GCOS7

System Administrator's Manual

DPS7000/XTA
NOVASCALE 7000

Operating System: Administration

REFERENCE
47 A2 54US 02

DPS7000/XTA NOVASCALE 7000 GCOS7

System Administrator's Manual

Operating System: Administration

June 2002

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

REFERENCE
47 A2 54US 02

Printed in France

Suggestions and criticisms concerning the form, content, and presentation of this book are invited. A form is provided at the end of this book for this purpose.

To order additional copies of this book or other Bull Technical Publications, you are invited to use the Ordering Form also provided at the end of this book.

Trademarks and Acknowledgements

Preface

Scope and Objectives

This manual groups together information which is of particular use to the System Administrator. It covers a very wide spectrum of topics. Some, such as Site Catalog management, environment management, and system accounting, are described in detail. This manual is the primary source of information for these topics. Other topics, such as job management and system operation are described with less detail than in other documents, but also from a different perspective.

The document is split implicitly into three parts:

- Chapters 1 to 7 discuss general principles and recommendations
- Chapters 8 to 13 explain specific administrative procedures
- Appendices A to D provide reference material

Intended Readers

This manual is intended for whoever is responsible for the setting up and overall management of the system. Typically, this is somebody who has a supervisory position and a practical grounding in DP operations. A Site Manager for example.

This manual is also intended for anyone to whom the Site Manager delegates administrative functions and for anyone accorded the right to log on as a SYSADMIN user.

Structure

Chapter 1 — *Role of the System Administrator*

Gives a short definition of who the System Administrator is and a summary of the System Administrator's main responsibilities.

Chapter 2 — *Basic Administration Concepts*

Discusses the fundamental topics: projects, environments, catalogs, data security, and data integrity.

Chapter 3 — *Installation and Configuration*

Summarizes the main concepts and utilities associated with installing, configuring, and updating GCOS 7.

Chapter 4 — *Communications: Generation and Support*

Summarizes the components and protocols which implement the system's network communications and support.

Chapter 5 — *System Operation*

Describes job class conventions, file and volume naming conventions, and also gives an overview of automated operation and coupled systems operation.

Chapter 6 — *System Regulation*

Describes how the system's mechanism and principles for self-regulation can be adapted to new operational and optimization objectives.

Chapter 7 — *System Maintenance*

Gives a short overview of Service Request procedures, Telemaintenance facilities, system dumps and trace facilities.

Chapter 8 — *Project/User/Billing/Station/Application*

Management procedures

Gives the procedures and command syntax for updating the Site Catalog.

Chapter 9	*Environment Management Procedures* Describes the standard GCL command environments and the procedures for adapting these environments to site requirements.
Chapter 10	*Operator Management Procedures* Describes the different types of operator and the procedures for setting operator access rights.
Chapter 11	*File and Volume Management Procedures* Describes file protection and recovery procedures, file security procedures, VBO<->FBO file migration, disk space management facilities (VOLSET and QUOTA), Automated Storage Manangement (ASM 7), and mirror disk facilities.
Chapter 12	*Job Management Procedures* Describes how to reduce job input overheads, ensure data security, and implement job checkpoint/restart and job accounting procedures.
Chapter 13	*System Regulation Procedures* Gives practical advice and procedures for balancing the system workload.
Appendix A	*The Standard Families* Gives a reference list of the GCL command families delivered with the system.
Appendix B	*Environment Management Examples* Gives practical examples on adapting the standard GCL command environments to site requirements.
Appendix C	*Environment Generation Using GCL* Summarizes the environments and associated facilities that are created by the STANDARD_ENVT command.
Appendix D	*System Accounting Records* Gives the COBOL and GPL declarations and detailed field descriptions of the system accounting records.

