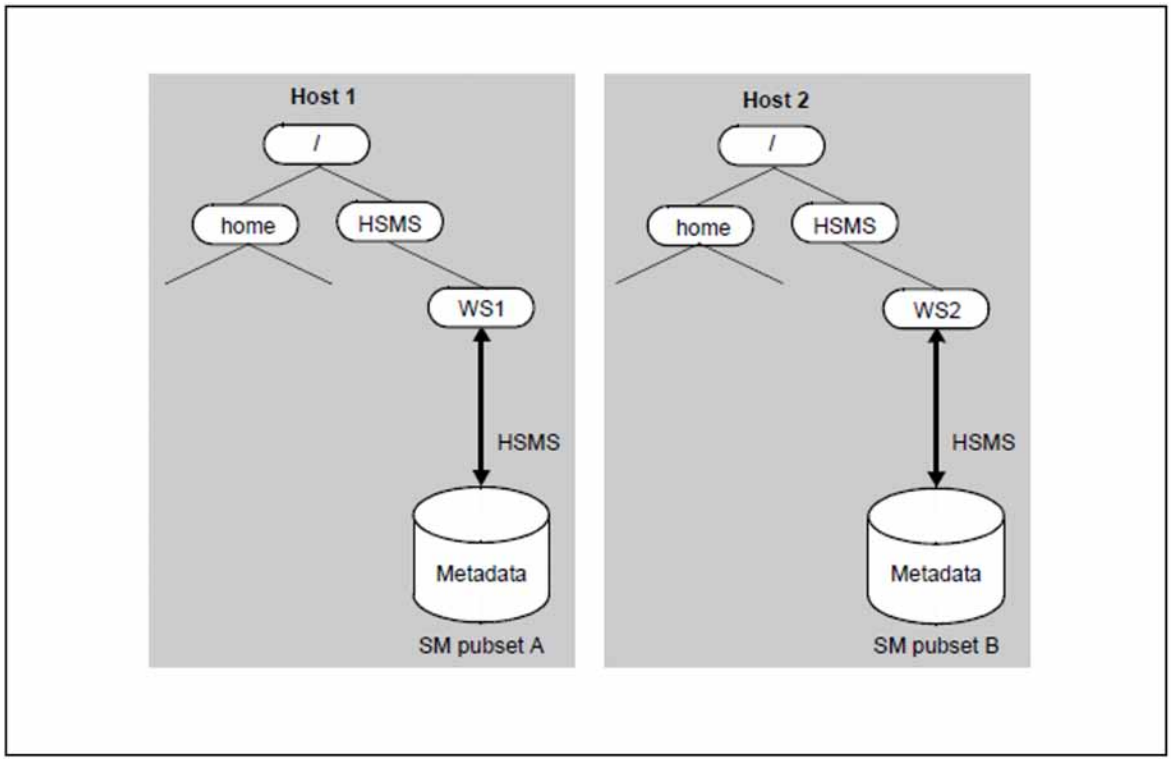


Figure 34: HIPLEX for workstations **with** NFS access: Access to metadata before a system failure

Workstation WS1 is mounted on Host 1 and workstation WS2 on Host 2. HSMS accesses both workstations using NFS.

After a system failure on Host 1 the system administrator must take the measures described in "Remote workstations: Measures to be taken after a system failure" so that the activities can be transferred to a substitute host. Once transferred, the data for workstation WS1 and the HSMS metadata are accessed as shown in the following diagram.

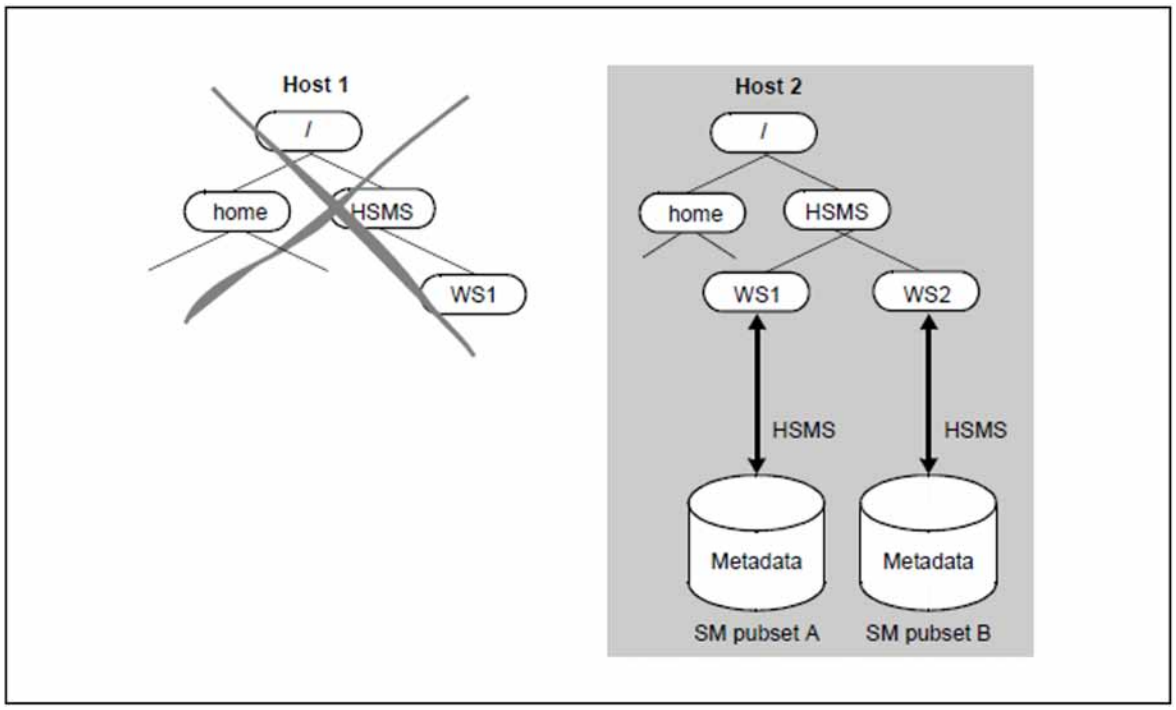


Figure 35: HIPLEX for workstations **with** NFS access: Access to metadata after a system failure

Workstations WS1 and WS2 which are managed by HSMS are now both mounted on Host 2.

5.8.3 Preparing HIPLEX support for UFS runs of HSMS

As previously described you require several SM pubsets to be able to save the HSMS metadata for the node files. For this reason, HIPLEX support requires the migration from an organization without SM pubsets to an organization with SM pubsets. The HIPLEX administrator must take the following measures in the sequence described:

1. Organize node data which is to be located on the SM pubsets.
2. Take the naming conventions for mount points into account for POSIX file systems.
3. Remove all node IDs from HSMS control (//MODIFY-NODE-PARAMETERS NODE-ID=<node-ID>, HSMS-CONTROL=*NO), except for BS2000-UFS, which continues to be valid, since the POSIX root file system always stays in the SF environment.
4. Bring under HSMS control the SM pubsets on which the node data is located (CREATE-SM-PUBSET-PARAMETERS).
5. Using the POSIX installation program, “mount” the user file systems which are to be provided with POSIX /HIPLEX support in the new SM environment.
6. Create new archives in the selected SM environment.
7. Redefine the various node-IDs to bring them under HSSM control in the correct SM environment (//MODIFY-NODE-PARAMETERS NODE-ID=<node-id>, HSMS-CONTROL=*YES, ENVIRONMENT=*SYS-MAN(CAT-ID=<cat-id)).

5.9 Migration control and management of migration archives

In contrast to the other HSMS functions, there are a great many control options for migration, and there is also a greater need for control.

In contrast to backup or archival, which can be used on any pubset, migration is possible only on pubsets that have been placed under HSMS management and assigned to storage level S0. The files of the pubset which reside on an assigned Net-Storage volume (storage type NET-STORAGE) are excluded from migration.

In particular, the HSMS administrator (or system administrator) is responsible for protecting important system files from being migrated (see section “Except file” for more information).

Migration in shared pubset systems

The migration of a file on a shared pubset can also be initiated from the slave. However, the actual processing of requests takes place on the master for performance reasons.

5.9.1 Controlling migration

Migration is controlled in two parts; the addressed settings are discussed in greater detail later on in this section:

- The global parameters (MODIFY-HSMS-PARAMETERS) can be used to define an HSMS-wide migration archive. In addition, the exception file can be defined, observation of the migration inhibit function stipulated and implicit recall controlled for all pubsets under HSMS control.
- The pubset parameters can be used to determine a deviating archive for each pubset. However, the extent to which migration is permitted at all for nonprivileged users is defined on a pubset-specific basis.

5.9.1.1 System migration archive

For migration to be possible, the HSMS administrator must create a migration archive. The system migration archive must be created as a public archive with write access (USER-ACCESS=*ALL-USERS and ACCESS=*WRITE) if nonprivileged users are also to use this archive for migration.

The default system archive is assigned globally for migration by means of:

```
//MODIFY-HSMS-PARAMETERS -
// DEFAULT-HSMS-STORAGE=*PAR(SYSMIGRATE=<archive name>)
```

A different pubset-specific migration archive can be assigned by means of

```
//MODIFY-PUBSET-PARAMETERS -
// STORAGE-LEVEL=*S0(SYSMIGRATE=<archive name>)
```

If SM pubsets are involved in the migration processing, the HSMS administrator must define default system archives for each SM pubset for which migration is permitted. The HSMS administrator can do this either during the definition of the SM pubset with the HSMS statement CREATE-SM-PUBSET-PARAMETERS or later by modifying the SM pubset parameters with the HSMS statement MODIFY-SM-PUBSET-PARAMETERS.

If no migration archive is assigned globally (SYSMIGRATE=*UNDEFINED), migration is not possible unless a pubset has explicitly been assigned a migration archive, since recourse to a global default setting is not possible in this case.

5.9.1.2 Allowing migration

If a pubset is under HSMS management, the HSMS administrator can allow varying degrees of migration for this pubset, depending on the users involved. The following HSMS statement serves this purpose:

```
//MODIFY-PUBSET-PARAMETERS STORAGE-LEVEL=*S0(MIGRATION=...)
```

The following settings can be made for the MIGRATION operand:

- *ALLOWED permits all users to migrate their files themselves to either level S1 or level S2.
- If *S2-ONLY is set, all users can migrate their files, but only to storage level S2. Only the HSMS administrator can migrate files to storage level S1.
- *INHIBITED allows files to be migrated only by the HSMS administrator. No other users can migrate their files themselves.

By default, the right to unrestricted migration is granted (*ALLOWED).

5.9.1.3 Suppressing a file migration inhibit

The HSMS administrator does not have to respect a migration inhibit which a user has set for his file (see section "File migration inhibit"). The following statement can be used by the HSMS administrator to overrule a file migration inhibit set by the file owner:

- in the SF pubset environment:

```
//MODIFY-HSMS-PARAMETERS -
//  MIGRATION-CONTROL=*PARAMETERS(FILE-INHIBIT=*IGNORED)
```

- in an SM pubset environment:

```
//MODIFY-SM-PUBSET-PARAMETERS -
//  SM-PUBSET-ID=<sm-pubset-id>, -
//  MIGRATION-CONTROL=*PARAMETERS(FILE-INHIBIT=*IGNORED)
```

By default, HSMS respects the migration inhibit.

5.9.1.4 Except file

Migration of files – particularly to the offline background level S2 – can result in restricted access to this file. This is not always useful or desirable. The BS2000 data management system and HSMS therefore offer several options for exempting files from migration. In particular, files which are needed for starting BS2000 must be exempted from migration, since HSMS has not yet been loaded at this point in time and the files cannot be recalled.

Important system files are supplied with a migration inhibit in their catalog entry (be sure to copy only with COPY-FILE-PROTECTION=*SAME). Customer files can also be protected with a migration inhibit. In respecting the migration inhibit, however, HSMS makes no distinction between different users (see section "Allowing migration").

Files such as the parameter file CMDFILE can be reliably protected against migration by entering these files (or, to be on the safe side, the entire TSOS user ID) in an except file in order to exempt them from migration.

To do this the HSMS administrator lists the files to be exempted from migration in a SAM file, record by record (uppercase characters only). Entire user IDs (such as TSOS) can be exempted from migration by specifying partially-qualified file names. Wildcards can be used to define the files.

The except file is assigned by specifying:

- in the SF pubset environment:

```
//MODIFY-HSMS-PARAMETERS -
// MIGRATION-CONTROL=*PARAMETERS(EXCEPT-FILE=<file name>)
```

- in an SM pubset environment:

```
//MODIFY-SM-PUBSET-PARAMETERS -
//  SM-PUBSET-ID=<sm-pubset-id>, -
//  MIGRATION-CONTROL=*PARAMETERS(EXCEPT-FILE=<file name>)
```

5.9.1.5 Controlling implicit recall

The HSMS administrator determines whether implicit recall is permitted for files migrated to S2. This is done by means of the statement:

```
//MODIFY-HSMS-PARAMETERS -
// MIGRATION-CONTROL=*PAR(RECALL-FROM-S2=...)
```

This relates to SM and SF pubset environments.

There are three possible settings:

- *ALLOWED generally permits implicit recall.
- If *NOT-ALLOWED is set, the file must be retrieved to the processing level with an HSMS statement before it can be processed. Calls by the data management system are rejected.
- If *BATCH-ONLY is set, the files are automatically recalled for batch requests. In dialog requests, they must be recalled with an HSMS statement.

5.9.1.6 Monitoring migration / recall with openSM2

HSMS provides an interface to the software monitor openSM2 for improving the migration procedure. The SM2 monitoring system supplies statistics on the performance of the DP system and the utilization of the resources. During a monitoring period, SM2 regularly – say, every minute – records HSMS information and outputs it to a global SM2 file. All this HSMS information can be accessed by SM2 users, who can analyze some general parameters and follow their development. The users can analyze some general parameters and follow their development. For more detailed information on openSM2, refer to the “openSM2” manual [17].

A migration or recall request is monitored on the host on which it is executed, but only if it is started and ended within the same monitoring period. If in a collector request only one request is to be monitored, all the requests in the collector request are monitored nevertheless.

The following HSMS information is passed to SM2:

- Total number of mounted magnetic tapes for all migration and recall activities
- Total size of all migrated files (in PAM-PAGES)
- Number of migration runs
- Total number of extents of the migrated files
- Number of files migrated from S0 to S1
- Number of files migrated from S0 to S2
- Number of files migrated from S1 to S2
- Total number of days between the last time a file was used and its migration from S0 to S1
- Total number of days between the last time a file was used and its migration from S0 to S2
- Total number of days between the migration of a file to S1 and its migration to S2
- Number of recall runs (explicit recall runs and implicit recall events)
- Total number of days a file spent on S1 before it was recalled to S0
- Total number of days a file spent on S2 before it was recalled to S0
- Number of files that were explicitly recalled from S1
- Number of files that were explicitly recalled from S2
- Number of files that were implicitly recalled from S1
- Number of files that were implicitly recalled from S2
- Number of recall requests that have lasted the following length of time (in minutes): less than 2, 2 thru 4, 4 thru 6, 6 thru 8, 8 thru 10, 10 thru 12, 12 thru 14, 14 thru 16, 16 thru 18 and more than 18 minutes
- Number of recall requests that have waited the following length of time (in minutes) from generation to processing: less than 2, 2 thru 4, 4 thru 6, 6 thru 8, 8 thru 10, 10 thru 12, 12 thru 14, 14 thru 16, 16 thru 18 and more than 18 minutes
- Number of recall requests that have waited the following length of time (in minutes) from the start of their processing until the ARCHIVE call: less than 2, 2 thru 4, 4 thru 6, 6 thru 8, 8 thru 10, 10 thru 12, 12 thru 14, 14 thru 16, 16 thru 18 and more than 18 minutes
- Number of ARCHIVE processing operations for recall requests that have lasted the following length of time (in minutes): less than 2, 2 thru 4, 4 thru 6, 6 thru 8, 8 thru 10, 10 thru 12, 12 thru 14, 14 thru 16, 16 thru 18 and more than 18 minutes

- Number of magnetic tape cartridges for recall requests that were mounted for the following length of time (in minutes): less than 2, 2 thru 4, 4 thru 6, 6 thru 8, 8 thru 10, 10 thru 12, 12 thru 14, 14 thru 16, 16 thru 18 and more than 18 minutes

With SM2 and the collected information, the HSMS administrator can adjust the migration parameters appropriately, as the following examples show.

Example 1

- *Finding:*
  The number of files per migration run is low.
- *Reason:*
  The period of time between two migration runs is probably too short. The HSMS administrator is starting some unnecessary migration runs.

Example 2

- *Finding:*
  The average time before migration of a file from S0 is short.
- *Reason:*
  The amount of usable storage space on the disk is too small for the number of users. Either there are too many users or there are too many files which cannot be migrated.

Example 3

- *Finding:*
  The average dwell time on S1 or S2 is short.
- *Reason:*
  If, according to the migration procedure, all files which have been unused for X days are migrated, the number X has been set too low.
- *Remedy:*
  If a higher number is specified for X, the storage space used on S1 is increased, as is the length of time before a recall. The number of recalled files is reduced and the migration becomes more efficient overall.

Example 4

- *Finding:*
  The average time between generation of a recall request and the start of its processing is too long.
- *Reason:*
  The number of HSMS subtasks is probably too small.

Example 5

- *Finding:*
  The average time during which the magnetic tape cartridges are mounted is too long.
- *Reason:*
  Too many files are being recalled.

5.9.1.7 Restricting migration to files which can be migrated quickly (quick migration)

The HSMS administrator restricts migration to files which can be migrated immediately, i.e. without tape processing.

```
//MIGRATE-FILES ...(MIGRATION-INFO=*REMIGRATION)
```

From the set of files selected HSMS migrates only those files which have not been modified since the last recall and for which the backup file still exists. These files are immediately, i.e. without tape access, assigned only the reference to their existing backup file (S1 or S2). The storage space occupied is immediately released and no additional storage space is required on S1 or S2 for the migration.

The HSMS administrator can use this function to free storage space on a pubset or within a user ID immediately.

Furthermore, using quick migration means that reorganization from S2 to S2 is required less frequently.

5.9.2 Reorganizing the migration archive

Files which have already been recalled, overwritten or deleted can no longer be recalled and are therefore invalid in the migration archive. It is therefore necessary to reorganize the migration archive regularly to ensure that only valid files remain in the migration archive and all invalid files are deleted. Reorganizing the migration archive can also be a means of avoiding bottlenecks on volumes used for the migration archive.

However, the validity of a file is not indicated in the migration archive but must be checked on the basis of the catalog when the archive is accessed (i.e. also during reorganization). A file is “valid” if it still has a catalog entry which marks it a “migrated” file and if its CFID and version number are equivalent to those in the save file.

If the pubset catalog is unavailable at the time the check is carried out, its files are considered to be valid. During reorganization, all the files are accepted without comment, even if they are in fact invalid.

Summary of reusable storage space

When S1 is reorganized, a summary can be displayed to help the HSMS administrator. The summary shows how much storage space at S1 is occupied by save files only partially filled with valid files:

```
//SHOW-PUBSET-USAGE INFORMATION=*REUSABLE-S1-SPACE
```

For more information on this information option, see "HSMS Vol. 2” [1], HSMS statement SHOW-PUBSET-USAGE.

The HSMS administrator has four options for reorganizing the migration archive; which one is selected depends, among other things, on whether data is migrated to S1 or only to S2.

5.9.2.1 Migration from S1 to S1

The HSMS administrator can reorganize save files at level S1 by “migrating” them from S1 to S1:

```
//MIGRATE-FILES FROM-STORAGE=*S1-STORAGE-LEVEL(...,
                            TO-STORAGE=*S1-STORAGE-LEVEL,...)
```

In doing this, only the valid files are transferred to the new save file. If the save file was empty, i.e. if it contained only invalid files, it is deleted. No new save file is created.
After a successful migration operation, the old save file is always deleted, regardless of the stipulated retention period.

The HSMS administrator has the following options when migrating files from S1 to S1:

- He can use the MINIMUM-/MAXIMUM-DAYS-ON-S1 operand to restrict selection to save files of a certain age. For example, he can reorganize all save files which have been on level S1 for the last 30 days.
- He can use MINIMUM-SIZE to restrict migration to save files of a certain minimum size.
- With the UNUSED-SPACE operand the save files can be selected according to the percentage of invalid files they contain. For example, UNUSED-SPACE=60 means that only those save files which are at least 60% empty are reorganized.
  When UNUSED-SPACE=100 is specified, the empty save files are deleted.
- RELEASE-PAGES can be used to restrict the quantity of save files to be migrated until a definable number of PAM pages has been released by reorganization (as in migration from S0).
- No specific volume sets can be selected for backups, migration and reorganization on the S1 level defined by all volume sets under HSMS control.

  However, the HSMS administrator can restrict the volume set usage with the following command:

  ```
  /MODIFY-PUBSET-RESTRICTIONS PUBSET=<SM_pubset_catid>, -
  /       PUBSET-TYPE=*S-M(VOL-SET=<catid_of_volume_set>, -
  /       RESTRICTION = *NEW-FILE-ALLOCATION)
  ```

Note on reorganizing from S1 to S1 in an SM pubset environment

If the S1 level is defined by all volume sets under HSMS control, migration from S1 level to S1 can be carried out with SAVE-FILE-PROCESSING=*HSMS-V10-COMPATIBLE only in BS2000 OSD/BC V11.0A or higher.

5.9.2.2 Migration from S1 to S2

When migrating files from S1 to S2, the load on storage level S1 is reduced even further by deleting the save files on S1 after migration. This, however, increases access time and implicit recall in interactive mode is only possible during tape sessions.
The catalog entry of the migrated file is updated, i.e. the entries for DEVICE-TYPE and STORAGE-LEVEL are modified (PUB/S2).

Files are migrated to S2 via the following HSMS statement:

```
//MIGRATE-FILES FROM-STORAGE=*S1-STORAGE-LEVEL(...,
                            TO-STORAGE=*S2-STORAGE-LEVEL,...)
```

In doing this, the HSMS administrator has the same options as for migration from S1 to S1.

In particular, we recommend migrating files via MINIMUM-DAYS-ON-S1, since the probability that a file will still be required for processing decreases over time.

Regardless of the settings of the SAVE-FILE-PROCESSING option at file creation on S1 level, this file can be migrated from S1 to S2 with any setting of the SAVE-FILE-PROCESSING option.

5.9.2.3 Reorganization of a migration archive in S2

The HSMS administrator can also reorganize save files with invalid files on storage level S2 using the MIGRATE-FILES statement:

```
//MIGRATE-FILES FROM-STORAGE=*S2-STORAGE-LEVEL(...)
```

When a save file is reorganized a new save file is created and the original save file is automatically deleted. Several save files can be reorganized in a single call.

The HSMS administrator has the following options when reorganizing on S2:

- With the SAVE-FILE-ID operand the administrator can specify a particular save file or restrict the selection to save files in a specific creation period. The default setting is the selection of all save files on S2, with the exception of the current standard save file, which is never reorganized.
- With the UNUSED-SPACE operand the save files can be selected according to the percentage of invalid files they contain. For example, UNUSED-SPACE=60 means that only those save files which are at least 60% empty are reorganized.
  When UNUSED-SPACE=100 is specified, the empty save files are deleted.

5.9.2.4 Copying the save files

Save files containing invalid files on level S2 can also be copied by using the COPY-SAVE-FILE statement:

```
//COPY-SAVE-FILE ...,MIGRATION-STATE=*MIGRATED-FILE
```

In this case as well, only valid files are accepted.

The original save file is not automatically deleted after copying; it must be released by means of //MODIFY-ARCHIVE. It is also recommended not to archive save files with the same content over a longer period of time. Older save files should in this case be deleted as soon as possible, otherwise redundant save versions of the migrated files will remain in one and the same save file in subsequent reorganizations with //MIGRATE-FILES.

During a reorganization carried out using //COPY-SAVE-FILE, only a single save version is created in the target save file, even if the original save file contains more than one save version. The effect of this is to reduce the size of the directory file associated with the migration archive and to reduce the subsequent processing time.

If a backup copy of a save file in a migration archive was created, the data is recalled from the copy in the following cases:

- Recall from the last copy by an HSMS operation, regardless of the storage level on which it resides
- Implicit recall from the last copy on the storage level indicated in the catalog entry for the file.

Of course, save files on S1 can likewise be copied via //COPY-SAVE-FILE.

But in any case, reorganization using the MIGRATE-FILES statement provides better options for control and selection.

5.9.3 Data backup and migrated files

As discussed in section “Saving BS2000 files and job variables”, not only catalog entries (save type MIGF) but also the data of migrated files can be saved during backups.

If the data from migrated files is to be included in the backup, it is not advisable to place the backup of files migrated to S1 on the S1 pubset. If this pubset were to fail, it would affect the data and the backup of that data. However, the BACKUP-FILES statement supports the TO-STORAGE operand, which provides a mechanism with which to perform a backup to any pubset in this particular situation.

During the restore, the HSMS administrator can specify whether just the catalog entries or the catalog entries and the data of the migrated files should be copied to storage level S0. It is also possible to restore the data of the migrated files to a migration level.

5.9.3.1 Inconsistent files

If //SHOW-PUBSET-USAGE INFORMATION=*MIGRATION-EVALUATION is specified, the HSMS administrator can determine whether such inconsistencies have occurred and the pubsets on which they occurred. The files which have been marked as migrated files in the catalog but which are not located on S1 or in the migration archive are output, organized according to pubset and user ID.

5.9.3.2 Correcting inconsistencies in migrated files

Data is taken during a file recall from the save file of the S1 or S2 storage level on which it was placed during the last migration, secondary migration or copy operation performed within the migration archive. This save file is registered in the localization information contained in the file’s catalog entry. Other save files in the migration archive are therefore only interpreted during a recall as being repositories for the data of migrated files.

If the save file registered in the localization information of a migrated file’s catalog entry does not exist, or the save file on the respective storage level no longer exists, it constitutes a migration file inconsistency. This also means that all files are inconsistent because their respective catalog entries contain no localization information.

Inconsistencies in migrated files can be displayed with the aid of the SHOW-PUBSET-USAGE statement’s INFORMATION=*MIGRATION-EVALUATION operand. This display function also takes into account the above-mentioned definition of inconsistency. This means that operator error may cause the system to indicate that a file is inconsistent when that file is still capable of being recalled. This situation can arise if the save file registered in the catalog entry exists but is no longer registered in the migration archive.

The HSMS statements REPAIR-CATALOG-BY-EXCHANGE and REPAIR-CATALOG-BY-RESTORE provide a means of correcting inconsistencies in migrated files.

5.9.4 Examples

This section provides examples dealing with the following subjects:

- Controlling migration
- Migration from S1 to S2
- Deletion of empty save files
- Reorganizing save files on S1
- Reorganizing migrate save files on S2
- Employing the SHOW-PUBSET-USAGE statement (with output of reusable storage space and inconsistencies).

The examples are based on the HSMS configuration described in "Creating an HSMS configuration (example)".

Controlling migration

The HSMS parameters are set so as to permit migration by the HSMS administrator only. However, files can be recalled by nonprivileged users also.

```
//MODIFY-ARCHIVE-ATTRIBUTES ARCH-NAME=$SYSHSMS.HSMS.MIG.Y, - ————————————  (1)
//  USER-ACCESS=*ALL-USERS(ACCESS=*READ)
% HSM0003 HSMS STATEMENT COMPLETED

//MODIFY-HSMS-PARAMETERS -
//  MIGR-CONTROL=*PAR(RECALL-FROM-S2=*ALLOWED)  ————————————————————————  (2)
% HSM0003 HSMS STATEMENT COMPLETED
```

(1) Only read access is permitted for the migration archive, i.e. only the HSMS administrator, who is the archive owner, can migrate files to this archive. Nonprivileged users can recall files from this archive.

(2) The implicit recall of files from S2 is permitted.

Migration from S1 to S2

```
//MIGRATE FILES -
//  FROM-STOR=*S1-STOR(S1-PUB-ID=2BC, -
//    ARCH-NAME=$SYSHSMS.HSMS.MI.2BY,MIN-DAYS-ON-S1=65), -  ————————————  (1)
//  OPER-CONTROL=*PAR(REPORT=*NONE)
% HSM0003 HSMS STATEMENT COMPLETED
```

(1) All files which have been on S1 for at least 65 days are migrated to S2; one can assume that migrated files which are used from time to time have already been recalled to S0 at some point. One can assume that migrated files which are used from time to time have already been recalled to S0 at some point. This means that only those files which were not accessed for at least the last 65+28 days (migration to S1) must be recalled from S2. All other files remain on S1.

Deletion of empty save files

```
//MIGRATE-FILES -  ————————————————————————————————————————————  (1)
//  FROM-STOR=*S1-STOR(S1-PUB-ID=2BC,UNUSED-SPACE=100, -
//    TO-STOR=*S1-STOR), -
//  OPER-CONTROL=*PAR(REPORT=*FULL, -
//    OUT=HSMS.MAN.R.MGF.5,WAIT-F-C=*YES)
% HSM0003 HSMS STATEMENT COMPLETED
...
Output from the report HSMS.MAN.R.MGF.5  ————————————————————————  (2)
```

(1) All save files in the system migration archive which contain only invalid files, i.e. which are considered empty, are migrated and thus deleted.

(2) The last page of the report has a different header, namely `DELETED SAVE VERSIONS`. Here, HSMS indicates those save files which have not only been reorganized but deleted as a result of migration.

Reorganizing save files on S1

```
//MIGRATE-FILES -  ————————————————————————————————————————————  (1)
//  FROM-STOR=*S1-STOR(S1-PUB-ID=2BC,TO-STOR=*S1-STOR), -
//  OPER-CONTROL=*PAR(REPORT=*FULL,OUT=HSMS.MAN.R.MGF.4, -
//    WAIT-F-C=*YES)
% HSM0003 HSMS STATEMENT COMPLETED
```

(1) The save files of the system migration archive on S1 are reorganized. All save files containing invalid files are converted (RELEASE-PAGES=*MAXIMUM is the default value).

Reorganizing migrate save files on S2

The $SYSHSMS.A.MIGRATE archive contains 2 save files and the standard save file:

```
SHOW-ARCHIVE (SAVE-FILES)             INFORMATION       = SUMMARY
ENVIRONMENT     = SF                  ARCHIVE-NAME      = $SYSHSMS.A.MIGRATE
SAVE-FILE-STATE = ANY                 SAVE-FILE-STORAGE = ANY
CREATED-BEFORE  = LATEST              EXPIRATION-BEFORE = LATEST
------------------------------------------------------------------------------
M SFID             CREA-DATE  EXP-DATE  OBS  ACCESS ST  DEVICE    #VOL #SV #RUNS
  S.160521.130622  16-05-21   16-05-21  YES  OWNER  TAP TAPE-C4   1    1    1
  S.160521.133641  16-05-21   16-05-21  YES  OWNER  TAP TAPE-C4   1    1    1
  S.160521.135324  16-05-21   16-05-21  YES  OWNER  TAP TAPE-C4   1    1    1









------------------------------------------------------------------------------
NEXT-PAGE : +   (+, -, ++, --, E)
%  HSM0012 END OF OUTPUT LIST REACHED
```

The save files from before the standard save file is changed are to be reorganized using the MIGRATE-FILES statement:

```
//MIGRATE-FILES -
//  FROM-STOR=*S2-STOR(SAVE-FILE-ID=*ALL, -
//    UNUSED-SPACE=*ANY,ARCHIVE-NAME=*SYSMIGRATE), -
//  OPER-CONTROL=*PAR(OUT=PROT.S2REO)
```

Output of the log:

```
***  MIGRATE - FILES                HSMS V12.0      SUMMARY          REPORT  *** 2016-05-
21  14:34:27
PAGE    1
    REQUEST-ENVIRONMENT=SF
    REQUEST-NAME=MGF#0015 REQUEST-DATE=2016-05-21 14:20:28 USER-ID=SYSHSMS  REQUEST-
STATE=COMPLETED   WITH
WARNINGS
STATEMENT LISTING:
MGF         FROM-STORAGE=*S2-STORAGE-LEVEL,OPERATION-CONTROL=*PARAMETERS(OUTPUT=PROT.S2REO)
           ENVIRONMENT                              : SF
           ARCHIVE-NAME                             : $SYSHSMS.A.MIGRATE
           SAVE-FILE ATTRIBUTES
           TO-STORAGE                               : S2-STORAGE-LEVEL
           DEVICE-TYPE                              : TAPE-C4
           RETENTION-PERIOD                         : 0
           SAVE-VERSION ATTRIBUTES
            SAVE-VERSION-NAME                        : MIGRATE
***  MIGRATE - FILES                HSMS V12.0      SUMMARY           REPORT  *** 2016-05-
21  14:34:27
PAGE  2
    REQUEST-ENVIRONMENT=SF
    REQUEST-NAME=MGF#0015 REQUEST-DATE=2016-05-21 14:20:28 USER-ID=SYSHSMS  REQUEST-
STATE=COMPLETED   WITH
WARNINGS
%  ARC0002 STATEMENT ACCEPTED. ARCHIVE SEQUENCE NUMBER 'A.160521.142625', VERSION '12.0'
%  HSM0476 SAVE FILE 'S.160521.130622' DELETED DURING REORGANIZATION
%  ARC0002 STATEMENT ACCEPTED. ARCHIVE SEQUENCE NUMBER 'A.160521.142747', VERSION '12.0'
%  ARC0096 MIGRATED FILE ':A:$TSOS.TTT1' NOT COPIED
%  ARC0033 ARCHIVE SUBTASK TSN '0AAL' GENERATED
%  ARC0815 SUBTASK '0' HAS TRANSFERRED '3' PAM PAGES FOR '1' FILES AND '0' JVS IN '10'
SECONDS
%  HSM0475 SAVE FILE 'S.160521.133641' REORGANIZED INTO SAVE FILE 'S.160521.142747'
%  ARC0002 STATEMENT ACCEPTED. ARCHIVE SEQUENCE NUMBER 'A.160521.143149', VERSION '12.0'
                                 SAVE FILE IDENTIFIER - S.160521.142747
          SUBSAVE
           NUMBER       VSNS
              0        TAPE04
***      E N D   O F                HSMS V12.0      SUMMARY           REPORT  *** 2016-05-
21  14:34:27
***
```

Save file S.160521.130622 no longer contained any valid files and was therefore deleted.
Save file S.160521.133641 was reorganized into save file S.160521.142747.

```
SHOW-ARCHIVE (SAVE-FILES)             INFORMATION       = SUMMARY
ENVIRONMENT     = SF                  ARCHIVE-NAME      = $SYSHSMS.A.MIGRATE
SAVE-FILE-STATE = ANY                 SAVE-FILE-STORAGE = ANY
CREATED-BEFORE  = LATEST              EXPIRATION-BEFORE = LATEST
------------------------------------------------------------------------------
M SFID             CREA-DATE  EXP-DATE  OBS  ACCESS ST  DEVICE    #VOL #SV #RUNS
  S.160521.135324  16-05-21   16-05-21  YES  OWNER  TAP TAPE-C4   1    1    1
  S.160521.142747  16-05-21   16-05-21  YES  OWNER  TAP TAPE-C4   1    1    1
------------------------------------------------------------------------------
NEXT-PAGE : +   (+, -, ++, --, E)
%  HSM0012 END OF OUTPUT LIST REACHED
```

Use of the SHOW-PUBSET-USAGE statement

Files are migrated, some of them are subsequently deleted. The reusable space to be gained by reorganization is displayed. Moreover, an inconsistency created in the migration archive is output.

```
//START-HSMS
//MIGRATE-FILES -  ————————————————————————————————————————————  (1)
//  FROM-STOR=*S0-STOR(F-NAMES=$MANUAL.FILE.*7*,TO-STOR=*S1-STOR) -
//  OPER-CONTROL=*PAR(REPORT=*FULL,OUT=HSMS.MAN.R.MGF.6, -
//    WAIT-F-C=*YES)
% HSM0003 HSMS STATEMENT COMPLETED
Report output HSMS.MAN.R.MGF.6  ————————————————————————————  (2)

.
Migration of further files, FILE.*6*, FILE.*5*, FILE.*4*, FILE.* ————  (3)
.
//BACKUP-FILES F-NAMES=$MANUAL., -  ——————————————————————————————  (4)
//  TO-STOR=*S1-STOR, -
//  OPER-CONTROL=*PAR(REPORT=*FULL,OUT=HSMS.MAN.R.BCF.7, -
//    WAIT-F-C=*YES)
% HSM0003 HSMS STATEMENT COMPLETED
Report output HSMS.MAN.R.BCF.7  ————————————————————————————  (5)

//END
% HSM0014 HSMS PROGRAM TERMINATED
/DELETE-FILE FILE-NAME=$MANUAL.FILE.*1*  ———————————————————————  (6)
/DELETE-FILE FILE-NAME=$MANUAL.FILE.*2
//SHOW-PUBSET-USAGE PUB-ID=2BC,INF=*REUS-S1-SPACE  ——————————————  (7)
```

```
SHOW-PUBSET-USAGE         PUBSET-ID = 2BC        INFORMATION = REUSABLE-S1-SPACE
MINIMUM-SIZE = NONE       MINIMUM-DAYS-ON-S1 = 0      MAXIMUM-DAYS-ON-S1 = 9999
ARCHIVE-NAME = *SYSMIGRATE
------------------------------------------------------------------------------
PUBSET: 2BC       CAPACITY:    663345        %USED:  84.6       %AVAIL:  15.4
------------------------------------------------------------------------------
% UNUSED-SPACE            #SAVE-FILES              #PAGES       #UNUSED-PAGES
       = 100                        0                   0                   0
    90 - 100                        0                   0                   0
    80 -  90                        0                   0                   0
    70 -  80                        0                   0                   0
    60 -  70                        1                  82                  53
    50 -  60                        0                   0                   0
    40 -  50                        0                   0                   0
    30 -  40                        4                 175                  57
    20 -  30                        0                   0                   0
    10 -  20                        0                   0                   0
    00 -  10                        0                   0                   0
       =  00                        0                   0                   0
------------------------------------------------------------------------------
       TOTAL                        5                 257                 110

------------------------------------------------------------------------------
NEXT-PAGE : __  (+, -, ++, --, E)
```

```
% HSM0003 HSMS STATEMENT COMPLETED
//MIGRATE-FILES-  ──────────────────────────────────────────────────────  (8)
//  FROM-STOR=*S1-STOR(S1-PUB-ID=2BC,TO-STOR=*S1-STOR), -
//  OPER-CONTROL=*PAR(REPORT=*FULL,OUT=HSMS.MAN.R.MGF.7, -
//    WAIT-F-C=*YES)
% HSM0003 HSMS STATEMENT COMPLETED
Report HSMS.MAN.R.MGF.7 (exctract):  ──────────────────────────────────  (9)

 ***    MIGRATE - FILES               HSMS V12.0        FULL         REPORT  *** 2016-08-12
14:54:39    PAGE    2
     REQUEST-ENVIRONMENT=SF
     REQUEST-NAME=MGF#0AAK REQUEST-DATE=2016-08-12 14:52:48 USER-ID=SYSHSMS  REQUEST-
STATE=COMPLETED WITH WARNINGS
 %  ARC0002 STATEMENT ACCEPTED. ARCHIVE SEQUENCE NUMBER 'A.160812.145256', VERSION='12.0'
 %  ARC0096 MIGRATED FILE ':2BY:$MANUAL.FILE.17' NOT COPIED
 %  ARC0033 ARCHIVE SUBTASK TSN '0ACQ' GENERATED
 %  HSM0475 SAVE FILE 'S.160812.145048' REORGANIZED INTO SAVE FILE 'S.160812.145256'
 %  ARC0002 STATEMENT ACCEPTED. ARCHIVE SEQUENCE NUMBER 'A.160812.145314', VERSION='12.0'
 %  ARC0002 STATEMENT ACCEPTED. ARCHIVE SEQUENCE NUMBER 'A.160812.145317', VERSION='12.0'
 %  ARC0096 MIGRATED FILE ':2BY:$MANUAL.FILE.16' NOT COPIED
 %  ARC0033 ARCHIVE SUBTASK TSN '0ACR' GENERATED
 usw.
 %  HSM0475 SAVE FILE 'S.160812.145148' REORGANIZED INTO SAVE FILE 'S.160812.145358'
 %  ARC0002 STATEMENT ACCEPTED. ARCHIVE SEQUENCE NUMBER 'A.160812.145415', VERSION='12.0'
 %  ARC0002 STATEMENT ACCEPTED. ARCHIVE SEQUENCE NUMBER 'A.160812.145418', VERSION='12.0'
 %  ARC0096 MIGRATED FILE ':2BY:$MANUAL.FILE.01' NOT COPIED
 %  ARC0096 MIGRATED FILE ':2BY:$MANUAL.FILE.02' NOT COPIED
 %  ARC0096 MIGRATED FILE ':2BY:$MANUAL.FILE.11' NOT COPIED
 %  ARC0096 MIGRATED FILE ':2BY:$MANUAL.FILE.12' NOT COPIED
 %  ARC0096 MIGRATED FILE ':2BY:$MANUAL.FILE.13' NOT COPIED
 %  ARC0096 MIGRATED FILE ':2BY:$MANUAL.FILE.21' NOT COPIED
 %  ARC0096 MIGRATED FILE ':2BY:$MANUAL.FILE.22' NOT COPIED
 %  ARC0033 ARCHIVE SUBTASK TSN '0ACU GENERATED
 %  HSM0475 SAVE FILE 'S.160812.145208' REORGANIZED INTO SAVE FILE 'S.160812.145418'
 %  ARC0002 STATEMENT ACCEPTED. ARCHIVE SEQUENCE NUMBER 'A.160812.145436', VERSION='12.0'
 %  HSM0473 REORGANIZATION COMPLETED. '114' PAGES ON S1 STORAGE RELEASED
```

```
//RESTORE-FILES -
//  F-NAMES=$MANUAL.FILE., -
//  REPLACE-FILES-AND-JV=*NO, -
//  OPER-CONTROL=*PAR(REPORT=*FULL,OUT=HSMS.MAN.R.RSF.7, -
//    WAIT-F-C=*YES)
% HSM0003 HSMS STATEMENT COMPLETED
Report HSMS.MAN.R.RSF.7 (extract):

***   RESTORE - FILES           HSMS V12.0     FULL      REPORT  *** 2016-08-12  14:54:48
PAGE    3
    REQUEST-ENVIRONMENT=SF
    REQUEST-NAME=RSF#0AAK REQUEST-DATE=2016-08-12 14:54:40 USER-ID=SYSHSMS  REQUEST-
STATE=COMPLETED WITH WARNINGS

                      ***   CATALOG - 2BY       USER - MANUAL     ***
           FILE/JOB VARIABLE NAME                   LASTPG/   SAVE VERSION  SAVE  INPUT   SUB
OUTPUT
                                                  VERS     SIZE    IDENTIFIER   TYPE   VSN   SAVE
DISK(S)

FILE.03                                             1       13  160812.145229  MIGF  0:2BC    0
%  ARC0035 FILE TO BE RESTORED ALREADY EXISTS. FILE NOT REPLACED
FILE.04                                             1       18  160812.145229  MIGF  0:2BC    0
%  ARC0035 FILE TO BE RESTORED ALREADY EXISTS. FILE NOT REPLACED
usw.
FILE.27                                             1        6  160812.145229  MIGF  0:2BC    0
%  ARC0035 FILE TO BE RESTORED ALREADY EXISTS. FILE NOT REPLACED
FILE.01                                             1        3  160812.145229  MIGF  0:2BC    0
FILE.02                                             1        8  160812.145229  MIGF  0:2BC    0
FILE.11                                             1        3  160812.145229  MIGF  0:2BC    0
FILE.12                                             1        8  160812.145229  MIGF  0:2BC    0
usw.
FILE.22                                             1        8  160812.145229  MIGF  0:2BC    0

***      E N D   O F                 HSMS V12.0        FULL         REPORT  *** 2016-08-12
14:54:48     ***
//END
```

```
% HSM0014 HSMS PROGRAM TERMINATED
/DELETE-FILE FILE-NAME=:2BC:$TSOS.ARCHIVE.SAVE.FILE. -  ——————————————— (10)
/  160812.145256.0
//SHOW-PUBSET-USAGE PUB-ID=2BY,INF=*MIG-EVAL ———————————————————————— (11)
% HSM0433 DMS ERROR '0333' DURING 'CATALOG-'ACCESS TO FILE
':2BC:$TSOS.ARCHIVE.SAVE.FILE.160812.145256.0'
```

```
SHOW-PUBSET-USAGE                              INFORMATION = MIGRATION-EVALUATION
S0-PUBSET: 2BY                                 S1-PUBSET: 2BC   USER-ID: MANUAL
------------------------------------------------------------------------------
INCONSISTENT FILE                                                    ERROR
FILE.01                                                              NOT-IN-ARC
FILE.02                                                              NOT-IN-ARC
FILE.07                                                              NO-S1-DATA
FILE.11                                                              NOT-IN-ARC
FILE.12                                                              NOT-IN-ARC
FILE.13                                                              NOT-IN-ARC
FILE.14                                                              NOT-IN-ARC
FILE.15                                                              NOT-IN-ARC
FILE.16                                                              NOT-IN-ARC
FILE.17                                                              NOT-IN-ARC
FILE.21                                                              NOT-IN-ARC
FILE.22                                                              NOT-IN-ARC
FILE.27                                                              NO-S1-DATA




------------------------------------------------------------------------------
NEXT-PAGE : __  (+, -, ++, --, E)
```

```
% HSM0003 HSMS STATEMENT COMPLETED
//END
% HSM0014 HSMS PROGRAM TERMINATED
```

(1) The files named FILE.*7* of user ID MANUAL are migrated to the system migration archive.

(2) The HSMS-generated report of the migration run is output. Among other things, it also contains the SFID of the migration run

(3) Further files of user ID MANUAL are migrated. This is done in separate migration runs so that different save files are created for the example.

(4) The migrated files are saved to the system backup archive. Since the SAVE-OPTIONS operand is omitted, the default value for the system backup archive applies.

(5) The HSMS-generated report is output. In accordance with the default setting, only the catalog entries of the migrated files have been saved (backup type MIGF).

(6) Some of the previously migrated and subsequently saved files are deleted on the processing level and thus become invalid.

(7) Output via SHOW-PUBSET-USAGE shows that some save files contain files that have become invalid through deletion of the respective catalog entries, and how many pages could be saved by reorganization.

(8) The migration within the S1 level reorganizes all save files on pubset 2BC.

(9) Message HSM0475 in the HSMS-generated report indicates the transferred save files. The files not copied because they are invalid (deleted) are listed with message ARC0096.
The save file S.160812.145048, for example, is written to the save file S.160812.145256. In this process FILE.17 is not copied because it is invalid. The save file still contains the files FILE.07 and .27.
HSM0473 is the concluding message indicating the number of pages released.

(10) All files previously saved are recalled to the processing level without overwriting any existing files.

(11) The report shows that the existing files have triggered warnings. The previously deleted files, on the other hand, have been recalled to the processing level.

(12) The save file containing files FILE.07 and .27 after reorganization (see above) is deleted (for demonstration purposes).

(13) As a consequence, SHOW-PUBSET-USAGE now indicates inconsistencies:
the previously deleted files were not transferred during reorganization and are therefore shown as NOT-IN-ARC.
The files contained in the deleted save file are no longer in a save file on S1 and are output with NO-S1-DATA.

5.10 Other control parameters

- Number of server tasks
- Common memory size
- Wait times for synchronous requests and for reservation
- Processing mode for save files
- Central request monitoring on the SE server
- Keep requests

5.10.1 Number of server tasks

HSMS provides a number of server tasks for processing requests simultaneously. The HSMS administrator can define the maximum number of server tasks permanently using the NUMBER-OF-SUBTASKS operand of the HSMS statement MODIFY-HSMS-PARAMETERS. The system administrator can change the value temporarily for the current HSMS session using the same operand of the START-HSMS statement.

At least one server task per default system archive should be available, with additional server tasks for requests concerning private archives. In addition, there should also be server tasks for read requests to S1. In addition, there should also be server tasks for read requests to S1. The number of server tasks in addition to those per system archive and for requests to S1 should correspond to the number of tape units available.

Server tasks are not automatically restarted following a program crash.
This means that the number of server tasks active at a given time may be smaller than desired, with the result that a request is not processed. If the situation arises in which requests are accepted but not started, the HSMS administrator should check how many server tasks are active (HSMS statement SHOW-HSMS-PARAMETERS). The HSMS administrator can use the HSMS statement MODIFY-HSMS-PARAMETERS to create new server tasks, if necessary.

5.10.2 Common memory size

HSMS makes use of common memory pools. The HSMS administrator can define the size of these common memory pools via the COMMON-MEMORY-SIZE operand of the HSMS statement MODIFY-HSMS-PARAMETERS, while the system administrator can use the same operand of the START-HSMS statement for this purpose.

If HSMS performs an internal calculation during loading and discovers that the size defined in MODIFY-HSMS-PARAMETERS is not large enough, it automatically creates the memory pool with the calculated size. Note that, in contrast to this, HSMS does not correct the value defined for the current HSMS session by means of START-HSMS.

SHOW-HSMS-PARAMETERS VALID-PERIOD=*SESSION causes HSMS to output the currently valid value, i.e. the corrected value, if applicable.

The value calculated by HSMS will usually be sufficiently large. Note, however, that bottlenecks may occur in cases where the computer center-specific application requires values that are markedly different from the parameters on which HSMS calculations are based. Such bottlenecks may, for instance, render the creation of archives impossible.

Situations of this sort will be reported by HSMS. When they arise, the HSMS administrator should increase the value for the common memory pool. The new value does not take effect, however, until the next HSMS session.

When selecting the value it should be remembered that the size of the common memory pool limits the user address space.

5.10.3 Wait times for synchronous requests and for reservation

In synchronous processing, maximum wait times apply: HSMS statements are rejected if HSMS fails to accept the request in question before the respective wait time has elapsed. If the request has not been completed, processing of the statement continues asynchronously.

The maximum wait times are defined by the HSMS administrator by means of

```
//MODIFY-HSMS-PARAMETERS REQUEST-WAIT-LIMITS=*PARAMETERS(...)
```

This statement serves to define:

- a maximum wait time for acceptance of the request (DIALOG- or BATCH-REQUEST-TIME) separately for interactive and batch requests.
- a maximum wait time for request execution (DIALOG- or BATCH-EXEC-TIME) separately for interactive and batch requests.

If the request is not terminated after the maximum wait time defined with DIALOG- or BATCH-REQUEST-TIME for request acceptance has elapsed, a check is made to determine whether the request has already been accepted by an HSMS server task. How processing continues is dependent on the result of this check:

- If the request has already been accepted, the system waits for the request to be executed. Here the maximum wait time for request execution applies which was defined with DIALOG- or BATCH-EXEC-TIME.
- If the request has not yet been accepted, it is rejected.

Note

HSMS issues a SECURE-RESOURCE-ALLOCATION command with wait time for the input and output volumes. This wait time is the same as the specification in the HSMS parameter BATCH-REQUEST-TIME (see also section "Volume and device reservation").

5.10.4 Processing mode for save files

In HSMS V10.0A and higher, the global parameter SAVE-FILE-PROCESSING can be used to define the processing mode for save files at storage level S1, on shared disk, and on Net-Storage: *HSMS-V9-COMPATIBLE or *HSMS-V10-COMPATIBLE.

Save files generated in processing mode *HSMS-V9-COMPATIBLE can be processed without restriction with HSMS versions lower than V10.0A. However, save files generated in processing mode *HSMS-V10-COMPATIBLE can only be processed with HSMS versions lower than V10.0A if the files are on tape or private disk. The reason for this is that save files on shared disk and Net-Storage which were created in *HSMS-V10-COMPATIBLE mode have a different name structure than the files created in *HSMS-V9-COMPATIBLE mode.

*HSMS-V9-COMPATIBLE mode is set by default. However, the full functionality of HSMS is not available in this mode. If you want to use the full functionality of HSMS, you must explicitly specify *HSMS-V10-COMPATIBLE mode.

The table below provides an overview of the functions available depending on the setting of the SAVE-FILE-PROCESSING parameter:

Functionality	**SAVE-FILE-PROCESSING**	
	***HSMS-V9-COMPATIBLE**	***HSMS-V10-COMPATIBLE**
Save file names independent of the storage location (in the case of shared disk and Net-Storage)	-	+
Use of S1-SM pubsets	-	+
Distributing save files to multiple volume sets within an S1-SM pubset	-	+
Sequence number in save file names (when using S1-SM pubsets)	-	+
Continuing save files created with HSMS versions < V10.0 on shared disk or Net-Storage	+	-
Read access to save files created with HSMS versions < V10.0	+	+
Access with HSMS versions < V10.0 to shared files on public disk or Net-Storage which were created with HSMS V10.0	+	-
Distribution of the save files to multiple volume sets in an SM environment when storage level S1 is defined as *ALL-HSMS-CONTROLLED	-	+

\+ Function is available

\- Function is not available

5.10.5 Central request monitoring on the SE server

The BS2000 backup monitor is integrated in the SE Manager's main menu **Applications** as an SE management application. The BS2000 backup monitor provides information on the status of the backup requests which were assigned in the SE server's BS2000 systems with the software products HSMS and FDDRL. For requests which have terminated, HSMS provides a request log in PDF format for the BS2000 backup monitor.

The following requests are monitored:

- Backup
- Restoring data
- Migration (HSMS only)
- Archival (HSMS only)
- Copying save files, including archive reorganization runs (HSMS only)
- Exporting and importing files (HSMS only)
- Backup files version (HSMS only)
- Reorganization of version backup (HSMS only)

The display of a BS2000 system's HSMS requests is controlled using the following settings:

- Globally using the HSMS parameter MONITORING (MODIFY-HSMS-PARAMETERS statement)
- On an archive-specific basis using the archive attribute MONITORING (CREATE-ARCHIVE or MODIFY-ARCHIVE-ATTRIBUTES statement)

When combined, these settings have the following effect:

- MONITORING=*ALL is set globally: monitoring is enabled globally for all requests regardless of the archive-specific setting.
- MONITORING=*BY-ARCHIVE-ATTRIBUTES is set globally: only requests which relate to archives with the setting MONITORING=*STD are displayed.
- MONITORING=*SYSHSMS-ONLY is set globally: only requests which relate to archives under the user ID SYSHSMS are displayed. Requests which do not concern any archives (IMPORT-FILES and EXPORT-FILES) are displayed only if they were started under a user ID with the HSMS-ADMINISTRATOR privilege.
- MONITORING=*NONE is set globally: monitoring is disabled globally for all requests regardless of the archive-specific setting.

Log in PDF file

At the end of the HSMS run, a log file with the following name is generated in PDF format:

```
$SYSHSMS.HSM.C.SYSHSMS.<date_time>.PDF
```

When the request is deleted, the PDF file is also deleted. Only the system can access this file.

If the request is restarted with //RESTART-REQUEST, the log file and thus the PDF file is created anew.

In HSMS V11.0 and higher, the log file is created in PDF format via the CONV2PDF subsystem.
If an error occurs or the subsystem is not available, HSMS creates the PDF file as previously using the $DHSCFTP procedure from the SYSPRC.HSSM.<version> library. Only in this case a temporary working file is necessary (see also “Temporary files” in "Work files").