Bibliography

Installation

GCOS 7-V8 System Installation, Configuration and Updating Guide ... 47 A2 19US
GCOS 7-V9 System Installation, Configuration and Updating Guide ... 47 A2 23US
GCOS 7-V7 TDS Administrator's Guide ... 47 A2 32UT

Communications

Network Overview ... 47 A2 92UC
Network Generation Guide ... 47 A2 93UC
Network User's Guide ... 47 A2 94UC
UFT User's Guide ... 47 A2 13UC
DJP User's Guide ... 47 A2 14UC

File and Volume Management

Data Management Utilities User's Guide ... 47 A2 26UF
GCOS 7 Guide to Security ... 47 A2 17UG
File Recovery Facilities User's Guide ... 47 A2 37UF
Administering the Storage Manager ... 47 A2 36UF
Catalog Management User's Guide ... 47 A2 35UF
File Migration Tool User's Guide ... 47 A2 32UF
Library Maintenance Reference Manual ... 47 A2 01UP
Mirror Disks User's Guide ... 47 A2 20UF
Introduction to ASM 7 ... 47 A2 48US
EpochBackup7 Installation and User's Guide ... 47 A2 49US
ASM 7 File Migration Administrator's Guide ... 47 A2 51US

System Operation

System Operator's Guide ... 47 A2 53US
DPS 7000/2x0/3x0 Operator Reference Manual ... 77 A1 52UU
DPS 7000/An Operator's Guide ... 77 A1 71UU
DPS 7000/4x0 Operator's Guide ... 77 A1 91UU
DPS 7000/5x0/7x0 Operator's Guide ... 77 A1 81UU
DPS 7000/5x0/8x0 Operator's Guide ... 77 A1 94UU
GCOS 7 Console Messages Directory ... 47 A2 61UU
Messages and Return Codes Directory ... 47 A2 10UJ
JCL Reference Manual ... 47 A2 11UJ
JCL User's Guide ... 47 A2 12UJ
IOF Terminal User's Reference Manual
Part 1: ... 47 A2 38UJ
Part 2: ... 47 A2 39UJ
Part 3: ... 47 A2 40UJ
DOF 7-PO User's Guide ... 47 A2 80UC
DOF 7-SM User's Guide ... 47 A2 84UC
Coupled Systems User's Guide ... 47 A2 30UF

Resource Management

System Behavior Reporter User's Guide ... 47 A2 03US
TILS User's Guide ... 47 A2 04US

ARM User's Guide *47 A2 11US*

Program Preparation
IOF Programmer's Manual *47 A2 37UJ*
GPL System Primitives Reference Manual *47 A2 34UL*
GCL Programmer's Manual *47 A2 36UJ*

Syntax Notation

The following conventions are used for presenting GCL command syntax.

ITEM — An item in upper case is a literal value, to be specified as shown. The upper case is merely a convention; in practice you can specify the item in upper or lower case.

item — An item in lower case is non-literal, indicating that a user-supplied value is expected.

In most cases it gives the type and maximum length of the value:

`char105`	a string of up to 105 alphanumeric characters
`dec5`	a decimal integer value of up to 5 digits
`name31`	a name of up to 31 characters
`file78`	a file name of up to 78 characters
`star31`	a star name of up to 31 characters

In some cases, it gives the format of the value:

`a`	a single alphabetic character
`nnn`	a 3-digit number
`hh.mm`	a time in hours and minutes

In other cases, it is simply descriptive of the value:

```
device-class
condition
any-characters
```

ITEM — An underlined item is a default value.

bool — A boolean value which is either 1 or 0. A boolean parameter can be specified by its keyword alone, optionally prefixed by "N". Specifying the keyword alone always sets the value to 1. Prefixing the keyword with "N" always sets it to 0.

{ } — Braces indicate a choice of values. Only one may be selected.

[ ]	Square brackets indicate that the enclosed item is optional. An item not enclosed in square brackets is mandatory.
( )	Parentheses indicate that a single value or a list of values can be specified. A list of values must be enclosed by parentheses, with each value separated by a comma or a space.
...	Ellipses indicate that the item concerned can be specified more than once.
+=$*/	These are literal characters to be specified as shown.
- - -	All parameters below a dashed line do not appear in the help menus.

EXAMPLE 1:

```
        { IMMED               }
[ WHEN={ [dd.mm.yy.]hh.mm   } ]
        { +nnnn{W|D|H|M}      }
```

❑

This means you can specify:

- Nothing at all (WHEN=IMMED applies).
- WHEN=IMMED (the same as nothing at all).
- WHEN=22.30 to specify a time (and today's date).
- WHEN=10.11.87.22.30 to specify a date and time.
- WHEN=+0002W to specify 2 weeks from now.
- WHEN=+0021D to specify 21 days from now.
- WHEN=+005H to specify 5 hours from now.
- WHEN=+0123M to specify 123 minutes from now.