5.10.6 Keep requests

HSMS provides functionality for keeping completed requests after a host crash or restart. Requests with status COMPLETED are kept based on the value of KEEP-REQUESTS. The value of KEEP-REQUEST determines the amount of days that completed request will be kept. Older completed requests will be deleted. The default value is *STD, that means 40 days. The system administrator can change the value temporarily for the current HSMS session using MODIFY-HSMS-PARAMETERS. However, this will not take effect, because the deletion of completed requests will be done at HSMS subsystem startup only. The value must be changed permanently in the control file via MODIFY-HSMS-PARAMETERS statement. The table below provides an overview of the possible parameter values:

Value	**Functionality** (only for completed requests and at every HSMS subsystem startup, corresponding PDF report files on SE server will be deleted as well)
***STD** (default value) or **40**	Requests will be deleted automatically after 40 days
***NO** or **0**	All requests will be deleted automatically
***YES**	Requests will never be deleted automatically
Integer <**0**..**32767**>	Requests older than <integer value> days will be deleted automatically

The value of KEEP-REQUESTS is defined by the HSMS administrator by means of

//MODIFY-HSMS-PARAMETERS OPERATION-CONTROL=*PARAMETERS(KEEP-REQUESTS=…)

5.11 Privileges and security aspects of HSMS

This section describes the privileges and the data security aspects of HSMS.

5.11.1 Privileges

HSMS checks user privileges.

In HSMS only a single SDF system syntax file is supplied. The selection of statements can be restricted using the software product SDF-A (see the “SDF-A” manual [14]).

HSMS checks following privileges:

- **HSMS administrator** (“HSMS administration” privilege)
  This privilege is reserved for user jobs running under a user ID to which the HSMS-ADMINISTRATION privilege has been assigned. By default, these are the SYSHSMS and TSOS user IDs.
  The security officer can use SET-PRIVILEGE command to assign this privilege to other user IDs (see the “SECOS” manual [16]). Any user working with HSMS under these privileged user IDs is referred to within the context of this manual as the HSMS administrator.
  It is not advisable to set up multiple HSMS administrations under different IDs at a given point in time.

  The statements reserved for the HSMS administrator, together with specific operands (e.g. EXPRESS-REQUEST), are available exclusively to users with this privilege.

  The HSMS administrator has the same file access rights under HSMS as the system administrator under TSOS. If he/she is not working under HSMS on the other hand, i.e. in command mode under the user ID SYSHSMS, the HSMS administrator has no special privileges. This also applies to the use of the software product ARCHIVE.
- **Subsystem administrator** (“SUBSYSTEM-MANAGEMENT” privilege)
  This privilege is required in order to start and terminate HSMS.
- **System administrator** (“TSOS” privilege)
  The HSMS server tasks are executed under this privilege. Furthermore, it is required for mounting remote file systems that are to be processed by HSMS.
- **User** (“standard processing”)
  Users have none of the above privileges and are therefore referred to in this manual as “nonprivileged”.

User ID SYSHSMS

The user ID SYSHSMS has a special meaning in HSMS: By default, the user ID SYSHSMS is assigned the HSMS-ADMINISTRATION privilege. Central HSMS work files such as control and request files are created under this user ID. Furthermore, SYSHSMS is the owner ID of all users who possess the HSMS administration privilege (see section "Privileges").

SYSHSMS is the owner ID assigned to

- archive names
- request names
- save version names.

5.11.2 Data privacy

The following mechanisms are provided for file protection:

- Only archive owners are permitted to use archive administration functions to modify and list archive attributes, delete an archive, or access an archive's save files and volumes, for example.
  Only the HSMS administrator and users working under the appropriate archive owner IDs can administer existing archives.
- Users' access rights to archives are determined by the archive definition. A public archive. All users are authorized to set up an archive that can be accessed by other users.
  The directory is created under the user ID of the archive owner (non-shareable, unless otherwise specified). If the archive directory is password-protected, this password must be entered before the archive is accessed, even by the HSMS administrator.
- Unless otherwise specified, HSMS creates save files as non-shareable files (USER-ACCESS=*USER-ONLY) under the user ID of the archive directory. This means that access to the contents of save files without using HSMS is possible only for the archive owner.
- If migrated files are to be erased by overwriting them with binary zero when deleted (DESTROY-BY-DELETE), then the storage space released by migrating such files is also cleared in this manner. The same procedure is used for deleting a save file in a migration archive on S1; the save files are created with DESTROY-BY-DELETE.
- The protection attributes of a file are not changed by recalling or copying it with the aid of HSMS.
- During automatic recall of a migrated file, for example with an OPEN command, DMS checks whether the user is authorized to access the file at all.
- All users – whether privileged or not– may use HSMS statements to access the files belonging to other user IDs if this is permitted by the data management system (DMS). All users can use the HSMS statement EXPORT-FILES to export accessible files belonging to other user IDs (with read access).
- HSMS does not check whether a user is authorized to access (possibly password-protected) files of his or her own user ID unless the contents of a file are accessed. This applies to deletion under the archival function and to the replacement of files existing in the system when restoration takes place.

Access to files of foreign user IDs

In calls of the HSMS administrator, files and job variables of foreign user IDs are supported in full.

The following restrictions apply for calls of a nonprivileged user:

- Foreign files and job variables for which co-ownership exists are supported with the statements BACKUP-FILES, MIGRATE-FILES, COPY-SAVE-FILE, RESTORE-FILES, SELECT-FILE-NAMES and SELECT-JV-NAMES.
- The recalling (both implicit and explicit) of foreign, migrated files is permissible if co-ownership exists or the file protection attributes allow access from another user ID.
- A special feature here is that, regardless of the access rights of the caller for the individual files, **all** files and job variables are taken over into the same archive or into the shadow archive when COPY-SAVE-FILE without file selection is used for copying.
- When SHOW-ARCHIVE with INFORMATION=*FILES is called, the names of foreign files and job variables are output only if co-ownership exists.
- When IMPORT- and EXPORT-FILES or ARCHIVE-FILES are called, all files and job variables are supported for which the caller has a DMS access right.

5.11.3 Handling file attributes

When SECOS is used, GUARDS is available as a mechanism for restricting access to files, libraries and library elements and job variables. Handling of file attributes is described in the section “File attribute handling” in the “ARCHIVE” manual [2]. The manual contains also information on the guard file and GUARDS. HSMS supports GUARDS in exactly the same way as the software product ARCHIVE.

In order to understand this section you must be aware that the statement sequence BACKUP-FILES/RESTORE-FILES is based on the ARCHIVE statements SAVE/ RESTORE, while the ARCHIVE-/RESTORE-FILES and EXPORT-/IMPORT-FILES statement sequences are based on ARCHIVE statements EXPORT/IMPORT.

This section has no relevance for migrated files, since such files are characterized by the fact that their catalog entry remains on the processing level and changes can be made during the period of migration. During recall, only the entries relating to the volume containing the file and the internal file name (CFID) are changed.

5.11.4 Data security

All HSMS statements that involve data manipulation are mapped onto ARCHIVE. In this respect, HSMS supports all error recovery measures carried out by ARCHIVE (see the sections “Tape and MT cartridge handling” and “Error handling for disk processing” in the “ARCHIVE” manual [2]).

5.11.5 Encrypted files

BS2000 also supports encrypted files. These are supported uniformly by HSMS in the following way:

- To save and restore, export and import, migrate and recall an encrypted file it is not necessary to specify the associated crypto password (this applies both for the administrator and nonprivileged users). No modifications are therefore required to take these files into account in procedures and jobs for saving and restoring.
- The file content is stored in the save file in encrypted format (i.e. transferred from the disk storage in the pubset in unchanged format). Loss or unauthorized reading of the backup volume does therefore not mean the file content can be read.
- Consequently in the event of a restore the encrypted file content is written back from the save file to disk unchanged.
- In this respect no performance loss is incurred when saving and restoring encrypted files.
- The restored file has the same encryption attributes as the file saved originally. In particular the same crypto password is required to open the file.

Restriction

An encrypted file cannot be restored on a private disk or Net-Storage.

5.12 Creating an HSMS configuration (example)

The following steps must be carried out to create an HSMS configuration:

- Assign pubsets to a storage level
- Creating the system archives

Optionally, volumes can be assigned to a system archive. However, this is not recommended for use with MAREN.

Allocate pubsets to a storage level

```
//START-HSMS
//MODIFY-PUBSET-PAR -  ————————————————————————————————————————  (1)
//  PUB-ID=2BC,STOR=*S1, -
//  SYSARCHIVE=*STD,SYSBACKUP=*STD,SYSVERSION=*UNDEFINED
% HSM0003 HSMS STATEMENT COMPLETED
//MODIFY-PUBSET-PAR -  ————————————————————————————————————————  (2)
//  PUB-ID=2BY,STOR=*S0(S1-PUB-ID=2BC,SYSMIGRATE=*STD), -
//  SYSARCHIVE=*STD,SYSBACKUP=*STD,SYSVERSION=*UNDEFINED
% HSM0003 HSMS STATEMENT COMPLETED
//MODIFY-PUBSET-PAR -
//  PUB-ID=BVWC,STOR=*S0(S1-PUB-ID=2BC,SYSMIGRATE=*STD), -
//  SYSARCHIVE=*STD,SYSBACKUP=*STD,SYSVERSION=*UNDEFINED
% HSM0003 HSMS STATEMENT COMPLETED
//SHOW-PUBSET-PAR ————————————————————————————————————————  (3)
```

```
SHOW-PUBSET-PARAMETERS    PUBSET-ID     = ALL      SYSBACKUP  = ANY
                          STORAGE-LEVEL = ANY      SYSARCHIVE = ANY
                          S1-PUBSET-ID  = ANY      SYSMIGRATE = ANY
                                                   SYSVERSION = ANY
------------------------------------------------------------------------
   PUBSET  ST  SYSBACKUP    SYSARCHIVE   SYSMIGRATE  S1-PUBSET  MIGRATION
               SYSBVERSION
    BVWC   S0  STD          STD          STD           2BC      ALLOWED
               NOT-DEFINED
    2BC    S1  STD          STD
               NOT-DEFINED
    2BY    S0  STD          STD          STD           2BC      ALLOWED
               NOT-DEFINED









------------------------------------------------------------------------
NEXT-PAGE : __  (+, -, ++, --, E)
```

```
% HSM0003 HSMS STATEMENT COMPLETED
//END
% HSM0014 HSMS PROGRAM TERMINATED
```

(1) First of all, pubset 2BC is assigned to storage level S1. In the following statements it can be assigned to S0 pubsets.

(2) Pubsets 2BY and BVWC are assigned to storage level S0. 2BC is defined as S1 pubset for both pubsets. 2BC is defined as S1 pubset for both pubsets. Both pubsets are to work with the default system archives defined in the global HSMS parameters.

(3) The parameters of the pubsets under HSMS management are displayed on the screen.

Creating the system archives

```
//CREATE-ARCHIVE ARCH-NAME=$SYSHSMS.HSMS.ARCHIVE, -  ————————————————————  (1)
//  DIR-NAME=$SYSHSMS.HSMS.ARCHIVE.DIR,ALLOWED-USAGE=*ARCHIVAL, -
//  TAPE-CONTROL=*PAR(NEW-STD-S-F=*IN-PERIODS(1)), -
//  USER-ACCES=*ALL-USERS(*WRITE),RETENTION-PERIOD=1
% HSM0003 HSMS STATEMENT COMPLETED
//CREATE-ARCHIVE ARCH-NAME=$SYSHSMS.HSMS.AR.2BY, -  ————————————————————  (2)
//  DIR-NAME=$SYSHSMS.HSMS.ARCHIVE.2BY.DIR,ALLOWED-USAGE= -
//  *ARCHIVAL,TAPE-CONTROL=*PAR(NEW-STD-S-F=*IN-PERIODS(1)), -
//  USER-ACCES=*ALL-USERS(*WRITE),RETENTION-PERIOD=0
% HSM0003 HSMS STATEMENT COMPLETED

//CREATE-ARCHIVE ARCH-NAME=$SYSHSMS.HSMS.ARC.BVWC, -
//  DIR=$SYSHSMS.HSMS.ARCHIVE.BVWC.DIR,ALLOWED-USAGE=*ARCHIVAL, -
//  TAPE-CONTROL=*PAR(NEW-STD-S-F=*IN-PERIODS(1)), -
//  USER-ACCES=*ALL-USERS(*WRITE),RETENTION-PERIOD=0
% HSM0003 HSMS STATEMENT COMPLETED
//CREATE-ARCHIVE ARCH-NAME=$SYSHSMS.HSMS.BACKUP, -  ————————————————————  (3)
//  DIR=$SYSHSMS.HSMS.BACKUP.DIR,ALLOWED-USAGE=*BACKUP, -
//  TAPE-CONTROL=*PAR(NEW-STD-S-F=*IN-PERIODS(3)), -
//  USER-ACCES=*ALL-USERS(*READ),RETENTION-PERIOD=0
% HSM0003 HSMS STATEMENT COMPLETED
//CREATE-ARCHIVE ARCH-NAME=$SYSHSMS.HSMS.VER.BVWC, -  ————————————————————  (4)
//  DIR=$SYSHSMS.HSMS.VER.BVWC.DIR,ALLOWED-USAGE=*VERSIONBACKUP, -
//  TAPE-CONTROL=*PAR(NEW-STD-S-F=*IN-PERIODS(3)), -
//  USER-ACCES=*ALL-USERS(*READ),RETENTION-PERIOD=0

% HSM0003 HSMS STATEMENT COMPLETED
//CREATE-ARCHIVE ARCH-NAME=$SYSHSMS.HSMS.MI.2BY, -  ————————————————————  (5)
//  DIR=$SYSHSMS.HSMS.MIGRATE.2BY.DIR,ALLOWED-USAGE=*MIGRATION, -
//  TAPE-CONTROL=*PAR(NEW-STD-S-F=*IN-PERIODS(14)), -
//  USER-ACCES=*ALL-USERS(*WRITE),RETENTION-PERIOD=365
% HSM0003 HSMS STATEMENT COMPLETED

//CREATE-ARCHIVE ARCH-NAME=$SYSHSMS.HSMS.MI.BVWC, -
//  DIR=$SYSHSMS.HSMS.MIGRATE.BVWC.DIR,ALLOWED-USAGE=*MIGRATION, -
//  TAPE-CONTROL=*PAR(NEW-STD-S-F=*IN-PERIODS(14)), -
//  USER-ACCES=*ALL-USERS(*WRITE),RETENTION-PERIOD=365
% HSM0003 HSMS STATEMENT COMPLETED
//MODIFY-HSMS-PAR DEFAULT-HSMS-STOR=*PAR(SYSARCHIVE= -  ————————————————————  (6)
//  $SYSHSMS.HSMS.ARCHIVE,SYSBACKUP=$SYSHSMS.HSMS.BACKUP, -
//  SYSMIGRATE=*UNDEF,S2-DEV-TYPE='TAPE-C4')
% HSM0003 HSMS STATEMENT COMPLETED
//SHOW-HSMS-PAR ————————————————————————————————————————————————————————  (7)
```

```
SHOW-HSMS-PARAMETERS (01)               VALID-PERIOD = SESSION
------------------------------------------------------------------------------
HSMS-ACCOUNT : HSMSACNB                   HSMS-VERSION : 12.0A00
                                         SAVE-FILE-PROC : HSMS-V10-COMP
OPERATION-CONTROL
   OPERATIONAL-MODUS   : OPERATION        NUMBER-OF-SUBTASK   : 5
   COMMON-MEMORY-SIZE  : 3                HSMS-SV-PORT-NUMBER : 1234
   FILE-SIZE-DEFAULT   : 24   ,192        FILE-SIZE-RESULT    : 24   ,192
   MONITORING : ALL                       OUTPUT              : PRINTER
                                          KEEP-REQUESTS       : 1
DEFAULT-HSMS-STORAGE
   S1-PUBSET-ID : SMS1                  S2-DEVICE-TYPE : TAPE-C4
   SYSMIGRATE : NOT-DEFINED             BACKUP-SERVER  : ABGSE714
   SYSBACKUP  : HSMS.BACKUP             SYSNODEBACKUP  : NOT-DEFINED
   SYSARCHIVE : HSMS.ARCHIVE            SYSNODEARCHIVE : NOT-DEFINED

MIGRATION-CONTROL
   EXCEPT-FILE      : NONE
   FILE-INHIBIT     : RESPECTED
   RECALL-FROM-S2   : ALLOWED             MAXIMUM-WAIT-TIME : -
   CANCEL-AT-RECALL : NOT-ALLOWED          BACKUP-MANDATORY : YES
------------------------------------------------------------------------------
NEXT-PAGE : __  (+, -, E)
```

```
 SHOW-HSMS-PARAMETERS (02)              VALID-PERIOD = SESSION
 ------------------------------------------------------------------------------
 REQUEST-WAIT-LIMITS
    DIALOG-REQUEST-TIME : 1800          BATCH-REQUEST-TIME : 3600
    DIALOG-EXEC-TIME    : 1800          BATCH-EXEC-TIME    : 3600


 DEFAULT-TAPE-CONTROL                   START-TIME     PERIOD
    READ-CONTROL    : PROCESS-REQUESTS
    WRITE-CONTROL   : PROCESS-REQUESTS
    EXPRESS-CONTROL : PROCESS-REQUESTS


 REQUEST-PRIORITIES     READ  WRITE                       READ  WRITE
    IMPORT/EXPORT :      128    128     IMPLICIT-RECALL :   128
    BACKUP        :      128    128     NODEBACKUP      :   128    128
    ARCHIVAL      :      128    128     NODEARCHIVAL    :   128    128
    MIGRATION     :      128    128     SHADOW          :   128    128
    VERSIONBACKUP :      128    128

------------------------------------------------------------------------------
NEXT-PAGE : __  (+, -, E)
```

```
% HSM0003 HSMS STATEMENT COMPLETED

//MODIFY-PUBSET-PAR -
//  PUB-ID=2BY,STOR=*S0(SYSMIGRATE=$SYSHSMS.MI.2BY), -  ————————————————  (8)
//  SYSARCHIVE=$SYSHSMS.HSMS.AR.2BY
% HSM0003 HSMS STATEMENT COMPLETED
//MODIFY-PUBSET-PAR -
//  PUB-ID=BVWC,STOR=*S0(SYSMIGRATE=$SYSHSMS.HSMS.MI.BVWC) -  ——————————  (9)
% HSM0003 HSMS STATEMENT COMPLETED
//SHOW-PUBSET-PAR ————————————————————————————————————————————————————  (10)
```

```
SHOW-PUBSET-PARAMETERS    PUBSET-ID     = ALL      SYSBACKUP  = ANY
                          STORAGE-LEVEL = ANY      SYSARCHIVE = ANY
                          S1-PUBSET-ID  = ANY      SYSMIGRATE = ANY
                                                   SYSVERSION = ANY
------------------------------------------------------------------------
   PUBSET  ST  SYSBACKUP    SYSARCHIVE   SYSMIGRATE  S1-PUBSET  MIGRATION
               SYSVERSION
    BVWC   S0  STD          STD          HSMS.MI.BVWC  2BC      ALLOWED
               HSMS.VER.BVWC
    2BC    S1  STD          STD
               NOT-DEFINED
    2BY    S0  STD          HSMS.MI.2BY  HSMS.MI.2BY   2BC      ALLOWED
               NOT-DEFINED




------------------------------------------------------------------------
NEXT-PAGE : __  (+, -, ++, --, E)
```

```
% HSM0003 HSMS STATEMENT COMPLETED
//END
% HSM0014 HSMS PROGRAM TERMINATED
```

(1) The global long-term archive is created. The standard save file is to be switched every day.

(2) Pubset-specific long-term archives are created; archival is to be pubset-based.

(3) The global backup archive is created as a public archive with read access.

(4) Pubset-specific version backup archive is created; version backup is always pubset-based only.

(5) Two pubset-specific migration archives are created. Migration is to be implemented for each pubset to prevent the archives from getting too large and to minimize the number of competing archive accesses. Switching the default save file only every 14 days enables a better utilization of the volume due to less migration.

(6) The created archives are defined as global default archives; TAPE-C4 is defined as the S2 standard volume.

(7) The selected parameters are displayed on the screen.

(8) S0 pubset 2BY is assigned a migration archive and a long-term archive on a pubset-specific basis.

(9) S0 pubset BVWC is assigned a migration archive on a pubset-specific basis. Archival takes place to the global long-term archive.

(10) The parameters of the pubsets under HSMS management and with the assigned system archives are displayed on the screen.

Filling the volume pool of a system archive

A system archive is assigned volumes which are to be used as default volumes for write operations directed at that archive.

```
//MODIFY-ARCHIVE ARCH-NAME=$SYSHSMS.HSMS.MI.2BY, -
//  VOL=*ADD(VOL=(HSMS11,HSMS22,HSMS33)) -   ————————————————————————  (1)
% HSM0003 HSMS STATEMENT COMPLETED
//END
% HSM0014 HSMS PROGRAM TERMINATED
Report (Ausgabe auf SYSLST):   ———————————————————————————————————  (2)

 ***  MODIFY - ARCHIVE               HSMS V12.0      SUMMARY      REPORT  *** 2016-08-12  14:
01:02    PAGE    1
REQUEST-NAME=TSOS     REQUEST-DATE=2016-08-12 14:01:01 USER-ID=TSOS     RUEST-
STATE=COMPLETED  WITHOUT ERROR  %
HSM0469 MODIFICATION OF ARCHIVE DIRECTORY STARTED FOR 'VOLUMES=ADD()'  %  ARC0002 STATEMENT
ACCEPTED. ARCHIVE
SEQUENCE NUMBER 'A.160812.140102', VERSION='12.0'  %  ARC0010 VOLUME OF TYPE 'TAPE-C4' WITH
VSN 'HSMS11' ADDED TO
THE POOL  %  ARC0010 VOLUME OF TYPE 'TAPE-C4' WITH VSN 'HSMS22' ADDED TO THE
POOL  %  ARC0010 VOLUME OF TYPE
'TAPE-C4' WITH VSN 'HSMS33' ADDED TO THE POOL  ***      E N D   O F                 HSMS V12.
0    SUMMARY
REPORT  *** 2016-08-12  14:01:02     ***
```

(1) Three magnetic tape cartridges are assigned to the migration archive. (In practice, of course, the number of magnetic tape cartridges to be assigned is considerably higher.)

(2) HSMS reports that the magnetic tape cartridges have been added to the pool. The volume type was taken from the global parameters.

6 Calling and executing HSMS

This chapter describes

- how the system administrator and operators load and unload HSMS
- how users or programs call HSMS
- the work files created on the invoking user’s ID during execution of HSMS (these do not include central work files which are created on the ID SYSHSMS)
- measures for enhancing performance (reducing the backup times)

6.1 Loading and unloading HSMS

HSMS consists of a privileged component (Task PRivileged, TPR modules) and a nonprivileged component (Task Unprivileged, TU load module).
The privileged component is implemented as a subsystem. This subsystem HSMS is loaded into the system memory area (class 4 memory).

HSMS must be loaded for a BS2000 session by a user equipped with the “subsystem management” privilege. After loading, HSMS is available to all users. HSMS is called by individual users via a load module.

The period of time between loading and unloading the HSMS subsystem is referred to in this manual as an HSMS session.

There are two alternative methods of loading HSMS:

Users assigned the “subsystem management” privilege can use the following DSSM command to load HSMS:

```
/START-SUBSYSTEM SUBSYSTEM-NAME=HSMS
```

HSMS is unloaded using the DSSM command

```
/STOP-SUBSYSTEM SUBSYSTEM-NAME=HSMS
```

If HSMS is loaded using a DSSM command, no control parameters can be passed to HSMS. Use of the DSSM commands permits HSMS to be loaded directly in a CMDFILE at BS2000 startup.

A user with the “subsystem management” privilege can load HSMS after calling the TU program HSMS (see section "Calling HSMS") by issuing the following statement:

```
//START-HSMS
```

This HSMS statement can be used to pass control parameters for the HSMS run (see “HSMS Vol. 2” [1], START-HSMS statement).

HSMS is unloaded with the statement:

```
//STOP-HSMS
```

Additional parameters controlling the unloading of HSMS can also be included with this HSMS statement (see “HSMS Vol. 2” [1], STOP-HSMS statement).

> **i** HSMS-SV is no longer included in the delivery package in HSMS V9.0B and higher. Consequently HSMS V9.0B or higher only supports workstations which are connected to the local BS2000-UFS via NFS.

HSMS attempts to start the ARCHIVE subsystem. If it does not succeed, HSMS itself likewise fails to start.

Any parameters modified temporarily during an HSMS session are lost even after a /HOLD-SUBSYSTEM or //STOP-HSMS SUBSYSTEM=HOLD followed by a restart.

One thing that happens when HSMS is loaded is that the BCAM host names that are to be valid for the HSMS session are defined. If work is to take place not only on the local computer, HSMS may not therefore be loaded until BCAM has been started.

6.2 Calling HSMS

- Startup and termination by the user
- SDF command processor
- Interactive mode and batch mode
- Error handling

6.2.1 Startup and termination by the user

The user calls HSMS by means of the START-HSMS command:

START-HSMS	Kurzname: **HSMS**
VERSION = ***STD** **,MONJV** = ***NONE** / <filename 1..54 without-gen-vers> ,**CPU-LIMIT** = ***JOB-REST** / <integer 1..32767 *seconds*>	

VERSION = *STD
The version selected with the SELECT-PRODUCT-VERSION command is loaded. If no version was selected, the highest available version of HSMS is loaded.

MONJV = *NONE / <filename 1..54 without-gen-vers>
Enables a monitoring job variable to be specified to monitor the program run.

CPU-LIMIT = *JOB-REST
Maximum CPU time, in seconds, which the program may use to run. The default is *JOB-REST, i.e. the remaining CPU time is used for the task.

CPU-LIMIT = <integer 1..32767 *seconds* >
The specified time is the maximum time that should be used.

Alternatively, HSMS can also be called with the START-EXECUTABLE-PROGRAM command, provided the user is permitted to execute this command:

```
/START-EXECUTABLE-PROGRAM FROM-FILE=$HSMS
```

The HSMS program is called on the system standard ID.

The user terminates HSMS by means of the statement:

```
//END
```

The period of time from calling HSMS to termination by means of END is referred to within this manual as the HSMS run.

HSMS can be invoked synchronously and asynchronously (see section "Asynchronous and synchronous processing").

6.2.2 SDF command processor

The SDF (System Dialog Facility) command processor enables the user to enter statements under menu guidance. The level of guidance can be defined by means of the command operand GUIDANCE or the standard statement MODIFY-SDF-OPTIONS.

The level of guidance is selected by specifying GUIDANCE =*MINIMUM, *MEDIUM or *MAXIMUM; the difference between the three levels is in the scope of the additional help texts supplied to the user. Irrespective of the guidance level set, the user can request additional explanations by entering “?” instead of an operand value.

Guided dialog is controlled via menus listing the statements and operands that may be specified. The list indicates the default values of the operands. The absence of a default value means that specification of an operand value is mandatory.
Parentheses “()” following an operand value indicate that the value introduces a structure, for which a submenu can be requested (by specifying the operand value).

The user can temporarily switch over to guided dialog at any time, even when in unguided dialog (GUIDANCE = *NO or *EXPERT). This is done by entering “?” instead of or following a statement. Switchover to a menu can also be requested by entering “?” after an operand that is followed by a comma.

Dialog entry of commands and statements is described in detail in the “SDF Dialog Interface” [5] and “Commands” [8] manuals.

6.2.3 Interactive mode and batch mode

HSMS can run either in interactive mode or in batch mode.

HSMS expects all the statements from the system file SYSDTA. In interactive mode, SYSDTA is assigned to the terminal, in batch mode to the ENTER file. If HSMS is to be called in a procedure, the system file must be assigned using

```
/ASSIGN-SYSDTA TO=*SYSCMD
```

6.2.4 Error handling

Before a user task regains control after an HSMS statement has been issued, HSMS sets Information about possible responses to error situations in batch mode and in procedures:

- In the event of warnings, task switch 30 is set.

  Examples:

  - A file specified for migration is not migratable.
  - A file to be restored already exists.
- In the event of simple errors, task switch 31 is set.

  Example:
  A specified file cannot be processed because of an input or output error.
- In the event of serious errors, spin-off is triggered.

  Examples:

  - An invalid statement was entered.
  - The maximum wait time was exceeded.
  - The volume for processing is missing.

The position of the task switches can be inquired in the SHOW-JOB-SWITCHES command. The SKIP-COMMANDS command can be used to branch to a branch destination, dependent on one or more of the task switches.

Spin-off causes all subsequent commands to be skipped up to the next SET-JOB-STEP or EXIT-JOB, or up to the end of the procedure or ENTER file.

With asynchronous processing, error situations detected by the HSMS server task result neither in a task switch being set nor in spin-off.

Monitoring with job variables

A monitoring job variable can be specified in the START-HSMS command for the HSMS run, and this monitors the entire program run (see the “Job Variables” manual [23]).

HSMS itself does not support monitoring job variables for its calls. However, for the individual calls a job variable can be specified in which HSMS places information over the status of this call (see section "Job variable for request monitoring").

6.3 Calling HSMS from programs

HSMS statements can also be issued from programs by means of the HSMS macro. In the HSMS macro, HSMS statements are issued in “free string format”, i.e. the same format used for statements to the program.
The only output is a return code written to the standard header (see the “Executive Macros” manual [9]). The macro destroys registers 1, 14 and 15.

If the macro has not been added to the system macro library, SYSLIB.HSMS.120 (see section "Structure of the release unit HSMS") must be assigned to the assembler as ALTLIB.
The same library must be assigned as the TASKLIB before linkage or before the object module is called.

Operation	Operands
HSMS	MF = S | L | C D | E [,LENGTH = <number>] [,ADDR = { <address> | <(reg)> }] [,PREFIX = <prefix>] [,PARAM = { <address> | <(reg)> }]

MF	A list of operands is generated, see the “Executive Macros” manual [9].
= S	Standard format, data and instruction code are not separated.
= L	Only data is generated.
= C	A CSECT (control section) is generated.
= D	A DSECT (dummy section) is generated.
= E	The instruction code only is generated.
LENGTH	defines the length of the memory area to be reserved for statement entry. LENGTH is mandatory if MF=S/L is specified, otherwise it is ignored. The maximum length is 16372 byte.
ADDR	defines the address of the memory area to be reserved for statement entry. LENGTH is mandatory if MF=S/L is specified, otherwise it is ignored.
PREFIX	defines the first character of the HSMS parameter list. The operand is ignored if MF=E/S is specified.
PARAM	controls access to the parameter list. PARAM is mandatory if MF=E isspecified, otherwise it is ignored. The default value is “(1)”, which means that the address of the parameterlist is expected to be found in register 1.

The parameter list starts with a standard header, and is followed by the length of the HSMS statement and the address of a field that contains the HSMS statement.

If the parameter list is updated dynamically, users are responsible for ensuring that the correct values are inserted in the correct fields.

Return codes and error classes

The standard header contains a main code (MC) plus two subcodes, subcode 1 (SC1) and subcode 2 (SC2). All three together supply precise and detailed information about the execution of the action initiated by the macro. The return codes are returned in the order SC1, SC2, MC (4 bytes).

MC	SC1	SC2	Meaning
X'0000'	X'00'	X'00'	The statement has been executed without errors.
		X'01'	The statement has been executed, no action was required.
		X'02'	The statement has been checked for validity and accepted for asynchronous processing.
X'0001'	X'00'		The statement has been executed with warnings (warnings are issued if HSMS is unable or not allowed to process a user-specified object: file cannot be migrated, file to be restored already exists, ...).
X'0002'	X'00'		The statement has been executed with errors (error messages are output if a user-specified object cannot be processed either partially or completely, e.g. file is locked, input or output error, etc.).
X'0003'	X'00'		The statement has not been executed because the maximum permissible wait time as defined in the HSMS control file elapsed while waiting for the request to be completed.
X'0004'	X'01'		The statement has been rejected because of a syntax error.
	X'02'		The statement has been rejected due to a system error or an internal HSMS error.
	X'40'		The statement has been rejected due to other errors, e.g. privilege errors.
X'0005'			Statement execution has been aborted due to an error which made continuation pointless, e.g. lack of resources. The statement can usually be continued by means of RESTART-REQUEST.
X'FFFF'	X'01'		HSMS is not known in the system.
	X'02'		HSMS is not available, e.g. because the syntax file is missing.
	X'03'		Version error, e.g. the version of the syntax file is invalid.
	X'..'		The remaining subcodes correspond to the standard header.

Example

The program below enables the user to enter HSMS statement; it calls HSMS via the program interface.

```
ASSEMBH
LISTING
09:05:55  2016-02-29   PAGE 0003
 LOCTN  OBJECT CODE    ADDR1    ADDR2     STMNT  M  SOURCE STATEMENT
 000000                                       1     HSMSMAC  CSECT ,
                                              2     HSMSMAC  AMODE ANY
                                              3     HSMSMAC  RMODE ANY
                                              4              GPARMOD 31
                                              5   1                *,MACRO: GPARMOD,
VERSION: VER121
 000000 0D A0                                 6              BASR  10,0
 000002                 00000002              7              USING *,10
 000002 41 70 A096      00000098              8              LA    7,LHSMS
 000006                 00000000              9              USING HSMSREF,7
                                             10              PRINT GEN
                                             11     WHAT     RDATA MF=(E,LRDATA)
 000006 41 10 A0A6      000000A8             12   1 WHAT     LA    1,LRDATA
LOAD ADDR PARAM LIST INTO R1
 00000A 0A 27                                13   1          SVC   39
SYSFILE SVC
                                             14   1 *
 00000C D5 03 A0EAA1F6 000000EC 000001F8     15              CLC   STMT(4),=C'END '
 000012 47 80 A05C      0000005E             16              BE    TERM
                                             17     CALL     HSMS  MF=E,PARAM=LHSMS
                                             18   1 CALL     MFCHK
MF=E,                                                 C
                                             18   1                SUPPORT=(C,D,E,L,
S),                                    C
                                             18   1
PREFIX=D,                                             C
                                             18   1
MACID=HSM,                                            C
                                             18   1
DMACID=HSM,                                           C
                                             18   1
DNAME=HSMPAR,                                         C
                                             18   1
PARAM=LHSMS,                                          C
                                             18   1
ENTRY=IDHSASS,                                        C
                                             18   1                ALIGN=F
 000016                                      19   2 CALL     DS    0Y
 000016 58 F0 A1FA      000001FC             20   2          L     15,=V(IDHSASS)
 00001A 41 10 A096      00000098             21   2          LA    1,LHSMS
 00001E 0D EF                                22   2          BASR  14,15
 000020 D2 03 A1BE7004 000001C0 00000004     23              MVC   MPACK,DHSMRET
 000026 F3 84 A1B2A1BE 000001B4 000001C0     24              UNPK  MRS(9),MPACK(5)
 00002C D4 07 A1B2A1D3 000001B4 000001D5     25              NC    MRS(8),ANDMASK
 000032 DC 07 A1B2A1C3 000001B4 000001C5     26              TR    MRS(8),MTAB
 000038 D2 03 A1E7A1B6 000001E9 000001B8     27              MVC   RCMC,MRS+4
 00003E D2 01 A1ECA1B4 000001EE 000001B6     28              MVC   RCSUB1,MRS+2
 000044 D2 01 A1EFA1B2 000001F1 000001B4     29              MVC   RCSUB2,MRS
                                             30     WRITERC  WROUT MF=(E,LWROUT)
```

```
 00004A 41 10 A0CA     000000CC              31   1 WRITERC  LA    1,LWROUT
LOAD ADDR PARAM LIST INTO R1
 00004E 0A 27                                32   1          SVC   39
SYSFILE SVC
                                             33   1 *
 000050 92 40 A0EA     000000EC              34              MVI   STMT,C' '
 000054 D2 C6 A0EBA0EA 000000ED 000000EC     35              MVC   STMT+1(L'STMT-1),STMT
 00005A 47 F0 A004     00000006              36              B     WHAT
                                             37     TERM     TERM
 00005E                                      38   1 TERM     DS
0H                                                    206
 00005E 41 10 A066     00000068              39   1          LA    1,
S0006D                                                205
 000062 47 F0 A076     00000078              40   1          B
S0006S                                                200
 000068                                      41   1 S0006D     DS
0F                                                    200
                                             42   1          FHDR  UNIT=6,FUNCT=40,
VERS=1                                 207
 000068                                      43   2          DS    0A
 000068                                      44   2          DS    0XL8
GENERAL OPERAND LIST HEADER
 000068 0006                                 45   2          DC    AL2(6)
FUNCTION UNIT NUMBER
 00006A 28                                   46   2          DC    AL1(40)
FUNCTION NUMBER
 00006B 01                                   47   2          DC    AL1(1)
FUNCTION INTERFACE VERSION NUMBER
 00006C FFFFFFFF                             48   2          DC    X'FFFFFFFF'
Returncode is virgin
 000070 01                                   49   1          DC
XL1'01'                                               207
 000071 00                                   50   1          DC    XL1'00'
 000072 00                                   51   1          DC    XL1'00'
 000073 04                                   52   1          DC    XL1'04'
 000074 40404040                             53   1          DC    CL4'    '
 000078                                      54   1 S0006S     DS
0Y                                                    200
 000078 0A 09                                55   1          SVC   9
                                             56     TERMD    TERM  DUMP=Y,MODE=A
ASSEMBH
LISTING
09:05:55  2016-02-29   PAGE 0004
 LOCTN  OBJECT CODE    ADDR1    ADDR2     STMNT  M  SOURCE STATEMENT
 00007A                                      57   1 TERMD    DS
0H                                                    206
 00007A 41 10 A082     00000084              58   1          LA    1,
S0008D                                                205
 00007E 47 F0 A092     00000094              59   1          B
S0008S                                                200
 000084                                      60   1 S0008D     DS
0F                                                    200
                                             61   1          FHDR  UNIT=6,FUNCT=40,
VERS=1                                 207
 000084                                      62   2          DS    0A
 000084                                      63   2          DS    0XL8
GENERAL OPERAND LIST HEADER
 000084 0006                                 64   2          DC    AL2(6)
FUNCTION UNIT NUMBER
```

```
 000086 28                                65   2          DC    AL1(40)
FUNCTION NUMBER
 000087 01                                66   2          DC    AL1(1)
FUNCTION INTERFACE VERSION NUMBER
 000088 FFFFFFFF                          67   2          DC    X'FFFFFFFF'
Returncode is virgin
 00008C 01                                68   1          DC
XL1'01'                                        207
 00008D 01                                69   1          DC    XL1'01'
 00008E 04                                70   1          DC    XL1'04'
 00008F 04                                71   1          DC    XL1'04'
 000090 40404040                          72   1          DC    CL4'    '
 000094                                   73   1 S0008S     DS
0Y                                             200
 000094 0A 09                             74   1          SVC   9
                                          75     *
                                          76     LHSMS    HSMS  ADDR=STMT,LENGTH=200,MF=L
                                          77   1 LHSMS    MFCHK
MF=L,                                                C
                                          77   1               SUPPORT=(C,D,E,L,
S),                                  C
                                          77   1
PREFIX=D,                                            C
                                          77   1
MACID=HSM,                                           C
                                          77   1
DMACID=HSM,                                          C
                                          77   1
DNAME=HSMPAR,                                        C
                                          77   1
PARAM=,                                              C
                                          77   1
ENTRY=IDHSASS,                                       C
                                          77   1
ALIGN=F                                                |      |
 000098                                  78   2 LHSMS    DS    0F
 000098                                  79   1 DHSMLST  DS
0H                                                     |      |
 000098                                  80   1 HSM20010     DS
0H                                                   |      |
                                         81   1          FHDR  MF=L,UNIT=73,FUNCT=1,
VERS=1                                |      |
 000098                                  82   2          DS    0A
 000098                                  83   2          DS    0XL8
GENERAL OPERAND LIST HEADER
 000098 0049                             84   2          DC    AL2(73)
FUNCTION UNIT NUMBER
 00009A 01                               85   2          DC    AL1(1)
FUNCTION NUMBER
 00009B 01                               86   2          DC    AL1(1)
FUNCTION INTERFACE VERSION NUMBER
 00009C FFFFFFFF                         87   2          DC    X'FFFFFFFF'
Returncode is virgin
 0000A0 00C8                             88   1          DC    Y(200)        LENGTH OF
STATEMENT                          |      |
 0000A2 00                               89   1          DC    X'0'
UNUSED                                |      |
 0000A3 FF                               90   1          DC    AL1(255)      REGISTER
NUMBER                              |      |
```

```
 0000A4 000000EC                          91   1          DC    A(STMT)       ADDRESS OF
STATEMENT                        |      |
                       00000010           92   1 DHSMLE   EQU   *-
DHSMLST                                        |      |
                                           93     *
                                           94     LRDATA   RDATA MSGOUT,TERMD,STMT#,
KEYOUT=Y,MF=L
 0000A8                                    95   1 S0013D     DS
0F                                            A340
                                           96   1 LRDATA   FHDR  UNIT=36,FUNCT=18,VERS=2
 0000A8                                    97   2          DS    0A
 0000A8                                    98   2 LRDATA   DS    0XL8
GENERAL OPERAND LIST HEADER
 0000A8 0024                               99   2          DC    AL2(36)
FUNCTION UNIT NUMBER
 0000AA 12                                100   2          DC    AL1(18)
FUNCTION NUMBER
 0000AB 02                                101   2          DC    AL1(2)
FUNCTION INTERFACE VERSION NUMBER
 0000AC FFFFFFFF                          102   2          DC    X'FFFFFFFF'
Returncode is virgin
                                          103   1 *
 0000B0 0000007A                          104   1          DC    A(TERMD)
ERROR ADDRESS
 0000B4 000000E8                          105   1          DC    AL4(MSGOUT)
READ IN AREA ADDRESS
 0000B8                                   106   1          DS    AL1(0)
PLACE FOR I.EDIT BYTE 1
 0000B9                                   107   1          DS    AL1(0)
PLACE FOR I.EDIT BYTE 2
 0000BA 00                                108   1          DC    AL1(0)
SYSDTA ASSIGNMENT
 0000BB 00                                109   1          DC    AL1(0)
FLAG BYTE 1
 0000BC 00CC                              110   1          DC    AL2(STMT#)
LENGTH OF READ
                                          111   1 *
 0000BE 80                                112   1          DC    AL1(128)
FLAG TABLE BYTE
 0000BF 00                                113   1          DC    AL1(0)
ASSIGNMENT CHANGE INDICATOR
 0000C0 0000                              114   1          DC    H'0'
KEY-POSITION
 0000C2 0000                              115   1          DC    H'0'
KEY-LENGTH
 0000C4 00000000                          116   1          DC    AL4(0)
VTSUCB ADDRESS
 0000C8 0000                              117   1          DC    AL2(0)
INPUT TIMER VALUE            009
ASSEMBH
LISTING
09:05:55  2012-02-29   PAGE 0005
 LOCTN  OBJECT CODE    ADDR1    ADDR2     STMNT  M  SOURCE STATEMENT
 0000CA 0000                              118   1          DC    H'0'
RES_FOR_TIAM                 007
                                          119   1 *
                                          120   1          @DCEI DCEDIT=,MODE=,IGETFC=,
ICFD=,                           C
                                          120   1                ITRSUP=,ILINEND=,
```

```
IGETBS=,                                C
                                              120   1                IMANUAL=,ILCASE=,
IHDR=,                                  C
                                              120   1                IGETIC=,RDA1=-20,RDA2=-19
 0000CC                 000000B8              121   2           ORG   *-20
 0000B8 00                                    122   2           DC    AL1(0)
 0000B9                 000000CC              123   2           ORG   *+20-1
 0000CC                 000000B9              124   2           ORG   *-19
 0000B9 00                                    125   2           DC    AL1(0)
 0000BA                 000000CC              126   2           ORG   *+19-1
                                              127   2                 *,@DCEI      999
921011   53531002
                                              128   1 *
                                              129     *
                                              130     LWROUT    WROUT RC,TERMD,MF=L
 0000CC                                       131   1 S0016D     DS
0F                                                     A340
                                              132   1 LWROUT    FHDR  UNIT=36,FUNCT=17,VERS=2
 0000CC                                       133   2           DS    0A
 0000CC                                       134   2 LWROUT    DS    0XL8
GENERAL OPERAND LIST HEADER
 0000CC 0024                                  135   2           DC    AL2(36)
FUNCTION UNIT NUMBER
 0000CE 11                                    136   2           DC    AL1(17)
FUNCTION NUMBER
 0000CF 02                                    137   2           DC    AL1(2)
FUNCTION INTERFACE VERSION NUMBER
 0000D0 FFFFFFFF                              138   2           DC    X'FFFFFFFF'
Returncode is virgin
                                              139   1 *
 0000D4 0000007A                              140   1           DC    AL4(TERMD)
ERROR ADDRESS
 0000D8 000001E0                              141   1           DC    AL4(RC)
MESSAGE AREA ADDRESS
 0000DC                                       142   1           DS    AL1(0)
PLACE FOR EDIT BYTE 1
 0000DD                                       143   1           DS    AL1(0)
PLACE FOR EDIT BYTE 2
 0000DE 00                                    144   1           DC    AL1(0)
RESERVED
 0000DF 00                                    145   1           DC    AL1(0)
FLAG BYTE 1
 0000E0 00000000                              146   1           DC    AL4(0)
VTSUCB ADDRESS
 0000E4 00000000                              147   1           DC    F'0'
RES_FOR_TIAM
                                              148   1 *
                                              149   1           @DCEO DCEDIT=,MODE=,
OEXTEND=,                                 C
                                              149   1                 OTRSUP=,OLINEND=,
OMANUAL=,                               C
                                              149   1                 OPTAPE=,OHCOPY=,
ONOPOSN=,                                C
                                              149   1                 OHDR=,OETB=,OHOM=,
OINFO=,                                 C
                                              149   1                 OBELL=,OTRANS=,
ONOLOGC=,                                 C
                                              149   1                 RDA1=-12,RDA2=-11
 0000E8                 000000DC              150   2           ORG   *-12
```

```
 0000DC 00                                 151   2          DC    AL1(0)
 0000DD                 000000E8           152   2          ORG   *+12-1
 0000E8                 000000DD           153   2          ORG   *-11
 0000DD 00                                 154   2          DC    AL1(0)
 0000DE                 000000E8           155   2          ORG   *+11-1
                                           156   2                *,@DCEO      999
921011   53531004
                                           157   1 *
                                           158     *
 0000E8                                    159     MSGOUT   DS    0F
 0000E8 00CC                               160     STMT_L   DC    Y(STMT#)
 0000EA 0000                               161     STMT_R   DC    X'0000'
 0000EC 4040404040404040                   162     STMT     DC    CL200' '
                        000000CC           163     STMT#    EQU   *-STMT_L
                                           164     *
 0001B4 4040404040404040                   165     MRS      DC    CL9' '
 0001C0                                    166     MPACK    DS    F
 0001C4 00                                 167              DC    X'00'
 0001C5 F0F1F2F3F4F5F6F7                   168     MTAB     DC    C'0123456789ABCDEF'
 0001D5 0F0F0F0F0F0F0F0F                   169     ANDMASK  DC    X'0F0F0F0F0F0F0F0F'
 0001E0                                    170     RC       DS    0F
 0001E0 0013                               171              DC    Y(RC#)
 0001E2 0000                               172              DC    X'0000'
 0001E4 00                                 173              DC    X'00'
 0001E5 D9C37A40                           174              DC    C'RC: '
 0001E9                                    175     RCMC     DS    CL4
 0001ED 40                                 176              DC    C' '
 0001EE                                    177     RCSUB1   DS    CL2
 0001F0 40                                 178              DC    C' '
 0001F1                                    179     RCSUB2   DS    CL2
                        00000013           180     RC#      EQU   *-RC
ASSEMBH
LISTING
09:05:55  2012-02-29   PAGE 0006
 LOCTN  OBJECT CODE    ADDR1    ADDR2     STMNT  M  SOURCE STATEMENT
                                           181     HSMSREF  HSMS  MF=D
                                           182   1 HSMSREF  MFCHK
MF=D,                                                  C
                                           182   1                SUPPORT=(C,D,E,L,
S),                                  C
                                           182   1
PREFIX=D,                                              C
                                           182   1
MACID=HSM,                                             C
                                           182   1
DMACID=HSM,                                            C
                                           182   1
DNAME=HSMPAR,                                          C
                                           182   1
PARAM=,                                                C
                                           182   1
ENTRY=IDHSASS,                                         C
                                           182   1
ALIGN=F                                                 |      |
 000000                                    183   2 HSMSREF  DSECT ,
                                           184   2                *,##### PREFIX=D,
MACID=HSM #####
                                           185   1          FHDR  MF=(C,DHSM),
EQUATES=YES                                 |      |
```

```
 000000                                  186   2          DS    0A
 000000                                  187   2 DHSMFHE  DS    0XL8            0
GENERAL PARAMETER AREA HEADER
                                        188   2 *
 000000                                  189   2 DHSMIFID DS    0A              0
INTERFACE IDENTIFIER
 000000                                  190   2 DHSMFCTU DS    AL2             0
FUNCTION UNIT NUMBER
                                        191   2 *                                  BIT
15    HEADER FLAG BIT,
                                        192   2 *                                  MUST
BE RESET UNTIL FURTHER NOTICE
                                        193   2 *                                  BIT
14-12 UNUSED, MUST BE RESET
                                        194   2 *                                  BIT
11-0  REAL FUNCTION UNIT NUMBER
 000002                                  195   2 DHSMFCT  DS    AL1             2
FUNCTION NUMBER
 000003                                  196   2 DHSMFCTV DS    AL1             3
FUNCTION INTERFACE VERSION NUMBER
                                        197   2 *
 000004                                  198   2 DHSMRET  DS    0A              4
GENERAL RETURN CODE
                                        199   2 *
                                        200   2 * GENERAL_RETURN_CODE CLEARED
(X'00000000') MEANS
                                        201   2 * REQUEST SUCCESSFUL PROCESSED AND NO
ADDITIONAL INFORMATION
                                        202   2 *
 000004                                  203   2 DHSMSRET DS    0AL2            4   SUB
RETURN CODE
 000004                                  204   2 DHSMSR2  DS    AL1             4   SUB
RETURN CODE 2
                                        205   2 * ALWAYS CLEARED (X'00') IF
MAIN_RETURN_CODE IS X'FFFF'
                                        206   2 * Standard subcode2 values as defined by
convention:
                       00000000         207   2 DHSMR2OK EQU   X'00'               All
correct, no additional info
                       00000001         208   2 DHSMR2NA EQU   X'01'
Successful, no action was necessary
                       00000002         209   2 DHSMR2WA EQU   X'02'
Warning, particular situation
 000005                                  210   2 DHSMSR1  DS    AL1             5   SUB
RETURN CODE 1
                                        211   2 *
                                        212   2 * GENERAL INDICATION OF ERROR CLASSES
                                        213   2 *
                                        214   2 * CLASS A    X'00'           FUNCTION
WAS SUCCESSFULLY PROCESSED
                                        215   2 * CLASS B    X'01' - X'1F'   PARAMETER
SYNTAX ERROR
                                        216   2 * CLASS C    X'20'           INTERNAL
ERROR IN CALLED FUNCTION
                                        217   2 * CLASS D    X'40' - X'7F'   NO CLASS
SPECIFIC REACTION POSSIBLE
                                        218   2 * CLASS E    X'80' - X'82'   WAIT AND
RETRY
                                        219   2 *
```

```
                        00000000             220   2 DHSMRFSP EQU   X'00'
FUNCTION SUCCESSFULLY PROCESSED
                        00000001             221   2 DHSMRPER EQU   X'01'
PARAMETER SYNTAX ERROR
                                             222   2 *  3 GLOBALLY DEFINED ISL ERROR CODES IN
CLASS X'01' - X'1F'
                        00000001             223   2 DHSMRFNS EQU   X'01'
CALLED FUNCTION NOT SUPPORTED
                        00000002             224   2 DHSMRFNA EQU   X'02'
CALLED FUNCTION NOT AVAILABLE
                        00000003             225   2 DHSMRVNA EQU   X'03'
INTERFACE VERSION NOT SUPPORTED
                                             226   2 *
                        00000004             227   2 DHSMRAER EQU   X'04'
ALIGNMENT ERROR
                        00000020             228   2 DHSMRIER EQU   X'20'
INTERNAL ERROR
                        00000040             229   2 DHSMRCAR EQU   X'40'
CORRECT AND RETRY
                                             230   2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN
CLASS X'40' - X'7F'
                        00000041             231   2 DHSMRECR EQU   X'41'
SUBSYSTEM (SS) MUST BE CREATED
                                             232   2 *
EXPLICITELY BY CREATE-SS
                        00000042             233   2 DHSMRECN EQU   X'42'              SS
MUST BE EXPLICITELY CONNECTED
                                             234   2 *
                        00000080             235   2 DHSMRWAR EQU   X'80'              WAIT
FOR A SHORT TIME AND RETRY
                        00000081             236   2 DHSMRWLR EQU   X'81'
"     LONG       "
                        00000082             237   2 DHSMRWUR EQU   X'82'              WAIT
TIME IS UNCALCULABLY LONG
                                             238   2 *                                BUT
RETRY IS POSSIBLE
                                             239   2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN
CLASS X'80' - X'82'
                        00000081             240   2 DHSMRTNA EQU   X'81'              SS
TEMPORARILY NOT AVAILABLE
                        00000082             241   2 DHSMRDH  EQU   X'82'              SS IN
DELETE / HOLD
                                             242   2 *
ASSEMBH
LISTING
09:05:55  2012-02-29   PAGE 0007
 LOCTN  OBJECT CODE    ADDR1    ADDR2     STMNT  M  SOURCE STATEMENT
 000006                                     243   2 DHSMMRET DS    0AL2            6   MAIN
RETURN CODE
 000006                                     244   2 DHSMMR2  DS    AL1             6   MAIN
RETURN CODE 2
 000007                                     245   2 DHSMMR1  DS    AL1             7   MAIN
RETURN CODE 1
                                            246   2 *
                                            247   2 * SPECIAL LAYOUT OF
LINKAGE_MAIN_RETURN_CODE (YYYY IN X'00XXYYYY')
                                            248   2 *
                        0000FFFF            249   2 DHSMRLNK EQU   X'FFFF'
LINKAGE ERROR / REQ. NOT PROCESSED
```

```
                       00000008             250   2 DHSMFHL  EQU   8               8
GENERAL OPERAND LIST HEADER LENGTH
                                            251   2 *
                       00000008             252   1 DHSMLIST EQU
*                                                        |      |
 000008                                     253   1 DHSMLGTH DS    H           LENGTH OF
STATEMENT STRING                    |      |
 00000A                                     254   1          DS    X
UNUSED                                       |      |
 00000B                                     255   1 DHSMREG  DS    AL1         REGISTER #
CONTAINING STRING ADDRESS          |      |
 00000C                                     256   1 DHSMADDR DS    F           ADDRESS OF
STATEMENT STRING                    |      |
                       00000008             257   1 DHSMLENG EQU   *-
DHSMLIST                                               |      |
                                            258              END ,  *** ,NO END CARD
 0001F8 C5D5C440                            259                  =C'END '
 0001FC 00000000                            260                  =V(IDHSASS)
 000200 0309290905446314                    261                  =X'0309290905446314'
CONSISTENCY CONSTANT FOR AID
FLAGS IN 00000 STATEMENTS, 000 PRIVILEGED FLAGS, 000 MNOTES
HIGHEST ERROR-WEIGHT : NOTE
THIS PROGRAM WAS ASSEMBLED BY ASSEMBH     V01.2C00     ON 2012-02-29 AT 09:05:44
ASSEMBH
LISTING
09:05:55  2012-02-29   PAGE 0008
USED FILES AND LIBRARIES
SOURCE FILE   :   :2OSG:$USER1.ALF.SRC.HSMSMAC
MACRO-LIBRARIES     LINKNAME  LIBRARY-NAME
                              :2OSG:$USER1.ALFX2.SYSLIB.HSMS.090
                              :2RZV:$RZV.SM.SSG.BS2CP.V18.0.GCLIB.UR
ASSEMBLY TIME :          0.672     SEC.
THIS LISTING WAS GENERATED BY THE LISTING GENERATOR V 1.2C00.
```

The HSMS macro is used in the default form MF=L. The address of the statement is specified; the maximum length is 200 bytes.

```
/START-BINDER   ————————————————————————————————————————————————————————  (1)
//START-LLM-CREATION INTERNAL-NAME=HSMS-MACRO-EXEC
//INCLUDE-MODULES MODULE-CONTAINER=*OMF(ELEMENT=*ALL)
//INCLUDE-MODULES MODULE-CONTAINER=*LIBRARY-ELEMENT -
//   (LIBRARY=$SYSHSMS.SYSLIB.HSMS.090,ELEMENT=HSMSGC)   ————————————————  (2)
//SAVE-LLM MODULE-CONTAINER=*LIBRARY-ELEMENT(LIBRARY=HSMS.MACRO.PL)
//END
/
/START-EXECUTABLE-PROGRAM FROM-FILE=*LIBRARY-ELEMENT -
/    (LIBRARY=HSMS.MACRO.PL,ELEMENT=HSMS-MACRO-EXEC)
```

(1) The BINDER linkage editor is called.