EXAMPLE 2:

```
[ PAGES=(dec4[-dec4] ...) ]
```

❑

An optional parameter giving either a single value or a list of values enclosed in parentheses, with each value separated by a comma or space. Each value can consist of either a single number or a pair of numbers connected by a hyphen. For example:

```
PAGES=(2,4,10-25,33-36,78 83 90)
```

EXAMPLE 3:

```
[ REPLACE ={ bool | 0 } ]
```

❑

A boolean parameter whose default value is zero. You can specify:

- Nothing at all (REPLACE=0 applies)
- REPLACE=0 or simply NREPLACE
- REPLACE=1 or simply REPLACE

Table of Contents

1. Role of the System Administrator

2. Basic Administration Concepts

3. Installation and Configuration

4. Communications: Generation and Support

5. System Operation

6. System Regulation

7. System Maintenance

8. Project/User/Billing/Station/Application Management Procedures

9. Environment Management Procedures

10. Operator Management Procedures

11. File and Volume Management Procedures

12. Job Management Procedures

13. System Regulation Procedures

A. The Standard Families

B. Environment Management Examples

C. Environment Generation Using GCL

D. System Accounting Records

E. Clean the IOF Mailbox

Index

Table of Graphics

Figures

Tables

1. Role of the System Administrator

The System Administrator is the person responsible for the overall management and co-ordination of the system. Typically, this is somebody who has a supervisory position and a practical grounding in DP operations. A Site Manager for example.

The range of responsibilities depends on the size of the installation and on the nature of its operations. On large installations running a mix of interactive, batch, remote, database, and transactional applications, administrative responsibilities are usually delegated. The System Administrator is often accompanied by specialist administrators such as the TDS Administrator, the IQS Administrator, and the Networks Administrator. This manual, however, concentrates on the general aspects of GCOS 7 system administration which fall into three broad categories:

- Set-up operations
- Daily operations
- Periodic operations

These consist mostly of the following kinds of activity:

- Project and User management
- Site Catalog management
- Access Rights management
- File and Volume management
- Environment management
- Resource management
- Operator management
- System maintenance

Whilst it is convenient to separate management activities in this manner, they are in practice interrelated activities whose combined, overall objective is to ensure:

- smooth *system operation*
- proper *system security*
- good *system performance*
- correct levels of *data security*
- high levels of *data integrity*

These are the guiding concerns that must underlie all off-line and on-line procedures laid down by the System Administrator for the site.

2. Basic Administration Concepts

This chapter provides a brief review of topics which are fundamental to GCOS 7 administration:

- Projects
- User Management
- System Security
- System Accounting
- Operator Management
- Environment Management
- File Management
- Directories
- Catalogs
- Site Catalog
- Data Security
- Data Integrity

These are concepts and concerns which underlie nearly all areas of system administration. They should be understood, at least in principle, before configuration of the system begins.

2.1 Projects

For the System Administrator, the most fundamental GCOS 7 concept is the project. It plays a key part in nearly all areas of system management.

In simple terms, a project is a list of facilities (applications, files, commands, printers, etc.) and a list of users who are allowed to log on to the project and use these facilities.