(2) The HSMS system library must be assigned when the generated module is linked, so that the entry HSMSGC can be linked.

6.4 Work files

HSMS requires a number of work files for operation. The control file and the request file are of primary importance for HSMS operation. They are created during installation. The other files are temporary files which serve as output or scratch files.
As of HSMS V7.0 the files are neutral created in NK format (using BLOCK-CONTROL-INFO=*WITHIN-DATA-4K-BLOCK and BUFFER-LENGTH=*STD(SIZE=4)).

The maximum size of the HSMS work files is restricted to 32 GB. If this size is exceeded, for example, as a result of user intervention, then HSMS is no longer able to open or process the work file.

Control files

All parameters defined for HSMS operation are stored in control files. That is why control files are so important to HSMS operation.

A central control file manages the global HSMS parameters, all the definitions for the SF pubset environment (archives and pubsets) and the definitions for the workstations. There is only one central control file. It is created under the SYSHSMS user ID and the default catalog ID. The name of the central control file is:

```
HSM.1.CONTROL.FILE
```

In addition to the central control file, a local control file is also created for each SM pubset under HSMS control. This file is created under the SYSHSMS user ID on the control volume set of the SM pubset. It contains the global HSMS parameters specific to, and the archives defined in, the SM pubset environment.
The name of the local control file for an SM pubset is:

```
HSM.1.SM.CONTROL.FILE
```

The control files are created as ISAM files with USER-ACCESS=*OWNER-ONLY without password protection.

The control files are checked at the start of each HSMS session. All the definitions contained in the control files are read into internal tables. Because the control files are upwardly compatible, old data records are automatically recognized and updated.
You must make a copy of a control file before an upgrade if you wish to retain the option of reverting back to your present version. For details on the compatibility of control files, see section "Compatibility of HSMS definitions in different HSMS environments".

A new central control file is created if no such file is found when an HSMS session is started. It already contains presettings. These presettings are listed in “HSMS Vol. 2” [1] at the end of the HSMS statement MODIFY-HSMS-PARAMETERS.

During an HSMS session, the global HSMS parameters are modified with the following statement:

```
//MODIFY-HSMS-PARAMETERS VALID-PERIOD=*PERMANENT,...
```

The control file of an SM pubset is created during the declaration of the SM pubset with the CREATE-SM-PUBSET-PARAMETERS statement.
A new control file is created for an SM pubset if none is found at the start of an HSMS session or when a pubset is imported. It already contains presettings. These presettings are listed in “HSMS Vol. 2” [1] at the end of the MODIFY-SM-PUBSET-PARAMETERS statement.

The parameters of a specific SM pubset can be changed during an HSMS session with the aid of the MODIFY-SM-PUBSET-PARAMETERS statement.

The central HSMS control file also contains the parameter definitions of the pubsets and workstations managed by HSMS.

> When HSMS is started, a check is made to determine whether remote workstations are entered in the control file. As HSMS-SV is normally no longer available in HSMS V9.0B and higher, these workstations are skipped and HSMS startup is continued. The remote workstations are not deleted from the control file.

Request file

The requests generated by HSMS as a result of action statements or DMS access to migrated files are all entered in the request file of the SF or SM pubset environment for which they were issued.

The central request file (requests for an SF environment) is created under the SYSHSMS user ID and under the default catalog ID. The name of the request file is:

```
HSM.2.REQUEST.FILE
```

In an SM pubset environment, a request file is created for each SM pubset under the SYSHSMS user ID and under the catalog ID of that SM pubset. The name of the request file for an SM pubset is:

```
HSM.2.SM.REQUEST.FILE
```

Request files created with earlier HSMS versions are not compatible. The central request file is checked when an HSMS session is started. If an incompatible request file is detected, the start is aborted. For details on the compatibility of request files, see section "Compatibility of HSMS definitions in different HSMS environments".

The request files for SM pubsets are checked at the start of an HSMS session (for the accessible environments) or for import processing. If an incompatible request file is detected, the start of that specific environment fails.

If a task detects that no request file is present, a new request file is automatically created. It is created empty as an ISAM (old) file with USER-ACCESS=*OWNER-ONLY without password protection.

This file is interrogated for the (re)start of asynchronous requests. Request management is handled by means of the HSMS statements SHOW-REQUESTS, RESTART-REQUESTS and DELETE-REQUESTS.

Files used in action statement processing

The *result file* with the name:

```
HSM.A.<user-id><date><time>.RESULT
```

contains the result of the ARCHIVE run initiated for the processing of an HSMS action statement. For data transfer, the result file is created under the caller’s user ID; otherwise it is created under the SYSHSMS user ID and the default catalog ID of the environment to which the request refers (default catalog ID or SM pubset ID).

The result file is created as an ISAM file with USER-ACCESS=*OWNER-ONLY without password protection. The <user-id> in the file name is the ID of the user who issued the request.

The disk storage space required for the result file depends on the number of objects processed as one record per object is written to the result file.

ARCHIVE needs the result file in order to restart using the RESTART-REQUESTS statement. It is deleted after normal termination or when a DELETE-REQUESTS statement is issued.

For requests in an SF environment, the result file with the following name is created for requests for a shared pubset imported in slave mode which are processed on the master:

```
HSM.A.<user-id><date><time>.<sysid>.RESULT
```

<sysid> the system ID of the system issuing the request in the shared public volume sets.

For requests in an SM environment, a result file with the following name is created in the SM pubset:

```
HSM.A.<user-id><date><time>.<rq-name>.RESULT
```

<rq-name> is the request name that was allocated by the user. Since the request name may only be up to 8 characters long, the suffix “RESULT” can be abbreviated so that the file name length complies with the DMS conventions. Only the first three characters of the suffix “RESULT” are guaranteed.

The *report file* with the name

```
HSM.B.<user-id><date><time>.REPORT
```

contains, among other things, the result of the evaluation of the ARCHIVE result file by HSMS. Just like the result file, it is created under the SYSHSMS user ID and under the catalog ID of the environment to which the request refers (default catalog ID or
SM pubset ID).

The report file is created as non-shareable without password protection. The report file is deleted after creation of the print file.

The disk storage space required for the report file depends on the number of objects processed as one record per object is written to the report file.

In an SF environment, a report file with the following name is created on the shared pubset (or on one of the shared pubsets) for requests for a shared pubset imported in slave mode which are processed on the master:

```
HSM.B.<user-id><date><time>.<sysid>.REPORT
```

<sysid> the system ID of the system issuing the request in the shared public volume sets.

For requests in an SM environment, a report file with the following name is created in the SM pubset:

```
HSM.A.<user-id><date><time>.<rq-name>.REPORT
```

<rq-name> is the request name that was allocated by the user. Since the request name may only be up to 8 characters long, the suffix “REPORT” can be abbreviated so that the file name length complies with the DMS conventions. Only the first three characters of the suffix “REPORT” are guaranteed.

The *print file* with the name

```
HSM.C.<user-id><date><time>.PRINT
```

contains the edited result of evaluation of the report file for those users for whom output of the report on the printer has been defined. It is created under the caller’s user ID (or, for requests from the HSMS administrator, under SYSHSMS) and deleted after printing.

For requests in an SM environment, a print file with the following name is created in the SM pubset:

`HSM.A.<user-id><date><time>.<rq-name>.PRINT`

<rq-name> is the request name that was allocated by the user. Since the request name may only be up to 8 characters long, the suffix “PRINT” can be abbreviated so that the file name length complies with the DMS conventions. Only the first three characters of the suffix “PRINT” are guaranteed.

Files used for directory modification

The result file with the name

`HSM.E.<user-id><date><time>.RESULT`

contains the result of the ARCHIVE run initiated for the modification of the archive directory. The result file is created under the caller’s user ID (or, for requests from the HSMS administrator, under SYSHSMS). The result file is deleted after evaluation by HSMS for the generation of a report file.

The report file with the name

`HSM.F.<user-id><date><time>.REPORT`

contains the result of the evaluation of the ARCHIVE result file by HSMS. Just like the result file, it is created under the caller’s user ID (or, for requests from the HSMS administrator, under SYSHSMS).
The report file is deleted after editing and output to the system file SYSLST.

The disk storage space required for the report file depends on the number of objects processed as one record per object is written to the report file.

Temporary files

When a migrated file is recalled, a scratch file with the name

`HSM.D.<user-id><date><time>.<version><cfid>`

is created temporarily under the SYSHSMS user ID and is then deleted after processing.

When the HSMS statement RESTORE-LIBRARY-ELEMENTS is processed, a temporary file with the name

`HSM.P.<user-id><date><time>.PLAM`

is created temporarily under the SYSHSMS user ID and is then deleted after processing.

For log file creation for the BS2000 backup monitor, a temporary file with the following name might be created:

`$SYSHSMS.HSM.C.SYSHSMS.<date_time>`

This temporary file is only created if the following criteria are met:

- Expansion of the user log was necessary (user log is written into the existing file)
- Subsystem CONV2PDF is not available or an error occurred (see also section "Central request monitoring on the SE server")

This file is deleted, once the $SYSHSMS.HSM.C.SYSHSMS.<date_time>.PDF PDF log file has been created.

Work files for backup using the Concurrent Copy function

By default, two temporary work files with the following names are created during backup using the Concurrent Copy function:

`S.DCH.<sysid>.<CC.session-id>.DATA`

`S.DCH.<sysid>.<CC.session-id>.META`

If a name was defined with the WORK-FILE-NAME operand of the HSMS statement BACKUP-FILES, only a work file which is assigned the specified name is created.

ARCHIVE.ASN file for serialization of parallel runs

A new work file with the name $TSOS.ARCHIVE.ASN file is introduced from HSMS V12.0A02.

The ARCHIVE.ASN file guarantees a unique SFID for the save files created on shared S1 storage level or shared SF public disk by several HSMS save tasks running simultaneously in SF environments on different systems.

The ARCHIVE.ASN is created under the userid TSOS on the corresponding output shared SF or S1-SM pubset during the first HSMS save run. The work file stores the last generated ARCHIVE Sequence Number (ASN) from the last save run on the shared pubset.

The ARCHIVE.ASN file is created with a read password and the system administrators can delete it (DELETE-FILE ..., IGNORE-PROTEXTION=*READ-PASSWORD), in case the pubset is no longer used for storing save files.

6.5 Options for enhancing performance

Both users and the HSMS administrator can reduce their backup times, provided they observe the information provided below.

- Restrict the file set effectively by using the functions for file selection
- Decentralized management of SF pubsets
- Do not allow the archive directories to become unnecessarily large and, if required, split them in a sensible manner (for example according to catalog IDs or UFS drives)
- Between full backups, only perform incremental backups or saves after catalog entries have been modified
- Reduce save time through parallel operation of MTC drives
- For BS2000-UFS backup, utilize multiplexing
- Reduce downtime of the application by saving “frozen” copies (BACKUP-FILES with CCOPY)
- Improvement of I/O operations (maximum tape size and linkage of devices)
- In shared pubset operation, have requests executed on the master computer (prevents temporary retry lockouts when sending a large number of catalog operations)
- Reduce report scope as far as possible when large file sets are used (with REPORT= *SUMMARY instead of *FULL)
- Work without restart points (WRITE-CHECKPOINTS=*NO)

6.5.1 Maximum tape block size

To enhance throughput when saving to tape it is recommendable to select large tape blocks (BLOCKING-FACTOR operand). The number of gaps on the tapes are then smaller and tape utilization improves. The tape block size can be selected for each statement, as an archive attribute or as a global parameter in ARCHIVE.

Details are provided in section “Size of tape blocks”.

6.5.2 Expedited copying of save files

Copying of save files can be expedited by not specifying a file selection in //COPY-SAVE-FILE. This enables complex accesses to the archive directory to be avoided.

7 HSMS installation

HSMS must be installed jointly by the system administrator and the HSMS administrator. The system manager has to run the respective IMON installation for HSMS. During this IMON installation, all necessary files for ARCHIVE are also installed.

When the system administrator initially loads HSMS, this is done in the operating mode DEFINE-SHOW, i.e. no action statements are as yet executed by HSMS. The HSMS administrator subsequently sets the control parameters and switches over to OPERATION mode.

Before installing a new HSMS version, the system administrator must find out what is required of the system environment and whether the different HSMS versions are compatible. In section "Compatibility of HSMS definitions in different HSMS environments" there is an explanation of how to perform a version upgrade and downgrade and which HSMS versions are compatible for processing shared pubsets. This section should be read very carefully, as the loss of a control file can also lead to the loss of the entire HSMS environment.

Since HSMS is often introduced in environments which used to use the software product ARCHIVE, the directory files created by ARCHIVE can be retained under HSMS. The directory files created by ARCHIVE can be retained under HSMS (see section "Special aspects when switching from ARCHIVE to HSMS operation").

7.1 Software requirements

- Processing data in BS2000

7.1.1 Processing data in BS2000

HSMS V12.0A can be used with BS2000 OSD/BC V10.0 or higher:

HSMS V12.0A requires the following software environment:

- ARCHIVE V12.0A (for processing action statements)
  Save files that have been created with older ARCHIVE versions can be read by ARCHIVE V12.0A and continued. ARCHIVE is part of HSMS.
- JV (job variables for automatic migration, Concurrent Copy, for processing job variables which have co-owners)
- POSIX (for processing of node files of the BS2000-UFS)
- NFS (to backup/restore node files on passive workstations)
- SECOS (for processing files which have co-owners)
- SHC-OSD (for using Concurrent Copy)
- MAREN (for volume management)
- DAB (for buffering HSMS metadata)

7.2 Scope of HSMS delivery

The following files are part of the release unit HSMS (<ver>=120):

File	Content
SYSPRG.HSMS. <ver>	Load module for starting HSMS (START-PROGRAM)
SYSLNK.HSMS.<ver>	Object module library containing the modules of the TU component of HSMS
SYSLNK.HSMS.<ver>.TPR SKMLNK.HSMS.<ver>.TPR	Object module library with the modules of the HSMS subsystem
SYSLIB.HSMS.<ver>	Library containing the HSMS macro and the object module required for program calls
SYSFHS.HSMS.<ver>	Library containing the FHS screen masks;
SYSMES.HSMS.<ver>	Complete message primary file (MSGMAKER format)
SYSSDF.HSMS.<ver>	System syntax file containing the HSMS statements and statement options for all users. The statements and/or operands which are available only to the HSMS administrator are marked as requiring the “HSMS administration” privilege.
SYSRMS.HSMS.<ver>	File containing corrections to the HSMS subsystem
SYSSSC.HSMS.<ver>	File containing the subsystem declarations for loading HSMS
SYSNRF.HSMS.<ver>	NOREF file for the HSMS subsystem
SYSFGM.HSMS.<ver>.D	Release Notice in German
SYSFGM.HSMS.<ver>.E	Release Notice in English
SYSSII.HSMS.<ver>	File containing information about the IMON installation of HSMS
SYSPRC.HSMS.<ver>	Library with compiled procedures for supporting the BS2000 backup monitor
SYSLIB.ARCHIVE.<ver>	Macro library containing the ARCHIVE macro (for details see the chapter "ARCHIVE macro").
SYSLNK.ARCHIVE.<ver>	Load module library for the nonprivileged part of ARCHIVE.
SYSLNK.ARCHIVE.<ver>.TPR	Load module library for the privileged part of ARCHIVE for use on servers and server units with /390 architecture.
SKMLNK.ARCHIVE.<ver>.TPR	Load module library for the privileged part of ARCHIVE for use on server units with x86 architecture.
SYSNRF.ARCHIVE.<ver>	NOREF file for the ARCHIVE subsystem.
SYSPAR.ARCHIVE.<ver>	Parameter file (SAM) with preset values for most ARCHIVE operands (for details see the section "Parameter file for ARCHIVE").

SYSMES.ARCHIVE.<ver>	Complete messages file containing the ARCHIVE messages (for details see the section "Message files and help texts").
SYSMSH.ARCHIVE.<ver>	ISAM file with help texts for the HELP command in German and English.
SYSPRG.ARCHIVE.<ver> (ARCHIVE)	Load module for compatibility with the command /START-PROGRAM ARCHIVE
SYSPRG.ARCHIVE.<ver>. DIRCONV	Conversion program for directory files.
SYSRMS.ARCHIVE.<ver>	RMS package for the ARCHIVE subsystem.
SYSSII.ARCHIVE.<ver>	File containing information on the structure and installation of ARCHIVE with IMON.
SYSSDF.ARCHIVE.<ver>	Syntax file for statements in SDF format (DIRCONV) and the commands `/START-ARCHIVE` and `/START-DIRCONV` .
SYSSSC.ARCHIVE.<ver>	File containing the subsystem declarations for DSSM

HSMS includes a SYSSII file to be used under IMON. The SYSSII file contains information about the structure and the installation of the product. Among other things, the SYSSII file contains information about the product (release unit) and the release items. Each release item has a logical name that is used during installation. The logical name used is not dependent on the version and is the part of the file name that describes the function.

In the following you will find the assignment of logical names to the release items of HSMS:

Structure of the release unit HSMS V12.0

Release item	Logical name
SYSPRG.HSMS.<ver>	SYSPRG
SYSFGM.HSMS.<ver>.E	SYSFGM.E
SYSFGM.HSMS.<ver>.D	SYSFGM.D
SYSFHS.HSMS.<ver>	SYSFHS
SYSLIB.HSMS.<ver>	SYSLIB
SYSLNK.HSMS.<ver>	SYSLNK
SYSLNK.HSMS.<ver>.TPR SKMLNK.HSMS.<ver>.TPR	SYSLNK.TPR
SYSMES.HSMS.<ver>	SYSMES
SYSREP.HSMS.<ver>	SYSREP
SYSRMS.HSMS.<ver>	SYSRMS
SYSNRF.HSMS.<ver>	SYSNRF
SYSSDF.HSMS.<ver>	SYSSDF
SYSSSC.HSMS.<ver>	SYSSSC
SYSPRC.HSMS.<ver>	SYSPRC

Structure of the release unit ARCHIVE V12.0

Release item	Logical name
SYSLIB.ARCHIVE.<ver>	SYSLIB
SYSLNK.ARCHIVE.<ver>	SYSLNK
SYSLNK.ARCHIVE.<ver>.TPR SKMLNK.ARCHIVE.<ver>.TPR	SYSLNK.TPR
SYSNRF.ARCHIVE.<ver>	SYSNRF
SYSPAR.ARCHIVE.<ver>	SYSPAR
SYSMES.ARCHIVE.<ver>	SYSMES

SYSMSH.ARCHIVE.<ver>	SYSMSH
SYSPRG.ARCHIVE.<ver> (ARCHIVE)	SYSPRG
SYSPRG.ARCHIVE.<ver>.DIRCONV	SYSPRG.DIRCONV
SYSRMS.ARCHIVE.<ver>	SYSRMS
SYSSDF.ARCHIVE.<ver>	SYSSDF
SYSSSC.ARCHIVE.<ver>	SYSSSC

For more information on release items, release units and logical names, see the “IMON” manual [7].

7.3 Compatibility of HSMS definitions in different HSMS environments

Before installing a new HSMS version you must check the compatibility of the various HSMS versions. With HSMS the version status is particularly important for

- the archive definitions stored in the control file.
- the request definitions stored in the request file.

You must also take the following into account:

- If shared pubsets are installed:
  the criteria for transferring requests from the slave host to the master host (see section "Shared pubsets and different HSMS versions").
- If SM pubsets are installed:
  the HSMS control and request files stored on these SM pubsets.

7.3.1 SF control file (home pubset)

The SF control file of HSMS can always be continued. An SF control file that was created by a lower HSMS version can always be read and continued by a higher HSMS version. The file is continued automatically on generation of the subsystem.

The SF control file, however, is not downwardly compatible. An SF control file that was created by a higher HSMS version cannot be read by a lower HSMS version. You are therefore strongly advised to make a copy of the current SF control file before a new HSMS version is started. The copy of the control file is very important in the event of a downgrade.

7.3.2 SF request file (home pubset)

The SF request file of HSMS is not compatible. It must therefore be deleted or emptied before a new HSMS version is started.

You empty an SF request file by deleting all the requests it contains. This is done using the HSMS statement DELETE-REQUESTS of the HSMS version that created the requests.

If on generation of the subsystem an incompatible request is found in the request file, the system cannot be booted.

7.3.3 Shared pubsets and different HSMS versions

For requests to shared pubsets it is generally the case that a slave sharer sends the requests to the master sharer for processing if BACKUP-SERVER-USAGE=*NO is specified in the archive definition, If *STD is specified, depending on the HSMS parameter BACKUP-SERVER the requests are sent to the master sharer or to the backup server which is specified explicitly.

HSMS supports the sending of requests without restriction if an earlier HSMS version is being used on the slave sharer than on the master sharer. In HSMS < V10.0 the requests are sent to the master sharer as the backup server function is not supported in these versions.

If an earlier HSMS version is being used on the master sharer than on the slave sharer, the request will only be accepted if the master sharer supports all the functions that are necessary for correct processing of the request (see section "Working with SM pubsets and different HSMS versions").

7.3.3.1 Working with S1-SM pubsets in different HSMS versions

Using S1-SM pubsets is only possible, if all accessing systems are running HSMS V10.0A or higher and the HSMS parameter SAVE-FILE-PROCESSING is set to *HSMS-V10-COMPATIBLE.

7.3.4 Control and request files on SM pubsets

SM pubsets can be transported from host to host or shared by different HSMS versions. Therefore the control and request files on the SM pubsets are both upwardly and downwardly compatible.

> ℹ During HSMS startup processing and HSMS import processing, the request files are accessible after a maximum wait time of 10 minutes.

An archive definition or a request created by any HSMS version can be read by all other HSMS versions. But the archive definition/the request can only be processed if the HSMS version concerned supports the requested function (see section "Working with SM pubsets and different HSMS versions").

7.3.5 Working with the extended storage level S1 in an SM environment

- Setup of the extended S1 level
- Backup, version backup archiving and migrating to the extended S1 level
- Reducing extended storage level S1 to one volume set again
- Compatibility with earlier versions or in the case of an inhomogeneous environment

7.3.5.1 Setup of the extended S1 level

Storage level S1 can be defined for an SM environment using the CREATE-SM-PUBSET-PARAMETERS or MODIFY-SM-PUBSET-PARAMETERS statement.

In this case, S1-VOLUME-SET=<cat-id> explicitly defines the specified volume set as S1 level. In HSMS V11.0 and higher, S1-VOLUME-SET=*ALL-HSMS-CONTROLLED can be used to define an extended storage level S1. All volume sets of the SM pubset which are under HSMS control are thus available as S1 level.

The following conditions are required for using the extended S1 level:

- BS2000 OSD/BC V11.0 and HSMS V11.0 or higher is used on the system. In the case of a shared pubset network, all pubset sharers of the SM pubset must satisfy this requirement.
- The value *HSMS-V10-COMPATIBLE must be specified for the SAVE-FILE-PROCESSING HSMS parameter. In case of a shared pubset group, this applies to all systems accessing the pubset.

> i All volume sets that belong to the S1 level need to be of NK2 format.

The CREATE-SM-PUBSET-PARAMETERS or MODIFY-SM-PUBSET-PARAMETERS statement is rejected in the following cases:

- If no volume set of the SM pubset is planned for HSMS usage, the specification is rejected by *ALL-HSMS-CONTROLLED.
- On a BS2000 system with BS2000 OSD/BC earlier than V11, the specification of *ALL-HSMS-CONTROLLED is rejected.

As soon as an extended S1 storage level has been set up on an SM pubset, systems with HSMS V10 can no longer take HSMS control over the SM pubset.

7.3.5.2 Backup, version backup archiving and migrating to the extended S1 level

After the extended S1 level has been set up as set out under section "Setup of the extended S1 level", the S1 level can be used as before. The statements for backup, version backup, archiving and migration do not have a special syntax for using the extended S1 level.

The storage of the save files is controlled by the allocator and cannot be influenced by the user. However, the system manager can deny the creation of new save files on certain volume sets with the following command:

```
/MODIFY-PUBSET-RESTRICTIONS PUBSET=<sm_pubset_catid>,
   PUBSET-TYPE=*S-M(VOL-SET=<volset_catid>,
   RESTRICTION=*NEW-FILE-ALLOCATION)
```

7.3.5.3 Reducing extended storage level S1 to one volume set again

If the extended storage level S1 has already been used for a while, various save files can be distributed over multiple volume sets. Therefore, the extended storage level S1 must be empty, before it can be reduced to a single volume set.

The following statement is required to reduce the extended storage level S1 to the volume set *VS01* for an SM pubset *SM01*:

```
//MODIFY-SM-PUBSET-PARAMETERS SM-PUBSET-ID=sm01,
                              S1-DEFINITION=*PAR(S1-VOLUME-SET=vs01
```

HSMS checks whether there are save files on volume sets controlled by HSMS. If this is the case, the MODIFY-SM-PUBSET-PARAMETERS statement is rejected with the HSM0500 message.
This also applies if the S1 level is to be completely removed with SM-PUBSET-ID=*UNDEFINED.

If it becomes necessary to switch to one volume set after the extended storage level S1 has been used, the administrator must empty the S1 level beforehand:

- Migrate the migration archives from S1 to S2 using the MIGRATE-FILES statement. Save files which contain no invalid files are not reorganized in this case.
- Copy backup and long-term archives from S1 to S2 using the COPY-SAVE-FILE statement. Then delete the save file from S1 with the following statement.
- Reorganize version backup archives to level S2 using the REORGANIZE-VERSION-BACKUP statement.

```
//MODIFY-ARCHIVE ... SAVE-FILE=*DELETE
```

7.3.5.4 Compatibility with earlier versions or in the case of an inhomogeneous environment

Before switching to the extended S1 level, ensure that all BS2000 systems accessing the SM run BS2000 OSD/BC V11.0 or higher and HSMS V11.0 or higher.

As soon as the extended S1 level for an SM pubset has been set up on a pubset sharer with HSMS V11.0A or higher, all pubset sharers running HSMS V10.0A or lower can no longer take HSMS control over the pubset – neither with IMPORT-PUBSET nor CREATE-SM-PUBSET-PARAMETERS.

In case an SM pubset was used on a pubset sharer with HSMS V10.0 before the extended S1 level for this pubset has been set up on a different sharer, this pubset remains under HSMS V10 control. However, the following functions are no longer available: backup, copying and migrating to S1 and changing pubset parameters. Write access to the save files on S1 level is still possible: recovery of saved files, restoring of migrated files, copying of save files to S2 level, migrating files from S1 to S2, changing files in the S1 level and deleting files from the S1 level.

Systems with HSMS V9.0 or lower can under no circumstances access the files in the extended S1 level.

Information output with SHOW-SM-PUBSET-PARAMETERS

On a pubset sharer with HSMS earlier than V11.0, with the extended S1 level the SHOW-SM-PUBSET-PARAMETERS displays the value **`NOT-DEFINED`** instead of **`ALL-HSMS-CONTROLLED`**. The HSMS VERSION field has the value 110.

Information output with SHOW-PUBSET-USAGE

If an extended S1 level for the pubset has been set up on a pubset sharer (with HSMS V11.0 or higher), on a sharer of the same pubset with HSMS prior to V11.0 the statement SHOW-PUBSET-USAGE acts as if no S1 level is defined:

- with INFORMATION=*REUSABLE-S1-SPACE the statement is rejected with the HSM0213 message.
- with INFORMATION=*MIGRATION-EVALUATION, files that have been migrated to S1 level are marked with 'S1-NOT-DEF'.

7.3.6 Version backup in shared environment

In an inhomogeneous BS2000/HSMS environment a //BACKUP-FILE-VERSIONS will not be transferred to a master (backup server) running BS2000 OSD/BC < V11.0B or HSMS < V12.0A. In this case, the statement //BACKUP-FILE-VERSIONS is aborted with a corresponding message on the host that starts the request.

Notes:

- It is strongly recommended to set the same value of the system parameter NUMBACK on all sharers with BS2000 OSD/BC version as of V11.0B. Otherwise newly created files will have different NUMBER-OF-BACKUP-VERSIONS depending on the system, where they were created.
- //BACKUP-FILE-VERSIONS is only possible in *HSMS-V10-COMPATIBLE mode.
- Restoring from version backup archives is possible only by HSMS V12.0A or higher. In case slave/master or backup server, the processing system must have HSMS V12.0A or higher installed.
- Restoring from version backup archives is only possible with BS2000 OSD/BC V11.0B or higher.
- //CHECK-CATALOG-FILES is only possible on systems with BS2000 OSD/BC V11.0B or higher.
- //REORGANIZE-VERSION-BACKUP requires BS2000 OSD/BC V11.0B or higher and is only possible in *HSMS-V10-COMPATIBLE mode.

Compatibility in a shared SM environment.

As soon as a system archive for version backup (SYSVERSION) is defined for an SM-pubset on a sharer with HSMS V12.0A or higher, modifying the SM-pubset parameters via MODIFY-SM-PUBSET-PARAMETERS is no longer possible on other sharers with lower HSMS versions. The function is again available when the sysversion backup archive is again set to *UNDEFINED.

7.4 User ID SYSHSMS

The user ID SYSHSMS is essential to HSMS operation. It is to this user ID that the product files of HSMS must be read, and under this user ID that HSMS creates its central work files. The user ID SYSHSMS is automatically set up by the system during initial system startup. The system administrator can use the MODIFY-USER-ATTRIBUTES command to modify the attributes of this user ID. The default pubset ID (DEFAULT-PUBSET) under the SYSHSMS user ID should not be placed on a shared pubset.

PUBLIC-SPACE-LIMIT should be set high enough to allow all necessary files to be created. In order to prevent an abort when larger work files are created, it should be possible to exceed this limit (PUBLIC-SPACE-EXCESS). This applies in particular when the HSMS migration function is used. The automatic recall mechanism recalls a migrated file first under a temporary file on the SYSHSMS user ID on the migrated file’s pubset before the allocation of the file is changed to the final catalog entry. In order to recall very large files, sufficient storage space must be available for SYSHSMS.

When the PUBLIC-SPACE-LIMIT is defined, it must also be borne in mind that the protocol files for the display in the BS2000 backup monitor of the SE Manager are also stored under the SYSHSMS ID. When the request is deleted, the PDF file is also deleted.

7.5 Creating an environment for backing up remote nodes S0

To back up node files with NFS, the system administrator must perform the following actions to create the required environment:

- The “HSMS” directory must be defined under the root directory in the local BS2000-UFS.
- A separate directory must be created under the “HSMS” directory for each remote computer to be accessed by HSMS. Each directory must be assigned the name of the remote computer it represents.

7.6 Initial startup of HSMS

The initial startup of HSMS is carried out as a normal startup of an HSMS session except that usually the control and request files are missing or are still available from a previous HSMS version. The normal startup processing, however, takes this into account.

At the start of each HSMS session, the system checks for all control and request files in the following order:

1. First the central control file is checked and read into internal tables.
   - If no central control file is available, a new one is created. It already contains presettings which are then used for the start.
   - If the central control file is available and contains definitions dating from an earlier HSMS version, the definitions are recognized and automatically adjusted. Downgrading to a lower version is not supported!
   - If the central control file contains invalid records or is inconsistent, the start is aborted. Additionally, error messages are output to the operator terminal.
2. The central request file is then checked.
   - If the central request file is not available, a new empty request file is created.
   - If the central request file is available and contains data records that date from an earlier HSMS version, the start is aborted with errors. The request file cannot be adjusted and must therefore be deleted before the startup processing is restarted.
3. Next the local control file and the local request file of each accessible SM pubset are checked. First the local request file is checked and written to internal tables.
   - If no local control file is available for a specific SM pubset environment, a new one is created. This file already contains presettings which are used to start the SM pubset environment.
   - If a local control file is available and contains definitions that date from an earlier HSMS version, the definitions are recognized and automatically adjusted.
   - If a local control file contains invalid records or is inconsistent, the local start is aborted and the SM pubset environment is skipped. The start continues with the next SM pubset environment.
   - If an SM pubset under HSMS control cannot be accessed during startup of the HSMS session, it is skipped. It is started, however, when the SM pubset is imported to the host. There the same start is carried out (with checking and reading of the local request file).

Once the HSMS start is completed, the global HSMS parameter is adjusted to the requirements of the respective system by means of the HSMS statement:

```
//MODIFY-HSMS-PARAMETERS VALID-PERIOD=*PERMANENT,...
```

The presettings after the initial startup of HSMS are described at the end of the HSMS statement MODIFY-HSMS-PARAMETERS in “HSMS Vol. 2” [1].

The SM pubset-specific parameters can be changed with the following HSMS statement:

```
//MODIFY-SM-PUBSET-PARAMETERS
```

The presettings for the control file of an SM pubset are described in “HSMS Vol. 2” [1] under the HSMS statement MODIFY-SM-PUBSET-PARAMETERS.

Subsequent step-by-step transition is then possible from the SIMULATION operating mode in which action statements are only simulated, i.e. undesired data manipulation is prevented, to the normal OPERATION mode. In this mode, archives can be created and pubsets placed under HSMS management.

After initial startup of HSMS, only the default pubset of the SYSHSMS user ID is under HSMS management. It is allocated to storage level S0.
For all other pubsets, the default values which are listed in "HSMS Vol. 2” [1] for the HSMS statement MODIFY-PUBSET-PARAMETERS apply.
Integration of the SF pubset in HSMS management is described in section "Managing SF pubsets".
In the section "Processing SM pubsets" you will find a description of how SM pubsets are placed under HSMS management.

7.7 Actions to be performed at each BS2000 system startup

At each BS2000 system startup, the system administrator has to mount each remote file system to be processed by HSMS on the appropriate node in the local BS2000-UFS. For this purpose, any file system to be mounted must have been declared shareable at the computer which is the owner of the file system.

If a file system resides on a new remote computer, the system administrator has to define an appropriate node under the “HSMS” directory in the local BS2000-UFS.

7.8 Notes on introducing the migration function

The following activities must be carried out when introducing migration:

- The migration function cannot be used for all files. The HSMS administrator should observe the notes in section "Except file", and all users should observe the notes in section "File migration inhibit".
- Please note that in many cases the files stay migrated for a long period of time. It is recommended to only migrate files that have already been saved (this is ensured by the HSMS parameter MIGRATION-CONTROL (BACKUP-MANDATORY=*YES), however, these backups become obsolete after the retention period expires and are deleted from the backup archive (//MODIFY-ARCHIVE SAVE-FILE=*DELETE). From this point in time, there is no existing copy of the file.

  Therefore migrated files should be backed up from time to time, if the backup timeframe allows it: BACKUP-FILES ... SAVE-OPTIONS=*PARAMETERS(SAVE-DATA=*S2-S2-S0). This ensures that there are always copies of the data of all storage levels. Version backup can also be used for the purpose.
- Aside from this, HSMS should be started as soon as possible after system initialization to ensure that the files are recalled as quickly as possible.
- Jobs which previously used /SHOW-FILE-ATTRIBUTES or /ADD-FILE-LINK to check the presence of files needed during the program run should use /SECURE-RESOURCE-ALLOCATION with a wait time instead (see section "Implicit recall on file access").
- Applications which use the internal file name (CFID) for checking and processing files (e.g. UPAM), must take the following into account: comparing the catalog entry before and after OPEN is not permitted, since opening on a migrated file recalls the latter, thereby changing the internal file name (CFID). If an FSTAT macro is issued before processing, the catalog entry should not be noted for a check. It is better to use /SECURE-RESOURCE-ALLOCATION with a wait time, since this will recall the file.
- For the SYSHSMS user ID, the storage space (PUBLIC-SPACE-LIMIT) should not be limited if very large files are recalled implicitly. When it is recalled, the file is first imported to the storage space of the SYSHSMS user ID on the pubset on which it was located prior to migration before the assignment is changed to the final user ID. When creating the SYSHSMS user ID, the storage space extension (PUBLIC-SPACE-EXCESS) should therefore be allowed.

Note

The appendix contains summarized information which all users should receive before the migration function is introduced. The restriction on reproduction therefore does not apply to the appendix. The system administrator can distribute copies of the appendix to future users and/or users affected by the migration function.

8 Interoperation of HSMS with ARCHIVE

HSMS interoperates with the BS2000 software product ARCHIVE.

8.1 ARCHIVE functions in HSMS

HSMS is based on the software product ARCHIVE. HSMS offers the functions supplied by ARCHIVE, but via a more modern user interface and with additional control features.

HSMS makes a clear distinction between the two functions backup and archival, which are not differentiated under ARCHIVE. HSMS offers separate statements for the individual functions as well as separate management of the corresponding data.

Most ARCHIVE functions are also supported by HSMS. But since the HSMS user interface is different, the same functions must be invoked by means of different statements under HSMS. The following table is designed to help users who are familiar with the software product ARCHIVE by giving a few examples of how to map ARCHIVE statements and operands onto their HSMS equivalents.

The second table maps the operands of the ARCHIVE statement PARAM and appropriate operands of HSMS statements.

ARCHIVE continues to be available to all HSMS users.

The following functions are only possible with HSMS:

- all actions affecting migration and node files
- updating of backups (Several-SVID format)
- version backups
- some additional functions such as the restoration of library elements (made possible by saving the metainformation of PLAM libraries) or all functions linked to archive atributes
- via the "select only files on Net-Storage" or "exclude files on Net-Storage" operand (control possible only via the parameter file for the entire ARCHIVE session, see “Saving Net-Storage files” in section "Restrictions due to new functions").

Device reservation in HSMS

HSMS reserves the tape devices required for the run itself. Tape devices that are reserved by a user in their own task prior to an HSMS run, are not available to HSMS.

Table of ARCHIVE and HSMS statements

ARCHIVE statement	**HSMS statement**
DELETE	DELETE-REQUESTS
EXPORT	EXPORT-FILES
,DIRSAVE=*YES	,DIRECTORY-NAME=<filename> – (SAVE-DIRECTORY=*YES)
IMPORT	IMPORT-FILES
FILES	SELECT-FILE-NAMES statement and FILE-NAMES operand (SELECT-FILES operand of COPY-SAVE-FILE)
,FROM=*PRDISC	BACKUP-FILES SUPPORT=*PRIVATE-DISK (VOLUMES=*ALL)
,FROM=*PRTAPE	BACKUP-FILES SUPPORT=*TAPE
INQUIRE	SHOW-ARCHIVE
JOBVAR	SELECT-JV-NAMES and operand JV-NAMES (SELECT-JV operand of COPY-SAVE-FILE)
LIST	LIST-VOLUMES
POOL	MODIFY-ARCHIVE VOLUMES=*ADD/*REMOVE
PURGE	MODIFY-ARCHIVE SAVE-FILES=*DELETE
PROCESS	RESTART-REQUESTS
RESTORE	RESTORE-FILES
,FROM=*LATEST	,SELECT-SAVE-VERSIONS=*ALL
,FROM=*LATEST,*STATE	,SELECT-SAVE-VERSIONS=*LATEST
SAVE	ARCHIVE-FILES, BACKUP-FILES
,CHANGED=*NO	BACKUP-FILES SELECT-FILES=*ALL-FILES
,CHANGED=*YES,*LARGE	BACKUP-FILES SELECT-FILES=*MODIFIED-FILES (PARTIAL-FILE-SAVE=*LARGE-FILES)
,DIRSAVE=*YES	BACKUP-FILES SAVE-OPTIONS=*PAR (SAVE-DIRECTORY=*YES)
,DRIVES=	BACKUP-FILES OPERATION-CONTROL (PARALLEL-RUNS=)

STATUS	SHOW-REQUESTS

Table 6: Mapping ARCHIVE statements onto HSMS statements

PARAM operand of ARCHIVE statement	HSMS operand
CNS=*YES/*NO	REPORT=*FULL/*SUMMARY
RESTART=*YES/*NO	WRITE-CHECKPOINTS=*YES/*NO
UNLOAD=*YES/*NO	UNLOAD-TAPE=*YES/*NO
OPERATOR=*YES/*NO	OPERATOR-INTERACTION=*YES/*NO
WRCHK=*YES/*NO	WRITE-CHECK=*YES/*NO
SNR=*YES/*NO	REPORT=*FULL/*RESTORED-FILES
DESTROY=*YES/*NO	DESTROY-BY-DELETE=*YES/*NO
CATID=*YES/*NO	CATALOG-ID-MODE=*YES/*NO (only for export/import, otherwise HSMS works only with catalog ID)
OLS=*YES/*NO	SAVE-OPTIONS=*PAR (SAVE-ONLINE-FILES=*YES/*NO) operand of BACKUP-FILES

Table 7: Mapping PARAM operands onto HSMS operands

8.2 Special aspects when switching from ARCHIVE to HSMS operation

Retention of ARCHIVE directory files

Since HSMS is often introduced in environments which used to use the software product ARCHIVE, the directory files created by ARCHIVE can be retained under HSMS. For this purpose, the directory file has to permit catalog IDs, i.e. must have been created by means of the ARCHIVE statement PARAM CATID=*YES. The DIRCONV routine, which is supplied together with ARCHIVE, serves to convert a directory file without catalog ID not created with CATID=*YES to one that permits catalog IDs.

HSMS can continue to use the existing volume pool and save files created using ARCHIVE. This is implemented by creating an HSMS archive: The old ARCHIVE directory file can be assigned to this HSMS archive as the archive directory. This is done using:

```
//CREATE-ARCHIVE ARCHIVE-NAME=<archive name>, -
//  DIRECTORY-NAME=<file name>(NEW-DIRECTORY=*NO)
```

The complete scope of HSMS functions is then available to this archive.

ARCHIVE directory files should be transferred to backup archives only, since this is subject to the fewest restrictions.

Conversion of export backups without directory to Several-SVID backups with directory

When switching from magnetic tape devices with small cartridges to devices with very large cartridges (e.g. LTO) it is also advisable to switch from ARCHIVE to HSMS operation since HSMS permits backups to be updated (Several-SVID format). It should also be possible to copy export backups without directory which were created on small volumes in ARCHIVE operation to updated HSMS long-term backups on large volumes. This special function for the switchover from ARCHIVE to HSMS operation is available as of HSMS V8.0A in the COPY-EXPORT-SAVE-FILE statement. Here the input save file can be specified as a volume list and the output save file in Several-SVID format with the option of continuation.

The directory created during COPY-EXPORT-SAVE-FILE can be specified as an HSMS archive directory when creating a long-term archive.

8.3 Restrictions on reverting to ARCHIVE

This section describes a number of restrictions which must be taken into consideration in connection with reversion to ARCHIVE.

8.3.1 Compatibility of directory files

With the transition from ARCHIVE to HSMS, existing ARCHIVE directory files can be included in new HSMS backup archives that are to be created. If the directory file has been included with the SAVE-FILE-STRUCTURE= *SEVERAL-SVID archive attribute and backups have been created under HSMS, reverting to ARCHIVE using this directory file is no longer possible. In general, archive directories of backup archives in SEVERAL-SVID structure as well as version backup, migration and long-term archives should under no circumstance be used under ARCHIVE.

8.3.2 Restrictions due to new functions

A number of restrictions on reverting to ARCHIVE are caused by the following new HSMS-specific functions:

- New protection mechanisms
- Updating (continuation) of save volumes
- Backup to S1 pubsets or volume sets
- Introduction of version backup
- Introduction of migrated files
- Backing up node files of the BS2000-UFS (POSIX) or of remote nodes S0 via POSIX
- Backup/restoration of files for which there are co-owners
- Saving files on Net-Storage

The restrictions governing reversion to ARCHIVE usually depend on the ARCHIVE version involved. They are described below.

Restrictions due to different protection mechanisms

Since HSMS save requests are processed by a privileged server task under TSOS (and not, as in ARCHIVE, by a subtask under the user's own ID), access to save copies of files under the user's own ID is not possible without further action.

Provided the user is working under the ID under which the directory file is cataloged or under TSOS, ARCHIVE can be used to restore files from that directory file. Otherwise, ARCHIVE can access only those save files in a directory file under TSOS that were created as shareable files (see "HSMS Vol. 2" [1], BACKUP-FILES statement, USER-ACCESS=*ALL-USER operand).

This means that a directory file created by HSMS under the user ID SYSHSMS may have to be copied to TSOS.

Restrictions due to tape continuation

With tape backups in SEVERAL-SVID format and generally in long-term archiving and migration, HSMS creates backup files in a way that they can consist of more than one save version. This affects the possibility of reverting to ARCHIVE.

There are no restrictions on read access to the save volumes under the user ID TSOS. However, an SVID must be specified when accessing volumes containing more than one save version. The LIST statement can only be used if the SVIDs on the volume are in ascending alphabetical order.

Directory file processing is supported in IMPORT/EXPORT runs only. Multiple save copies of a file in one and the same save file can be accessed directly and restored using IMPORT.

ARCHIVE does not support the continuation of save files with more than one save version.

Restrictions due to backup to pubsets

HSMS permits backup to (S1) pubsets. This has the following consequences for reverting to ARCHIVE:

With a directory file, restoring and listing are possible without restrictions, if the backup has been performed under HSMS with the setting SAVE-FILE-PROCESSING=*HSMS-V9-COMPATIBLE.
Access to a save file without a directory file is possible only if the save file resides on the home pubset. If this is not the case, it must first be copied to it.

Restrictions due to file migration

HSMS introduces the concept of file migration. This has the following consequences for reverting to ARCHIVE:

Any EXPORT referring to a migrated file or any SAVE referring to a user ID != TSOS is rejected with an error message.
The catalog entries of migrated files can only be saved under the user ID TSOS; renaming during the restore is not permitted. The SPACE operand is ignored.

Restoring files from node S0:

HSMS is an essential requirement to restore files of the BS2000-UFS (POSIX) or of remote nodes S0 (e.g. UNIX workstations) which were mounted via POSIX and saved with HSMS (BACKUP-NODE-FILES, ARCHIVE-NODE-FILES).

Saving Net-Storage files

When HSMS is saved you can define which files are to be saved at statement level:

- files on pubset, Net-Storage and private disk
- only files on pubset and private disk
- only files on Net-Storage

In ARCHIVE this behavior cannot be controlled at statement level. The STORAGE-TYPE parameter in the parameter file SYSPAR.ARCHIVE offers the following options for controlling the entire ARCHIVE session:

- **`STORAGE-TYPE=ANY`** or no parameter specified

  If the FROM operand was not specified in the FILES statement, all the files specified in the NAME operand are saved which reside on pubset, private disk, or Net-Storage. Only the catalog entries of tape files are saved.

  If FROM=PUBLIC was specified, only files which are specified in the NAME operand and which reside on public volumes or on Net-Storage are saved.

- **`STORAGE-TYPE=PUBLIC-SPACE`**

  If the FROM operand was not specified in the FILES statement, all the files specified in the NAME operand are saved which reside on pubset or private disk. Only the catalog entries of tape files are saved.

  If FROM=PUBLIC was specified, only files which are specified in the NAME operand and which reside on public volumes are saved.

  As files on Net-Storages are excluded from the backup in both cases, data security for these files can only be provided in this case by backing up the file system concerned to the net server itself.

The same applies for the TO operand of the FILES statement for reconstruction:

- If STORAGE-TYPE = ANY is specified and the TO operand is missing, the files are restored to their original media. When TO=PUBLIC, all files, including files from private volumes, are written back to public volumes. In this case files that were saved from Net-Storage are restored to Net-Storage. Files on Net-Storage are public files.
- If STORAGE-TYPE = PUBLIC-SPACE is specified, all files are restored on the defined pubset, irrespective of the medium they have been saved from. ARCHIVE does not save the SAM structure of SAM node files, therefore, these files can only be restored on Net-Storage with the node file type. ARCHIVE outputs a message into the log file for each file that could not be processed on the public volume due to this.

9 Messages

This chapter contains an overview of the HSMS message classes and notes on messages output by other products which are displayed in unchanged form when HSMS is invoked.

The command/standard statement HELP-MSG-INFORMATION enables you to query further information on the meaning of individual messages.

You can find all HSMS messages using the HTML application "System messages" on the manual server (URL: **http://bs2manuals.ts.fujitsu.com**) and on the "BS2000/OSD SoftBooks" DVD.

9.1 Message classes

HSMS outputs messages with a 7-character message code. The message ID is “HSM”. All HSMS messages have the format “HSM0ynn”, where “y” indicates the message class and nn is the sequence number within the message class.

HSMS distinguishes between the following message classes:

Message class	Type of message
Class 0	Result message
Class 1	Console message
Class 2	Error message and rejection of a statement
Class 4	Warning or error message (during processing of a statement)
Class 9	Error message with memory dump

Messages in classes 0, 2 and 4 are statement-specific in terms of their meaning and effect.

The messages of classes 1 and 9 are of interest for the HSMS administrator (and the operator) only; not for nonprivileged users.

The location at which HSMS messages are displayed depends primarily on the system environment of the respective task (interactive or batch mode, user request, server task or main task).

HSMS outputs messages to the following destinations:

Type of task	Message class	Output destination
Interactive user task	0, 2, 4	SYSOUT
	9	Console
Batch user task	0, 2, 4	SYSLST
	9	Console
Server task	4	Report
	1, 9	Console, SYSOUT
Main task	9	Console

Result messages (class 0)

Result messages are output in order to supply a user with information about the result of statement processing. Normally (i.e. with asynchronous processing), this is a temporary result.

The default weight of class 0 messages is “10”; a weight of “50” indicates that statement processing has not been terminated successfully.

Console messages (class 1)

Console message are output in order to provide the operator and the HSMS administrator with information on the HSMS status or an error situation in HSMS whose cause is known. The messages output during HSMS loading are typical examples of such messages.

Class 1 messages are normally output by the HSMS task or the HSMS server tasks; they are output to the console with routing code “U”. Messages from server tasks are additionally output to SYSOUT.

Error messages with rejection (class 2)

Class 2 error messages are output if HSMS rejects a statement for any of the following reasons:

- invalid input
- missing privilege
- unavailable resources (no tape session)

The default weight of class 2 messages is “99”.

Messages during processing (class 4)

Class 4 messages are output during processing of an HSMS statement. They typically report an error which has occurred during processing or the reason why processing is aborted. Class 4 messages can be classified as follows:

- Warnings indicate that an object specified by the user (file, volume, etc.) cannot be processed (object not available, user not privileged).
  Processing is normally continued after output of the warning.
  The default weight is “30” (normal error).
- Error messages after which processing is continued indicate that an object specified by the user cannot be processed or can only be partially processed due to an error.
  The default weight is “50” (important error).
- Error messages after which processing is aborted indicate that an error has occurred in one of the basic HSMS functions which rendered processing impossible.
  The default weight is “99”.

Error messages with dump (class 9)

Class 9 error messages indicate that an error occurred in HSMS whose cause is not known; error documentation is produced for diagnostic purposes.

Class 9 messages are output to the console with routing code “U”. Messages from interactive tasks are additionally output to SYSOUT, those from batch tasks to SYSLST.
The default weight of class 9 messages is “99”.

9.2 Messages output by other products

When HSMS is invoked, messages output by the following products are passed on to the user unchanged:

- ARCHIVE messages
  ARCHIVE messages sent to the console are displayed in unchanged form. All other ARCHIVE messages are logged in a report list for the ARCHIVE run. The report list is created when HSMS statements which invoke ARCHIVE are used.
- SDF messages
  These are sent to SYSOUT when SDF detects a syntax error.
- DSSM messages
  These are displayed when HSMS starts up and shuts down.

10 Accounting record

HSMS provides a precise accounting procedure:

Users pay for exactly the amount of memory and CPU time they have used and the number of inputs and outputs.

10.1 CPU time and inputs/outputs for HSMS activities

This accounting type distinguishes between DEFINE-SHOW mode (see also "DEFINE-SHOW mode") and the action statements:

DEFINE-SHOW mode

There is no new accounting procedure for DEFINE-SHOW mode, as the resources used are already calculated in the current BS2000 accounting procedure.

Action statements

For action statements (requests), special new accounting records are written to the accounting file. These accounting records contain information about the HSMS user, the CPU time used and the number of inputs/outputs. They also log the date and time of the request for which the accounting records are written.

The accounting records are written for all activities which result in a request. These are:

- backup/restore for DMS and UFS runs
- archival/restore for DMS and UFS runs
- migration/recall for DMS runs
- export/import for DMS runs
- copying of save files for DMS and UFS runs
- updating of backups (UPDATE-EXPORT-SAVE-FILE statement) for DMS runs
- all administrative activities which involve a server task (REPAIR-CATALOG-BY-RESTORE and REPAIR-SAVE-FILE-BY-RESTORE statements)

The accounting values for the following four tasks are measured separately for each request:

- the user task
- the server task
- the ARCHIVE subtasks
- the communication task

At the beginning and end of each task, an accounting record is written. The accounting record contains the values measured at the start and end of the task. To obtain the amount of resources currently being consumed for a particular task, you must calculate the difference between the values in the first and second accounting records.

The number of accounting records written for each request depends on the type of the request:

1. For each individual “normal” HSMS request, at least six accounting records are written:
   - 2 for the user task
   - 2 for the server task
   - 2 for each generated ARCHIVE subtask.

   The accounting records are identified by the user ID of the user who issued the request, the TSN and the account number of the user task.
2. For a collector request, the following accounting records are written:
   - 2 for each request in the user task
   - 2 for the total collector request in the server task, and
   - 2 for each generated ARCHIVE subtask.

   All requests are processed independently of each other in the user task. Therefore the accounting records are identified by the user ID of the user who issued the request, the TSN and the account number of the user task.