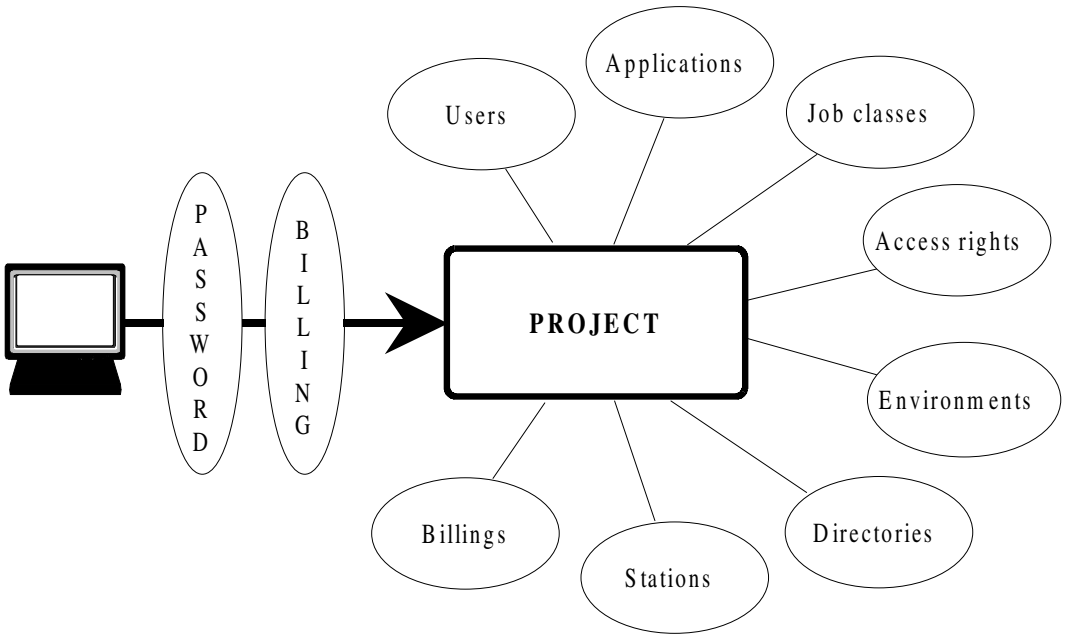

Figure 2-1. Project Concept

This is a practical but somewhat limited definition of a project. The System Administrator should also see projects as an important administrative strategy in:

- user management
- operator management
- environment management
- file management
- system security
- system accounting

2.1.1 Projects and User Management

Projects are a means of organizing the system's users into manageable, functional, and accountable subgroups. Each subgroup belongs to a project, and each project is configured according to the needs and context of the subgroup.

Typically, persons performing the same kind of task within the same operational unit of the company belong to the same project. The project structure may thus reflect the organizational structure of the company itself. This is not obligatory however, and any configuration is possible including one project per user, or even a single project for all users. The essential point is that all individuals belonging to the same project have the same rights and the same restrictions. The System Administrator must keep this in mind rather than any social organization, which may or may not be based on such criteria.

2.1.2 Projects and Operator Management

System Operators are experienced persons responsible for the day-to-day running, monitoring, and maintenance of the system. They are a special category of user, and like all users they must belong to a project. This is usually the standard OPERATOR project or a project created by the System Administrator and modelled on the OPERATOR project.

With large DPS 7000 installations in particular, System Operators have a great variety of tasks to perform and it is important that the System Administrator define clearly who does what. To facilitate this, GCOS 7 allows two different types of operator project to be defined, namely:

- the Main Operator project
- the Station Operator project

Users of Main Operator projects are known as Main Operators or System Operators, and have access to commands which provide direct supervision of the system. These are called Main Operator commands and some, the MODIFY_DIMENSION command for example, are very powerful. The OPERATOR project is itself a Main Operator project.

Users of Station Operator projects are known as Station Operators, and have access to a reserved set of output control commands for managing device queues and printers.

2.1.3 Projects and Environment Management

An environment is a set of GCL commands. Different environments consist of different commands, though some commands can be common to more than one environment.

A project usually has a default environment, and one or more alternative environments. If the user does not specify an environment at log-on, the default environment is assumed. If a project has no attached environments, its users have access to the entire range of GCL commands.