   The resources used by a server task affect several collector requests and therefore often various users also. They are registered under the TSOS user ID. Accounting records for a server task are identified by the TSOS user ID, the TSN of the server task and the account numbers for server tasks specified in the HSMS parameters. If this account number is not defined, “ADMINSTR” is used.

   To enable the resources used by each collector request in the server task to be calculated correctly, the accounting records for the server task contain an additional extension. This extension contains the user IDs, the account numbers and the TSNs and account numbers of the collector requests. The resources consumed by the server task are distributed equally among all collector requests and included in the accounting for the affected collector user IDs. Two accounting records for each catalog ID and user ID processed are written to the ARCHIVE subtasks.
3. The accounting records of node requests which come from a workstation are identified by the SYSHSMS user ID, the TSN and the account number of the communication subtask.
4. For requests which affect the objects on a shared pubset and which are processed on the master host, one accounting record is created on the slave host and one on the master host.
5. For implicit recall requests that are sent to a master host, the request date and time are not entered, as no request is generated on the slave host.
6. With a RESTART-REQUESTS statement that restarts several requests, the following accounting records are written:
   - only 2 for all requests in the user task, whereby the request date and time are not entered
   - 2 for restarted requests in the server task
   - 2 for a generated ARCHIVE subtask

   A RESTART-REQUESTS statement which only affects one request is processed like a single “normal” request (see number 1).

Format of the HSMS accounting record:

(A) Record description (length: 20 bytes)

Field no.	Distance		Length	Format	Meaning
	hex	dec	(byte)		
1	00	0	4	A	ID of the accounting record: “HSMS”
2	04	4	8	B2	Timestamp of the TOD system clock
3	0C	12	2	B	Length of the identification section
4	0E	14	2	B	Length of the basic information
5	10	16	4	B	– reserved –

(B) User identification (length: 28 bytes)

Field no.	Distance		Length	Format	Meaning
	hex	dec	(byte)		
1	14	20	8	A	User ID
2	1C	28	8	A	Account number
3	24	36	4	A	Task sequence number (TSN)
4	28	40	8	A	Group name

(C) Basic information (length: 40 bytes)

Field no.	Distance		Length	Format	Meaning
	hex	dec	(byte)		
1	30	48	4+4	B2	Task CPU time (TU and TPR) [1)]
2	38	56	4	B	Number of inputs/outputs [2)]
3	3C	60	17	A	Timestamp of the request [3)]
4	4D	77	4	A	ID of the task type [4)]
5	51	81	4	A	TSN of the executing task
6	55	85	1	A	Record index [5)]
7	56	86	2	B	– reserved –

Note

1) The task CPU time includes the CPU time used by the user programs in TU status and the CPU time used in TPR status for processing SVC calls, program errors and commands. The first word contains the full seconds and the second word contains the remaining fraction in nanoseconds.

2) Here this means inputs and outputs to local, peripheral devices in as far as they are supported by the DMS access methods (SAM, ISAM, UPAM, BTAM, PPAM) or by the SYSFILE management. The terminal and paging inputs and outputs are not counted.

3) The request timestamp has the format “yy-mm-dd hh-mm-ss”

4) ID of the task type:

USER	for a user task
SERV	for a server task
ASUB	for an ARCHIVE subtask
COMM	for a communication task

5) Record index:
“A” for the first record of a record pair.
“B” for the second record of a record pair.
A record with index “B” cannot occur without a corresponding record with index “A”.
A record with index “A” can, however, occur without the corresponding record with index “B”. This record must then be ignored.

(D) Variable information (length: 8 bytes)

The variable information of the HSMS accounting record contains a record extension.

Field no.	Distance		Length	Format	Meaning
	hex	dec	(byte)		
1	58	88	2	B	Number of the extension [1]
2	5A	90	2	B	Distance to the first extension
3	5C	92	2	B	Distance to the second extension
4	5E	94	2	B	Distance to the third extension

[1] The number of extensions is always 3. If there is no third extension, the distance to the third extension is shown as 0.

1st extension (maximum length: 12 bytes)

Field no.	Distance		Length (byte)	Format	Meaning
	hex	dec			
1	00	00	2	A	Extension ID “ID”
2	02	02	1	B	X’00’
3	03	03	1	B	Length of accounting ID
4	04	04	L [1]	C/X	Accounting ID

[1] The accounting ID may be a maximum of 8 bytes in length. It corresponds to the user ID which is a maximum of 8 bytes in length as entered by the user with the AREC macro (ID operand). If no user ID was specified for request processing, this field is filled with X’ FFFFFFFFFFFFFFFF’.

2nd extension (length: 24 bytes)

Field no.	Distance		Length (byte)	Format	Meaning
	hex	dec			
1	00	00	2	A	Extension ID “IO”
2	02	02	1	B	Number of elements (X'01')
3	03	03	1	B	Length of element (X'14')
4	04	04	4	B	Number of I/Os per device for pubsets
5	08	8	4	B	Number of I/Os per device for shared private disks
6	EC	12	4	B	Number of I/Os per device for exclusive private disks
7	10	16	4	B	Number of I/Os per device for magnetic tape cartridges
8	14	20	4	B	Number of I/Os per device for Unit Record Devices

3rd extension

Field no.	Distance		Length	Format	Meaning
	hex	dec	(byte)		
1	00	00	2	A	Extension ID “CO”
2	02	02	1	B	Number of elements (X'xx')
3	03	03	1	B	Length of element (X'20')
4	04	04	8	A	User ID
5	0C	12	8	A	Account number
6	14	20	4	A	Task sequence number (TSN)
7	18	24	4	A	Length of accounting numbers for collector requests
8	1C	28	8	A	Accounting numbers for collector requests
4 + x * 32 Bytes (where x is the number of elements)					

In the 3rd extension the number of elements is variable (X’xx’).

The maximum length of an accounting record with the extension “IO”, which is always present, is 132 bytes.

The minimum length of an accounting record with the extension “CO” (only present for collector requests) is 132 bytes + 36 bytes (complete extension: ID, length, information on the 1st collector user) + 32 bytes (only information on the next collector user).

As an accounting record can be a maximum of 496 bytes long, it is only possible to output information on up to 11 collector users. If more collector users are included in this request they are ignored in the accounting record.

10.2 Occupied storage space (storage levels S1 and S2)

The storage space inventory accounting record (record identifier DSPC; see also the “Accounting records” manual [28]) contains the following for a pubset for each user ID:

- number of occupied PAM blocks on S0
- number of PAM blocks on S1 occupied by migration
- number of PAM blocks on S2 occupied migration

11 Appendix

The sections "Introduction of HSMS in the computer center", "Effect on user applications" and "Notes on the generation of automatic procedures" below are exempt from the restriction on reproduction. System and HSMS administrators may copy them and distribute them to the computer center users concerned prior to the introduction of HSMS. They essentially contain excerpts from the present manual.

In addition, the Appendix contains information on troubleshooting (section "Diagnostic tools and trace functions").

11.1 Introduction of HSMS in the computer center

HSMS (**H**ierarchical **S**torage **M**anagement **S**ystem) is a software product which serves to

- organize data backup and archival
- optimize the utilization of external high-speed storages in a BS2000 system
- transfer data
- version backup

HSMS distinguishes between the three storage levels S0, S1 and S2, which are also addressed as such in HSMS statements and BS2000 commands.

Overview of HSMS storage levels:

storage level	HSMS utilization	Volume	Access	Access time
S0 processing level	Archive management	Disks	Online	Very short
S1 background level	Archival, backup, version backup and migration	Disks	Online	Short
S2 archive level	Archival, backup, version backup and migration	Magnetic tape cartridges	Online	Long

HSMS will be introduced on your system in the near future.

The following pages discuss in detail the repercussions of utilization optimization by means of the “migration” function, since this will also affect non-HSMS users.

Further information on the data backup, archival, and data transfer functionalities can be found in “HSMS Volume 1: Functions, Management and Installation”.

11.1.1 Motivation for use of the migration function

HSMS helps optimize usage of high-speed disk storage units by offering the migration function. Particularly in the case of frequently changing applications it makes sense to migrate currently inactive (not required) data from the processing level to a background level; such data can still be addressed via the corresponding catalog entry. Saturation states on public volumes or pubspace limits being reached by user IDs can thus be avoided without compromising file availability.

Whenever the Data Management System (DMS) attempts to access a migrated file, an HSMS request is generated automatically; the file is then recalled to the processing level without user intervention. Migration permits more efficient utilization of expensive high-speed external storages without increasing personnel requirements. Since only those pages are migrated which are actually occupied, storage space is saved at the background level in comparison with the processing level.

11.1.2 Migrated files

Following migration, a file is marked as “migrated”, i.e. it has a catalog entry on the processing level but occupies no space there. The catalog entry shows to which storage level (S1 or S2) the data of the file in question has been migrated.

Depending on the background level to which the file has been migrated, PUB/S1 or PUB/S2 is output for SUPPORT. Migration does not affect the file size and last-page pointer entries.

An overview of the migrated files can be obtained by issuing the command:

```
/SHOW-FILE-ATTRIBUTES FILE-NAME=*ALL, -
/  SELECT=*BY-ATTR(STORAGE-LEVEL=(*S1,*S2))
```

Sample output for migrated files

```
/SHOW-FILE-ATTRIBUTES FILE-NAME=TEST.HSMS,INFORMATION=*ALL-ATTRIBUTES
%00000543#:10SN:$MANUALE.TEST.HSMS
%  ------------------------------ HISTORY      ------------------------------
%  CRE-DATE   = 2019-01-29  ACC-DATE   = 2019-02-05  CHANG-DATE = 2019-01-29
%  CRE-TIME   =   15:04:05  ACC-TIME   =   15:58:22  CHANG-TIME =   15:04:20
%  ACC-COUNT  = 2           S-ALLO-NUM = 61
%  ------------------------------ SECURITY     ------------------------------
%  READ-PASS  = NONE        WRITE-PASS = NONE        EXEC-PASS  = NONE
%  USER-ACC   = ALL-USERS   ACCESS     = WRITE       ACL        = NO
%  AUDIT      = NONE        FREE-DEL-D = *NONE       EXPIR-DATE = 2019-01-29
%  DESTROY    = NO          FREE-DEL-T = *NONE       EXPIR-TIME =   00:00:00
%  SP-REL-LOCK= NO          ENCRYPTION = *NONE
%  ------------------------------ BACKUP       ------------------------------
%  BACK-CLASS = A           SAVED-PAG  = COMPL-FILE  VERSION    = 1
%  MIGRATE    = ALLOWED     STOR-LEVEL = S2
%  #BACK-VERS = 0
%  ------------------------------ ORGANIZATION ------------------------------
%  FILE-STRUC = ISAM        BUF-LEN    = STD(1)      BLK-CONTR  = PAMKEY
%  IO(USAGE)  = READ-WRITE  IO(PERF)   = STD         DISK-WRITE = IMMEDIATE
%  REC-FORM   = (V,N)       REC-SIZE   = 0
%  AVAIL      = *STD
%  WORK-FILE  = *NO         F-PREFORM  = *K         S0-MIGR    = *ALLOWED
%  ------------------------------ ALLOCATION   ------------------------------
%  SUPPORT    = PUB/S2      S-ALLOC    = 16          HIGH-US-PA = 543
%  EXTENTS     VOLUME     DEVICE-TYPE
%  NONE        NONE         NONE
%:10SN: PUB/S2:      1 FILE  RES=      543  FRE=        0 REL=        0 PAGES
```

Migrated files are identified by a number sign (#) between the file size and catalog ID entries. As long as a file is migrated, the following applies to accessing this file and its catalog entry:

- Changes to file attributes such as protection attributes, passwords, etc. are possible and remain in effect after the file is recalled.
- However, the file cannot be renamed. Attempts to do so will be rejected with an error message.
- You can change the file size (SPACE operand in MODIFY-FILE-ATTRIBUTES). They do not take effect or become visible until the file is recalled.

- The file can be deleted at any time. A migrated file whose catalog entry has been deleted is invalid for HSMS. After all, it cannot be recalled without a catalog entry.
- Overwriting the file (OPEN OUTPUT, OUTIN or UPDATE) is permitted at the processing level. The file is then no longer considered as migrated and also becomes invalid for the migration archive.

11.1.3 Recalling migrated files

The opposite of migration, i.e. copying migrated data back to the processing level, is known as “recall”.

The number of files that can be recalled to S0 is restricted by the space allocated to the individual user IDs.

Only files cataloged as migrated files and only that status in the migration archive which matches the catalog entry (CFID and version) can be recalled. The current state of a file on the processing level is not changed by a recall.

Nonprivileged users can recall the files under their own user ID. Users can only recall files of another user ID they are co-owners of the file or the file access rights permit access for other user IDs. In particular when the recall concerns a migrated file of another user ID it should be borne in mind that this also increases the storage space occupation of this user ID.

Warning

> The following file attributes are modified by a recall: CFID (coded file ID), DEVICE-TYPE, SUPPORT, VOLUME and EXTENTS. This must be taken into consideration in applications which evaluate these attributes.

HSMS offers two different options for recalling a migrated file:

- recall by HSMS statement (explicit recall)
- recall on access (implicit recall)

11.1.3.1 Explicit recall by HSMS statement

Any user can recall the data of one or more migrated files to the processing level (S0) by calling the HSMS program and issuing the HSMS statement RECALL-MIGRATED-FILES. For a detailed description of this statement and its operands, see “HSMS Vol. 2” [1].

11.1.3.2 Implicit recall by file access

Users can recall migrated files without explicitly invoking HSMS.

As soon as an access attempt is made, the Data Management System (DMS) of BS2000 determines on the basis of the catalog entry that the data belonging to a migrated file is not available on the processing level.

If a user or a user program attempts read access to a migrated file (OPEN modes INPUT, INOUT, but also UPDATE, REVERSE and EXTEND), the data of that file is automatically (implicitly) recalled to the processing level.

If a user wants only write access to the file (OPEN mode OUTPUT or OUTIN), all data of the file is to be overwritten, i.e. no previous recall is necessary.

In the case of an implicit recall, the data of the file is fetched from a save file on the storage level entered in the catalog.

The procedure for file access via SECURE-RESOURCE-ALLOCATION is somewhat different. This BS2000 command enables you to announce the upcoming access to one or more files. As also in conjunction with devices and exclusively occupied volumes (e.g. magnetic tape cartridges), in this case the command is also used to make all the required resources available (in this case files and their data), i.e. if necessary to initiate a recall.

11.2 Effect on user applications

- Recall on file access
- Recall via SECURE-RESOURCE-ALLOCATION
- Protecting files against migration

11.2.1 Recall on file access

Recalling migrated files has the following repercussions:

- The recall is initiated implicitly when the file is opened. This is possible for any file.
- Recalling takes a certain amount of time, depending on the current system and device loads and on the type of background storage used (disks in the case of S1, magnetic tape cartridges S2).
- This means that processing in a user application can be interrupted (i.e. delayed) at a number of points (whenever a file is opened).
- The background storage is accessed every time a recall request is issued. In particular for migration to S2, implicit recalls can considerably increase the mount rate, which usually leads to a corresponding increase in runtime.

11.2.2 Recall via SECURE-RESOURCE-ALLOCATION

If the recall procedure is controlled via SECURE-RESOURCE-ALLOCATION, the following effects are to be expected:

- Implicit recall is initiated only if the user is ready to wait (operand WAIT=(TIME=wait-time)). Default for interactive tasks: no wait time. Such a wait time should be sufficiently long so as to enable tapes to be accessed (possible value: 2 minutes).
- All the files required for a job (up to 48) can be requested with one command and recalled via one request. This reduces the number of HSMS calls and possibly also the number of tape requests.
- There is a defined waiting point after which processing either can start or must be aborted. “Unpleasant surprises” caused by missing files during processing can thus be avoided.
- HSMS outputs a summary report for recall requests initiated in this way. Such a report for implicit recalls is otherwise printed in the event of an error only.

Tape access can be coordinated by the system administrator by means of defined tape processing times. Please contact your system administrator if restrictions of this kind are reported after you tried to access a migrated file.

11.2.3 Protecting files against migration

A file can be exempted from migration in the following ways:

- File attribute “MIGRATE”: the migration inhibit is set via MIGRATE=*INHIBITED of CREATE-FILE or MODIFY-FILE-ATTRIBUTES commands (canceled with MIGRATE=*ALLOWED).
  This protection can be disabled by the system administrator, i.e. the file is only protected against “standard” migration. This attribute should therefore be used sparingly so that migration of non-protected files can relieve the load on disk storage and the system administrator is not forced to migrate protected files.

  User IDs that have the right to make physical allocations also have the MIGRATE=*FORBIDDEN option. This inhibits migration completely.
- The system administrator can keep a list of important files that are to be protected. This list may also include user files. In addition to setting the file attribute MIGRATE=*FORBIDDEN, only the entry in this list protects the file concerned from migration.
- All files that are open when the migration request is processed are likewise exempted from migration. However, this can normally be avoided by processing the migration request outside the standard session times.

11.3 Notes on the generation of automatic procedures

- Recommended changes
- Mandatory checks
- Uncritical handling of migrated files

11.3.1 Recommended changes

The aspects described below merely serve for application runtime optimization and may otherwise be ignored. The relevant procedure is recommended, however, to reduce the overall system load.

- Checking/ensuring the existence of a file should be effected using the SECURE-RESOURCE-ALLOCATION command rather than the commands SHOW-FILE-ATTRIBUTES or ADD-FILE-LINK.
- Before processing is started, a check should be made for all files to be subsequently opened for reading to ensure that their data is actually present on the processing level; this is done with the aid of a SECURE-RESOURCE-ALLOCATION command *with* specification of a wait time. A positive side effect of this is that several files are recalled by one request, which can relieve the load on the system (see above).
- If deletion of a file is to cause its data to be overwritten (DESTROY operand), this should be defined as an attribute of the file (MODIFY-FILE-ATTRIBUTES command) and not only when the file is actually deleted (DELETE-FILE command). Otherwise deletion of the data within the framework of migration will *not* overwrite the data.
- If the retention period for a file migrated to S2 is changed, availability of the corresponding data during that interval cannot be guaranteed. A previous recall is therefore recommended.

11.3.2 Mandatory checks

The aspects described below must be observed in order to ensure safe execution of the applications.

- Unless it is ascertained that the file has been recalled by a preceding /SECURE-RESOURCE-ALLOCATION command, the contents of the catalog entry for a file *before* opening the file must *not* be compared with the contents *after* opening it. This applies to the attributes:
  - coded file ID (CFID)
  - locality description (VOLUME, extent count and extent list)
- Applications using the internal file name (CFID) for checking and processing files (e.g. with UPAM) must not determine the CFID until any recall operation has been completed.
- Migrated files may not be renamed. Corresponding actions are rejected with error messages.

  If an application wants to rename files that have not previously been processed, a SECURE-RESOURCE-ALLOCATION command must be issued, for example, to ensure that the file has not been migrated.

11.3.3 Uncritical handling of migrated files

- Files can be deleted regardless of the level on which they are stored (S0, S1 or S2). For migrated files, this merely means that their catalog entry is deleted.
- File attributes (protection attributes, passwords etc.) can be changed. Such changes take effect immediately and are retained even after recalling the data.
- If the size of a migrated file is changed (SPACE operand of the MODIFY-FILE-ATTRIBUTES command), only the catalog entry is modified initially. This does not influence the storage space situation until the data has been recalled to the processing level.

11.4 Diagnostic tools and trace functions

The tools discussed below are used to diagnose specific problems in customer operation.

Selection and use of a tool should always be agreed on with the field engineers responsible.

11.4.1 HSMS trace

Use of this trace makes sense if execution problems are suspected with requests in HSMS.

The HSMS trace is activated and deactivated with the privileged HSMS statements START-TRACE and STOP-TRACE.

START-TRACE	*starts the HSMS trace*
DOMAIN = **<u>*ALL</u>** / list-poss(32): ***REPOSITORY** / ***MESSAGES** / ***SHARE-PVS** / ***STATEMENTS** ,**TSN** = **<u>*ALL</u>** / <alphanum-name 1..4>	

STOP-TRACE	*terminates the HSMS trace*
CLOSE-MODE = **<u>*NORMAL-CLOSE</u>** / ***PSEUDO-CLOSE**	

`The DOMAIN operand determines whether the trace should cover all or some specific functional areas:`

- The operand value *ALL initiates trace entries for all functional areas which are subsequently named.
- The operand value *REPOSITORY currently generates no trace entries.
- The operand value *MESSAGES generates trace entries for all HSMS message outputs (parameter list of MSG7) via the DHSBMSG module.
- The operand value *SHARE-PVS generates trace entries for the HSMS requests sent in shared pubset operation (internal HSMS commands) via the DHSBCIH module.
- The operand value *STATEMENTS generates trace entries for all HSMS statements entered via the DHSEVAL module.

When the trace is deactivated the diagnostic information is written to a file. The CLOSE-MODE=*PSEUDO-CLOSE operand allows the trace to continue, but the data written up to this point can be read.

Name of the trace file: $SYSHSMS.HSM.Y.TRACE.<yyyymmdd>.<hhmmss>

The status of the trace is output using the HSMS statement SHOW-LOGGING-STATUS.

The content of the trace file can be edited and written to SYSLST using the HSMS statement LIST-LOGGING-FILE.

11.4.2 ARCHIVE trace

Use of this trace makes sense if execution problems are suspected with requests in ARCHIVE.

The ARCHIVE trace is activated and deactivated with the privileged commands START-ARCHIVE-TRACE and STOP-ARCHIVE-TRACE. This trace can only be started for specific functional areas and tasks.

START-ARCHIVE-TRACE	*starts the ARCHIVE- trace*
TSN = ***ALL** / <alphanum-name 1..4> **,DOMAIN** = ***ALL** / list-poss(32): ***DEVICE** / ***MULTIPLEXING** / ***PACKET** / ***SYNCRONIZATION** / ***PLAM** / ***MEMORY** / ***DIRECTORY**	

STOP-ARCHIVE-TRACE	*terminates the ARCHIVE trace*
CLOSE-MODE = ***NORMAL-CLOSE** / ***PSEUDO-CLOSE**	

MODIFY-ARCHIVE-TRACE	*modifies the active ARCHIVE trace*
TSN = ***ALL** / <alphanum-name 1..4> **,DOMAIN** = ***ALL** / list-poss(32): ***DEVICE** / ***MULTIPLEXING** / ***PACKET** / ***SYNCRONIZATION** / ***PLAM** / ***MEMORY** / ***DIRECTORY**	

`The DOMAIN operand determines whether the trace should cover all or only (several) specific functional areas:`

- The operand value *ALL initiates trace entries for all functional areas which are subsequently named.
- The operand value *DEVICE initiates trace entries for all volume and tape device requests to MAREN and NDM in the modules FAPO, FASC, FASR and FAUTDM.
- The operand value *MULTIPLEXING initiates trace entries for all requests between tape driver and multiplexing subtasks.
- The operand value *PACKET initiates trace entries for packet processing in the modules FAFN, FAFNSV and FARPX.
- The operand value *SYNCHRONISATION initiates trace entries for synchronization between ARCHIVE main task and subtasks in the FASC module.
- The operand value *PLAM initiates trace entries for writing and reading the PLAM info in the FAFNPL module.
- The operand value *MEMORY initiates trace entries for storage requests/releases in the FAUTMR module.
- The operand value *DIRECTORY initiates trace entries for accesses to directories in the modules FARAM and FAUTRM.

When the ARCHIVE trace is started or stopped the associated trace ID is output.

When the trace is deactivated the diagnostic information is written to a file and the associated file name is output. The CLOSE-MODE=*PSEUDO-CLOSE operand allows the trace to continue, but the data written up to this point can be read.

11.4.3 Performance statistics

Normally the ARC0815/ARC0825 and ARC0816/ARC0826 messages supply information on the performance of a backup or restore. The size and number of files and job variables transferred are output, plus the time required. When recording the time the wait time while the tape is mounted is not taken into account. The timer only runs while the save file ARCHIVE.SAVE.FILE is open.

A message is output for each ARCHIVE subtask. Examples:

```
ARC0815 SUBTASK '0' HAS TRANSFERRED '10' PAM PAGES FOR '1' FILES AND '3' JVS IN '4' SECONDS
```

```
ARC0816 SUBTASK '1' HAS TRANSFERRED '27' MEGABYTES FOR '1305' FILES IN '324' SECONDS
```

If the save times are unexpectedly long or problems are suspected in individual files or tape transition, additional performance statistics can help with problem diagnosis.

These performance statistics are requested with the PERFORMANCE-ANALYSIS=*YES operand of the statement for backup or restore.

The performance statistics are entered in a file for each subtask under the user ID SYSHSMS under the following name:

`HSM.S.SYSHSMS<user-id><year><julian date><hour><min><secs>.STATS.<subtask-nr>`

The component <user-id> in the name is only included for nonprivileged users.
Nonprivileged users can generate this file under the user ID SYSHSMS, but no access rights are set for them.

A new information line is written into the statistics file in the following cases:

- when opening a save file ARCHIVE.SAVE.FILE
- when closing the save file
- at the start of backing up or restoring a new file
- every 2 seconds when backup or restore takes place

The performance statistics thus provide a detailed overview of all the activities of a subtask. Among other things, the time required to back up or restore each file can be ascertained. To facilitate analysis of the statistical data by program, the individual fields are separated from each other by a separator character which can be freely defined when the statistical data is requested (the default is a semicolon). The structure of an information line is described in the table below.

Field	Position	Length	Value	Meaning
Time stamp	1	19		YYYY-MM-DD HH:MM:SS
Separator 1	20	1	;	Separator
Number of files	21	11		Number of files/paths processed so far
Separator 2	32	1	;	Separator
Number of job variables	33	11		Number of job variables processed so far (always 0 in a UFS file isystem or Windows)
Separator 3	44	1	;	Separator
Number of PAM pages or MBytes transferred internally	45	20		PAM pages with DMS files, MBytes with UFS or Windows; transfer from/to save file (internal transfer)
Separator 4	65	1	;	Separator
Number of xBytes transferred externally; x= K / M / G	66	11		KBytes/MBytes/GBytes which have been transferred over the network with TCP/IP (external transfer)
Separator 5	77	1	;	Separator
Unit for xBytes	78	1		’1’ = K; ’2’ = M; ’3’ = G
Separator 6	79	1	;	Separator
Name of the workstation	80	48		Name of the current workstation; empty if DMS file or *BS2-UFS
Separator 7	128	1	;	Separator

Although the output of the performance statistics only increases the load slightly, it should not be used constantly in normal operation.

The performance statistics can be requested with the following statements (see the “HSMS Statements” manual [1]):

Statement	Function
ARCHIVE-FILES	Archive files and job variables
ARCHIVE-NODE-FILES	Archive node files
BACKUP-FILES	Save files and job variables
BACKUP-NODE-FILES	Save node files
BACKUP-FILE-VERSIONS	Save versions of files
COPY-EXPORT-SAVE-FILE	Copy save file
COPY-NODE-SAVE-FILE	Copy node save file
COPY-SAVE-FILE	Copy save file
EXPORT-FILES	Export files and job variables
IMPORT-FILES	Import files and job variables
MIGRATE-FILES	Migrate files
MOVE-SAVE-FILES	Moving save or node save files
RECALL-MIGRATED-FILES	Recalling migrated files
REORGANIZE-VERSION-BACKUP	Reorganization of version backup archive
REPAIR-CATALOG-BY-EXCHANGE	Removing inconsistencies in migrated files using EXCHANGE
REPAIR-CATALOG-BY-RESTORE	Repair inconsistencies in migrated files using RESTORE
REPLACE-SAVE-FILE-BY-EXCHANGE	Replacing save file in the migration archive using EXCHANGE
REPLACE-SAVE-FILE-BY-RESTORE	Replace save file in migration archive using RESTORE
RESTORE-FILES	Restore files and job variables
RESTORE-LIBRARY-ELEMENTS	Restore elements of a library file
RESTORE-NODE-FILES	Restore node files
UPDATE-EXPORT-SAVE-FILE	Update backup

Table 8: HSMS statements supporting the use of perfomance statistics

11.4.4 Further information on diagnostics

If faults during tape operation mean that problem analysis that goes beyond ARCHIVE is necessary, an NDM trace should be created after the field engineers responsible have been consulted. Creation of the NDM trace is described in the “Diagnostics Handbook” [27].

12 Glossary

action statement

Statement issued to HSMS which, in order to be processed, requires I/Os to and from user files. The following are action statements:
ARCHIVE-FILES, ARCHIVE-NODE-FILES , BACKUP-FILES, BACKUP-FILE-VERSIONS, BACKUP-NODE-FILES, COPY-EXPORT-SAVE-FILE, COPY-NODE-SAVE-FILE, COPY-SAVE-FILE, EXPORT-FILES, IMPORT-FILES, LIST-VOLUMES, MIGRATE-FILES, MOVE-SAVE-FILES, RECALL-MIGRATED-FILES, REORGANIZE-VERSION-BACKUP, REPAIR-CATALOG-BY-RESTORE, REPLACE-SAVE-FILE-BY-RESTORE, RESTORE-FILES, RESTORE-NODE-FILES, SELECT-FILE-NAMES, SELECT-JV-NAMES, SELECT-NODE-FILES, UPDATE-EXPORT-SAVE-FILE

additional mirror unit

-> BCV data volume or -> clone unit

archival

Long-term saving of selected files, e.g. for product liability or revision security reasons. The files are deleted from the processing level once they have been saved.

ARCHIVE

BS2000 software product which saves files and job variables logically.
ARCHIVE has an internal interface with HSMS and implements the -> HSMS action statements.

archive

Management unit for files under HSMS management, consisting of the archive definition and the associated archive directory (directory file). Archives are addressed via their -> owner ID and the archive name.
HSMS makes a distinction between various archive types:
For DMS files: archives for -> data backup (backup archives), -> files version backup (version backup archives), -> archival (long-term archives) -> migration (migration archives) and -> shadow archives.
For node files: archives for node backup (node backup archives) and node archival (node long-term archives).
An archive can be private, i.e. only available to the -> archive owner, or public, i.e. available to all users.

archive directory

File used for managing the objects saved in an archive, i.e. files, job variables,-> save files, -> save versions and the -> volume pool, and implemented as an ARCHIVE directory file (-> ARCHIVE).

archive owner

Users have the right to create archives by means of the CREATE-ARCHIVE statement. The creator of an archive is the archive owner. Only the HSMS administrator may enter any user ID in the archive name, using the CREATE-ARCHIVE statement. In doing this, the corresponding user becomes the archive owner.

archive type

Determines the basic HSMS function for which an archive is to be used.

backup

The regular creation of copies of the data inventory for the restoration of data lost due to hardware or software errors or inadvertent deletion, etc. This facility can also be used to reorganize disk storage.

backup archive

HSMS archive used for backup.

backup class

Save level assigned to a file and entered in the catalog; this level is used to determine the frequency with which the file is backed up. Possible values are *A, *B, *C, *D and *E. Files of backup class *A are backed up with each backup run. Files with backup class *E are saved only if MAXIMUM-BACKUP-CLASS=*E was specified explicitly.

backup monitor

-> BS2000 Backup Monitor

backup server

Backup server for workstations which are connected to the local BS2000-UFS (POSIX) via NFS.

BCV data volume

Additional -> mirror disk in a -> disk storage system system which can be split off for other purposes (backups, test operations etc.) without impairing current operation. A BCV data volume is also referred to as an Additional Mirror Unit.

BS2000 Backup Monitor

The BS2000 backup monitor is an SE management application which is integrated into the **Applications** main menu of the SE Manager. The BS2000 backup monitor provides information on the status of the backup requests which were assigned in the SE server's BS2000 systems with the software products HSMS and FDDRL.

BS2000 file

Specifies a file which is created and processed exclusively by BS2000. BS2000 files have been processed on Net-Storage (FILE-TYPE=BS2000) since BS2000/OSD-BC V9.0. They reside directly on a Net-Storage volume. Open systems may only access them in read mode.

BS2000 net storage file

-> net storage file

BS2000-UFS

UNIX file system residing on a BS2000 server.

Cataloged-Not-Saved (CNS)

Indicates that a file was not saved either because an incremental backup was performed and the file had not been changed since the previous backup, or because an error (e.g. open error) prevented it from being saved.

CFID

Coded File ID -> internal file name

client

Computer that accesses the services of another computer (-> server) via the network. Any one computer can simultaneously function as a client requesting services from other computers and as a server offering services to other computers.
see also -> HSMS client

client/server architecture

System architecture in which computing capacities and applications are distributed over -> clients and -> servers.
Client functions are typically performed by PCs, workstations and -> UNIX systems.
Server functions are primarily performed by mainframes and UNIX systems subject to certain constraints.
Client and server systems can be combined as required; in principle, any client has access to any server.

clone unit

A clone unit is the copy of an (original) unit at a particular time (Point-in-Time Copy) which, depending on the disk storage system, is created by the component EquivalentCopy, TimeFinder /Clone or SnapViewClone. Following activation, the unit and the clone unit are separated from each other, and applications can access both.

CNS

-> Cataloged-Not-Saved

coded file ID

-> internal file name

collector request

Request for migration to S2 or requests for archival in the -> standard save file of a system archive can be combined in a collector request unless they were issued by the HSMS administrator. The requests are then executed together (thus reducing the number of tape access and positioning operations) but HSMS still outputs separate request-specific reports. An aborted collector request must be restarted by the HSMS administrator.

Concurrent Copy

Option of the HSMS statements BACKUP-FILES and BACKUP-FILE-VERSIONS which permits BS2000 files to be modified during a backup.

computer network

Organization of several computers interconnected via a physical line with the purpose of permitting peer-to-peer data exchange between these computers. A distinction is made between local computer networks (-> LAN) and non-local computer networks (-> WAN).

continuation period

Period of time during which the standard save file of an archive is continued. The continuation period may have any length between a day and a month.

control file

File under user ID SYSHSMS that contains the HSMS control parameters and the archive definitions. On system managed storage, the control file is located under the SYSHSMS user ID of an SM pubset; it contains the control parameters for this SM pubset.

co-ownership

Nonprivileged users can also work with files and job variables of which they are co-owners.

data

Within the context of HSMS, files, job variables and catalog entries of files (on magnetic tape cartridge or private disk).

data transfer

The transfer of files, job variables or catalog entries from files to other BS2000 systems or other user IDs, implemented by exporting data to magnetic tape cartridge, public disk or -> Net-Storage and then importing it to the destination.

default system archive

-> Archive assigned globally to the entire system or to specific SF/SM pubsets and accessed unless another archive is specifically specified. There is a separate default system archive for each of the basic HSMS functions -> migration (-> SYSMIGRATE), -> version backup (-> SYSVERSION (only specific to SF/SM pubsets)), -> backup (-> SYSBACKUP), -> archival (-> SYSARCHIVE), node backup (-> SYSNODEBACKUP) and node archival (-> SYSNODEARCHIVE).

directory

Part of a UNIX file system that serves to group and organize files and directories.

directory file

A term used synonymously for -> archive directory; for more detailed information about the structure of a directory file, see “Directory file“, in the “ARCHIVE“ manual [2].

disk storage system

External disk system. Mirroring functions (e.g. backing up Snapsets) can be used in BS2000/OSD for disk storage systems which the BS2000 host component SHC-OSD manages.

ETERNUS disk system

-> disk storage system of the company Fujitsu Technology Solutions.

Ethernet

Standardized method of computer interconnection used to build -> local area networks (LAN).

except file

File maintained by the system administrator, containing the (fully or partially qualified) names of the files which are to be exempted from migration. The names are stored in the except file.

export

Writing date to magnetic tape cartridge, public disk or -> Net-Storage for the purpose of -> data transfer.

express request

Category of requests restricted to the HSMS administrator for which special -> tape sessions can be defined.

file expiration date

-> retention period for the save version in a long-term archive.

file system

Hierarchical collection of -> directories and files forming a tree structure. The origin of this tree structure is referred to as the -> root directory (/). All other directories are branches of this tree. Each file of a file system is accessible via exactly one path of this file system.

HSMS

Hierarchical Storage Management System: BS2000 software product offering such functions as -> migration, -> backup, -> archival, and -> data transfer, implemented in a -> storage hierarchy and in -> archives.

HSMS administrator

User enjoying the HSMS administrator privilege. This privilege is granted to any user working under the user ID SYSHSMS and under TSOS.
The HSMS administrator can use all HSMS functions without restrictions. The following are typical HSMS administrator tasks: managing the storage hierarchy, creating the default system archives, system backup and control of tape processing.

HSMS client

-> passive HSMS client

HSMS functions

-> HSMS

HSMS run

Period of time between starting the HSMS program and terminating it with END.

HSMS session

Period of time between loading subsystem HSMS (with /START-SUBSYSTEM or //START-HSMS) and unloading it (with /STOP-SUBSYSTEM or //STOP-HSMS); see also -> SM pubset session.

implicit recall

Automatic recall of migrated files as a result of attempts, on the part of DMS, to access these files, as opposed to -> recall requested via an HSMS statement.

import

The loading of exported data to the destination during -> data transfer.

inactive date

Date on which a file was last accessed.

inactive files

Files with a high -> inactive date, i.e. files which have not been referenced for a long time, are obvious candidates for -> migration.

internal file name

Internal name (CFID) listed in the catalog along with the file name; this name uniquely identifies the file and is modified each time the file is altered. It is transferred in the FSTAT macro but not output with SHOW-FILE-ATTRIBUTES.

invalid file

-> Migrated file which is still contained in the -> save file of the -> migration archive, but which was already recalled, overwritten or deleted on processing level S0.

LAN (Local Area Network)

Hardware configuration of a local -> network consisting of data display terminals and other devices installed in proximity to one another, e.g. in the same building.
The proximity permits the use of simple transmission technology and thus transmission at higher speed and lower cost.
Depending on national legislation, the boundaries of the user's premises may constitute the boundaries of the LAN. A LAN may be connected to other -> computer networks as a private subnetwork and thus become part of a larger network such as a WAN.
Synonyms: local computer network, local network.

level

-> storage hierarchy

locally available

-> pubset

local computer

The computer at which the user is working. All other computers in a -> computer network are -> remote computers for that user. Once a user has connected up with a remote computer, that computer becomes that user's local computer.

long-term archive

HSMS archive used for archival.

master computer

Computer on which all the DMS management functions of a -> shared pubset are executed.

MDS

-> Modified-during-save

migrated file

A migrated file is a file whose data has been deleted from the processing level but whose catalog entry remains on this level. The catalog entry indicates the background level to which the data was migrated.

migration

Moving -> inactive files from the processing level to a background level without deleting the catalog entry.

migration archive

HSMS archive used for migration.

mirror disk

Disk set consisting of at least two disks having identical contents.

Modified-during-save

Indicates that a -> node file was opened by another user while being saved.

monitoring

If HSMS is linked to the application SM2, SM2 users can better monitor migration or recall activities. The resulting information can result in the migration parameters having to be reset.

MPVS

Multiple public volume set: public volumes distributed over several -> pubsets.
It is possible to assign specific pubsets to specific users.

multiplexing operation

Multiplexing operation is only available for HSMS backup server functions, i.e. for backing up and restoring files at remote workstations. In this operating mode, multiple ARCHIVE subtasks can simultaneously share the same MTC devices and magnetic tape cartridges in parallel. This ensures enhanced performance and the optimum utilization of MTC devices and guarantees that magnetic tape cartridges are optimally filled.

network

A complex structure of lines and control units that provides data communication services.

Net-Storage

The storage space provided by a Net-Server in the computer network and released for use by external servers. Net-Storage can be a file system or also only a node in the Net-Server’s file system.

Net-Storage file

Specifies a file which is created on a Net-Storage volume. On Net-Storage a distinction is made between two file types: BS2000 file and node file.

NFS (Network File System)

BS2000 software product that permits distributed data storage in a heterogeneous -> computer network. It enables users to access remote files as if they were residing on their -> local computer. NFS is thus used for connecting systems. Furthermore, the automatic and reliable BS2000 data saving functions can be made available for the files of such systems via NFS.

node

Computer (workstation or PC) connected to a computer network.

node backup archive

HSMS archive used to -> back up node files.

node file

Meaning in Net-storage context:
Specifies a Net-Storage file (FILE-TYPE=NODE-FILE) which can be created and processed both by BS2000 and open systems. Node files are supported in BS2000 OSD/BC V10.0 and higher. They reside on a Net-Storage volume in a user-specific directory (name of the user ID), and the file names comply with the BS2000 naming conventions.
Meaning for UNIX workstations:
Node files are files or -> directories in a UNIX file system (UFS) of a UNIX workstation. When implementing -> NFS, node files are understood to be the files or -> directories of a workstation which can be accessed via -> BS2000-UFS. Node files are case-sensitive.
If directories or file systems have been mounted via BS2000-UFS (POSIX), the files contained therein can be saved or archived with HSMS BACKUP-NODE- FILES oder ARCHIVE-NODE-FILES, however, they cannot be processed by BS2000 DMS directly.

node long-term archive

HSMS archive for the -> archival of node files.

node S0

Designates a remote -> file system mounted on the “HSMS“ node of a -> local BS2000-UFS by the system administrator.
The local BS2000-UFS is also considered as a node S0.

number of backup versions

Number in the range of from 0 to 32 which denotes maximum value of file versions guaranteed to be kept within version backup archives. It is a file attribute which is stored in its catalog entry and can be specified within create-file or modified during modify-file-attributes. In SM-environment, it also can be given within management classes. The value is (re)written into directory of a version backup archive during version backups (//BACKUP-FILE-VERSION) and also is updated to the actual during //CHECK-CATALOG-FILES.

owner ID

Upon creation of an archive, the user ID of the -> archive owner is stored as the owner ID in the HSMS control file.
The owner ID SYSHSMS is assumed if the HSMS administrator creates an archive without specifying an owner ID.

passive HSMS client

For UNIX workstations:
The software product -> NFS is required in addition.
A passive HSMS client is mounted using -> NFS in BS2000's POSIX files system. -> Backup /archiving takes place using HSMS with the BACKUP-NODE-FILES or ARCHIVE-NODE-FILES statement. No additional software is required on the node.

path name

Under -> UNIX, each file and each -> directory is identified by a unique path name. The path name designates the position of the file or the directory within the -> file system as well as indicating how the file/directory can be accessed. The path name is formed by the names of a file's or directory's parent directories, starting with the -> root directory, and its own name. The names of the individual directories are separated by a slash.
(Example: c:dir1/dir2/protocol)
UNIX distinguishes between absolute and relative path names.

pool

-> volume pool

protection attributes

Security-relevant attributes of an object (file, job variable, etc.) which determine the type and scope of access to this object.
Examples of file protection attributes are: ACCESS, USER-ACCESS, AUDIT READ-PASSWORD, WRITE-PASSWORD, EXEC-PASSWORD, RETENTION-PERIOD, BASIC-ACL and GUARD.

pubset

Public volume set: set of public disks which belong together and may, for instance, have a common catalog. A pubset is considered to be locally available once its catalog and data have been imported (IMPORT-PUBLIC).
A pubset is identified by a one- to four-character ID.

recall

To move migrated files back from a background level to the processing level -> S0. The recall can be started explicitly using the RECALL-MIGRATED-FILES statement or implicitly by opening the file.

remote computer

A -> network comprises remote and -> local computers. All computers in a network other than the one at which the user is working are remote computers for that user. Users can communicate with all remote computers in the network. Once a user has connected up with a remote computer, that computer becomes that user's local computer.

reorganization

HSMS function used primarily to reorganize the migration archive, i.e. to reshuffle save files without transferring -> invalid files.
When long-term archives are reorganized, only save versions whose-> file expiration date has not yet been reached are copied. Obsolete save versions are not accepted.
However, HSMS can also reorganize -> pubsets and private disks (-> backup).

repository

-> archive directory for node files

request

Generated for the processing of -> action statements and in connection with -> implicit recall. All requests are entered in the -> request file.

request file

File under the SYSHSMS user ID containing the requests. It is from this file that the tasks which process the -> action statements are started.

restore

To move data from an HSMS archive back to the processing level -> S0.

retention period

Period of time during which data modification or deletion is prohibited. The (physical) retention period prevents save files and save volumes from being overwritten during this time, while the (logical) retention period defined by the file expiration date prevents files from being modified or deleted.

root directory

Main directory of a tree-structured -> file system which constitutes the origin of all other -> directories. The root directory must be present in any file system.

S0

Online processing level, implemented by disk storages with short access times.
Its management units are the pubsets.

S1

Online background level, implemented by disk storages (possibly with more capacity and longer access times than -> S0); its management units are the pubsets.

S1-SM-pubset

An SM pubset which is used as S1 storage level of SF pubsets. This ensures that the storage space available for backup and migration at storage level S1 is no longer limited to the size of an SF pubset of 4TB, but can almost be 1000TB in size.

S2

Offline background level implemented by archives on magnetic tape cartridges.

save

Generally speaking the copying of data to a -> save file, irrespective of the basic function used. Also used as a synonym for -> backup.

save, logical

To read data from one or more volumes and write it contiguously to one or more volumes.

save, physical

Block-by-block copying of the entire data of a volume, including the volume labels, to another volume, so that the contents and structure of the second are identical to those of the original volume.

save file

“Receptacle“ for saved files and job variables. The save file contains one or more -> save versions and consists of a set of volumes which all have the same owner and -> retention period. The distinction between save file and save versions is introduced with HSMS, where it is possible to manage several save versions in a single save file in conjunction with archival, migration and node backup. Only entire save files can be released. Each save file is identified by a save file ID (-> SFID) formed by the date and time of its creation.

save version

Result of a backup or archival request. The save version is internally identified by an SVID (save version ID). The user can refer to it via its creation date or the name assigned to it at creation.

server

Computer providing services to other computers (-> clients) in a network.

SFID

Identifies a -> save file; the save file ID has the following format:
S.yymmdd.hhmmss

SF pubset

Abbreviation for “single-feature pubset”.
An SF pubset consists of a set of public disk storages with a shared catalog. It is defined by its catalog ID.
An SF pubset is not the same as an -> SM pubset.

SF pubset environment

Set of HSMS parameters which are specific to all -> SF pubsets. The SF pubset environment includes the parameters of the global control file and the global request file.

shadow archive

Archive that is linked to a -> backup or -> long-term archive. A shadow archive holds the automatically created copies of data which is stored in a backup or long-term archive.

shared pubset

Pubset that can be accessed simultaneously by more than one computer.

sharers

Computers that have simultaneously imported a particular pubset (-> shared pubset).

single-feature pubset

-> SF pubset

SM pubset

Abbreviation for system-managed pubset.
The DMS object SM pubset consists of a set of -> volume sets. An SM pubset is not the same as an -> SF pubset.
The term SM pubset is also used in this manual to denote -> “SM pubset under HSMS control”.

SM pubset not under HSMS control

SM pubset that was not declared for HSMS and can therefore also not be managed by HSMS.

SM pubset under HSMS control

Closed container that holds both the data and the metadata.
In this manual also often simply referred to as -> SM pubset.

SM pubset session

For HSMS, the time from the start of the HSMS subsystem (or the import of the SM pubset if this is done later) to the time the HSMS subsystem is stopped (or the SM pubset is exported if this is done later).

SM pubset environment

Set of HSMS parameters that are specific to an -> SM pubset. The SM pubset environment includes the parameters of the SM pubset control file and the SM pubset request file.

Snapset

Copy of a -> pubset on additional -> mirror disks (Snap disks) which as of BS2000/OSD V7.0 can be created as a pubset backup in a -> disk storage system with SHC-OSD. Files and job variables can be restored from a Snapset using a DMS command or saved with HSMS.

standard save file

Save file to which -> backups for an -> archive are written unless otherwise specified.

storage hierarchy

Assignment of storage units to different storage levels, depending on their availability, access time and storage costs (-> S0,-> S1,-> S2).

storage level

-> storage hierarchy

SYSARCHIVE

-> default system archive for -> archival.

SYSBACKUP

-> default system archive for -> backup.

SYSMIGRATE

-> default system archive for -> migration.

SYSNODEARCHIVE

-> default system archive for -> archival of -> node files.

SYSNODEBACKUP

-> default system archive for -> backup of -> node files.

system archive

-> default system archive

system-managed pubset

-> SM pubset

system startup

Loading of the BS2000 operating system software. The following variants exist: AUTOMATIC-STARTUP, DIALOG-STARTUP and FAST-STARTUP.
The variants are differentiated by their automation. Synonym: system initialization

SYSVERSION

-> default pubset-specific system archive for -> version backup of -> files.

TAPE

Volumes of the class “TAPE“ are assigned to storage level -> S2, these are magnetic tapes with a recording density of 1600 bpi (T1600) and 6250 bpi (T6250) as well as magnetic tape cartridges (TAPE-C2, TAPE-C3, etc.).

tape session

Period of time during which the I/Os to and from -> S2 are processed.

UFS (UNIX File System)

Component of -> NFS providing access to local file systems.

UNIX

Interactive operating system developed by Bell Laboratories in 1969 for use in small computers but meanwhile used in all computer classes. Since only the central UNIX kernel is hardware-dependent, UNIX can be installed on a wide range of systems of different manufacturers.

valid file

-> Migrated file which, up to now, has not been recalled or deleted or overwritten on the processing level. It is marked as a migrated file in the catalog (# before its name).

version backup archive

A new type of archive specially introduced for version backups. In contrast to a backup archive, a version backup archive contains a user-defined number of file versions per file. In the directory file additional information about a files deletion date and if a file should be removed from the archive is stored.

volume pool

Set of volumes managed by an archive and registered in the directory. Volumes required for save requests are normally fetched from the free volume pool of the archive.

volume set

Set of disks that can be addressed by their volume set ID. Several volume sets make up an -> SM pubset.

WAN (Wide Area Network)

-> Computer network not restricted to a limited area. A WAN may include several -> LANs.

13 Abbreviations

ACS	Alias Catalog System
AML	Automatic Media Library
ASCII	American Standard Code for Information Interchange
BACL	Basic Access Control List
BCAM	Basic Communication Access Method
BCV	Business Continuance Volumes
BLS	Binder-Lader-Starter
Catid	Catalog Identifier
CCOPY	Concurrent Copy
CFID	Coded File ID
CHKPT	Checkpoint
CL	Client
CNS	Cataloged Not Saved
DAB	Disk Access Buffer
DCAM	Data Communication Access Method
DEFLUID	Default User ID
DFS	Distributed File System
FGG	File Generation Group
DMS	Data Management System
DSECT	Dummy Section
DSSM	Dynamic Subsystem Management
DMS	Data Management System
EBCDIC	Extended Binary Coded Decimal Interchange Code
EDT	Editor
EOF	End of File
EOV	End of Volume
FARMTSAV	File Archiving Metadata Save

FCB	File Control Block
FDDRL	Fast Disk Dump and Reload
FGG	File Generation Group
FHS	Format Handling System
FIFO	First In First Out
FTP	File Tranfer Protocol
GCF	Generic Catalog Facility
GS	Global Storage (Globalspeicher)
GUARDS	Generally Usable Access Control Administration System
HIPLEX	Highly Integrated System Complex
HSMS	Hierarchical Storage Management System
IFG	Interactive Format Generator
IMON	Installation MONitor
INOP	INOPerable (device status)
ISAM	Indexed Sequential Access Method
JV	Job Variable
K-format	Key format
KB	Kilobyte
LMS	Library Maintenance System
MAREN	Volume archiving system in computer networks (German abbreviation)
MTC	Magnetic Tape Cartridge
MIP	Message Improvement Processing
MPVS	Multiple Public Volume Set
MRSCAT	MasteR CATalog
MSCF	Multiple System Control Facility
MSG	Message
NDM	Nucleus Device Management
NFS	Network File System

NK-Format	NonKey format
OPS	Output Presentation Services
OSD	Open Server Dimension (up to now: Open Systems Direction)
PAM	Primary Access Method
PCS	Performance Control Subsystem
PFA	Performant File Access
POSIX	Portable Open System Interface for UNIX
PVS	Public Volume Set (Pubset)
RAID	Redundant Array of Independent Disks
RFA	Remote File Access
ROBAR	ROBot ARchive
RSO	Remote SPOOL Output
RU	Release Unit
SAM	Sequential Access Method
SDF	System Dialog Facility
SECOS	Security Control System
SF pubset	Single-Feature pubset
SHC-OSD	Storage Host Component
SM pubset	System Managed pubset
SMS	System Managed Storage
SNMP	Simple Network Management Protocol
SOLIS	SOftware suppLy and Information System
SPVS	Shared Public Volume Set
SFID	Save File Identification
SRDF	Symmetrix Remote Data Facility
SRPM	System Resources and Privileges Management
SV	SerVer
SVID	Save Version Identification

SYSSII	Structure and Installation Information
TCP/IP	Transmission Control Protocol / Internet Protocol
TEMPFILE	TEMPorary FILE
TFT	Task File Table
TIAM	Terminal Interactive Access Method
TLS	Tape Library System
TPR	Task Privileged
TSN	Task Sequence Number
TSOS	Time Sharing Operating System
TU	Task Unprivileged
UDS	Universal Database System
UFS	Unix File System
IDRC	Improved Data Recording Capability
VOL	Volume Header Label
VSN	Volume Serial Number

14 Related publications

You will find the manuals on the internet at http://manuals.ts.fujitsu.com. You can order printed versions of manuals which are displayed with the order number.

[1] **HSMS V12.0A** (BS2000)
Hierarchical Storage Management System
Volume 2: Statements
User Guide

[2] **ARCHIVE** (BS2000)
User Guide

[3] **BS2000 OSD/BC**
Utility Routines
User Guide

[4] **BS2000 OSD/BC**
Introduction to DVS
User Guide

[5] **SDF** (BS2000)
SDF Dialog Interface
User Guide

[6] **BS2000 OSD/BC**
Introduction to System Administration
User Guide

[7] **IMON** (BS2000)
Installation Monitor
User Guide

[8] **BS2000 OSD/BC**
Commands
User Guide

[9] **BS2000 OSD/BC**
Executive Macros
User Guide

[10] **MAREN** (BS2000)
Volume 1: Basics of MTC Management
Volume 2: User Interfaces
User Guide

[11] **NFS** (BS2000)
Network File System
User Guide

[12] **POSIX** (BS2000)
POSIX Basics for Users and System Administrators
User Guide

[13] **POSIX** (BS2000)
Commands
User Guide

[14] **SDF-A** (BS2000)
User Guide

[15] **SDF-P** (BS2000)
Programming in the Command Language
User Guide

[16] **SECOS** (BS2000)
Security Control System
User Guide

[17] **openSM2** (BS2000)
Software Monitor
User Guide

[18] **BS2000 OSD/BC**
system-managed storage
User Guide

[19] **PROP-XT (**BS2000**)**
Programmed Operating with SDF-P
Product Manual

[20] **HIPLEX MSCF** (BS2000)
BS2000-Processor Networks
User Guide

[21] **SHC-OSD** (BS2000)
Storage-Management für BS2000
User Guide

[22] **BS2000 OSD/BC**
DMS Macros
User Guide

[23] **JV** (BS2000)
Job Variables
User Guide

[24] **openNet Server** (BS2000)
BCAM
User Guide

[25] **SQ Series Business Servers**
Operation and Administration
User Guide

[26] **BS2000 OSD/BC**
Performance Handbook

[27] **BS2000 OSD/BC**
Diagnostics Handbook
User Guide

[28] **BS2000 OSD/BC**
Accounting record
User Guide

[29] **interNet Services** (BS2000)
Administrator Guide