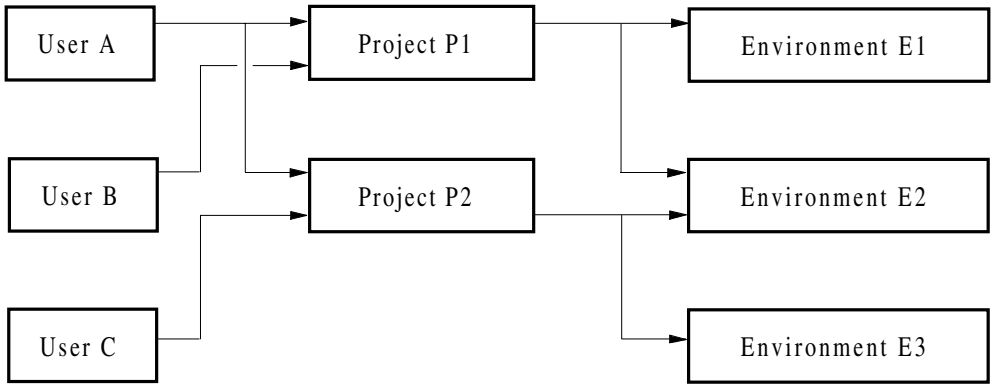

Figure 2-2. Project Environments

Attaching environments to projects is one means of limiting the use of certain commands to certain categories of users. The following are some reasons why this may be necessary:

- certain commands may be hazardous and may cause lasting damage if used inappropriately or inadvertently by inexperienced users.
- some commands may cause operating difficulties or consume too many resources to be allowed to all users, e.g. interactive compilers.
- some commands may encourage the user to ignore installation-defined rules or procedures, e.g. the use of uncataloged files.
- some commands may be simply irrelevant to a user's designated set of tasks, e.g. FORTRAN commands in a COBOL programming context.

It is also widely recognized that, the wider the set of available commands, the more difficult the choice, especially for inexperienced users.

GCOS 7 is delivered with seven standard environments, each suited to a particular kind of project. For example, there is the PROGRAM_PREP environment for COBOL programmers and the TDSAPPL_PREP environment for TPR programmers.

The System Administrator can also build new environments by modelling them on the standard environments and families (command subsets).

2.1.4 Projects and File Management

The System Administrator is strongly advised to exercise some control over the inevitable proliferation of user files in the system. The most obvious measure is to organize user files on a project basis. In practice, this means implementing an installation-wide naming convention in which file names are prefixed by the name of the project under which the files are created. The best way of encouraging users to respect this convention is to make it both easy and beneficial for them to do so.

The System Administrator should therefore allocate a catalog to each project and give the catalog the same name as the project. This provides the project users with a directory structure in which their file names are automatically prefixed with the name of their project. The main benefit to the user is that they can name their files as they please, regardless of what names are being used in other projects or even in other subdirectories of their own project.

There are other benefits, but these are discussed later in this chapter under "Catalog Concepts".

2.1.5 Projects and System Accounting

GCOS 7 has an internal accounting mechanism which continuously monitors the resource usage across the system, but which can also be used for project accounting purposes.

To do this, the System Administrator must allocate each project a billing. A project billing is the name of an account to which all resources consumed by the project are charged. A project usually has a default billing and, if its scope justifies it, one or more additional billings. A single billing can also be shared by more than one project.

Each time a project uses some resource (CPU, memory, disk controller, etc.) this is registered in a system file SYS.ACT1 or SYS.ACT2. Each record written contains details of the resource usage and also the identity of the project, user, and billing associated with the resource usage. In order to extract this data for project accounting purposes, it must be dumped from the SYS.ACTi file to a UFAS sequential file using the DUMPACT facility, and then processed by the EDITACT facility or by a customer application.

2.1.6 Projects and System Security

By their very nature, projects provide two levels of security. They ensure that:

- those who are not allowed to use the system, are not able to do so (project logon rights)
- those who are allowed to use the system, are not likely to cause it any damage (project access rights)

Anyone new wanting to use the system must apply to the System Administrator for a user name, a private password, and the name of each project they can log on to. It is up to the System Administrator to implement appropriate off-line 'vetting' procedures and to judge which projects are suitable for the individual concerned. Inexperienced users, for example, should not be allowed on to the System Administrator or System Operator projects. These projects provide access to vital control facilities which, if misused, could cause serious damage to the system's security.

2.2 Catalogs

Catalogs are used to manage access to files, volumes, generation groups, TPRs, and other GCOS 7 objects. Primarily, however, they are a file management and file access tool and it is from this perspective that they are discussed here.

A catalog is a set of directory names and file names linked, tree-like, under a common root. The root is an implicit starting point and by convention has no name. Immediately below the root is the master directory name; this is also the name of the catalog. Under the master directory name is at least one file name and possibly one or more directory names. Under these are more file names and perhaps further directory names, and so on.

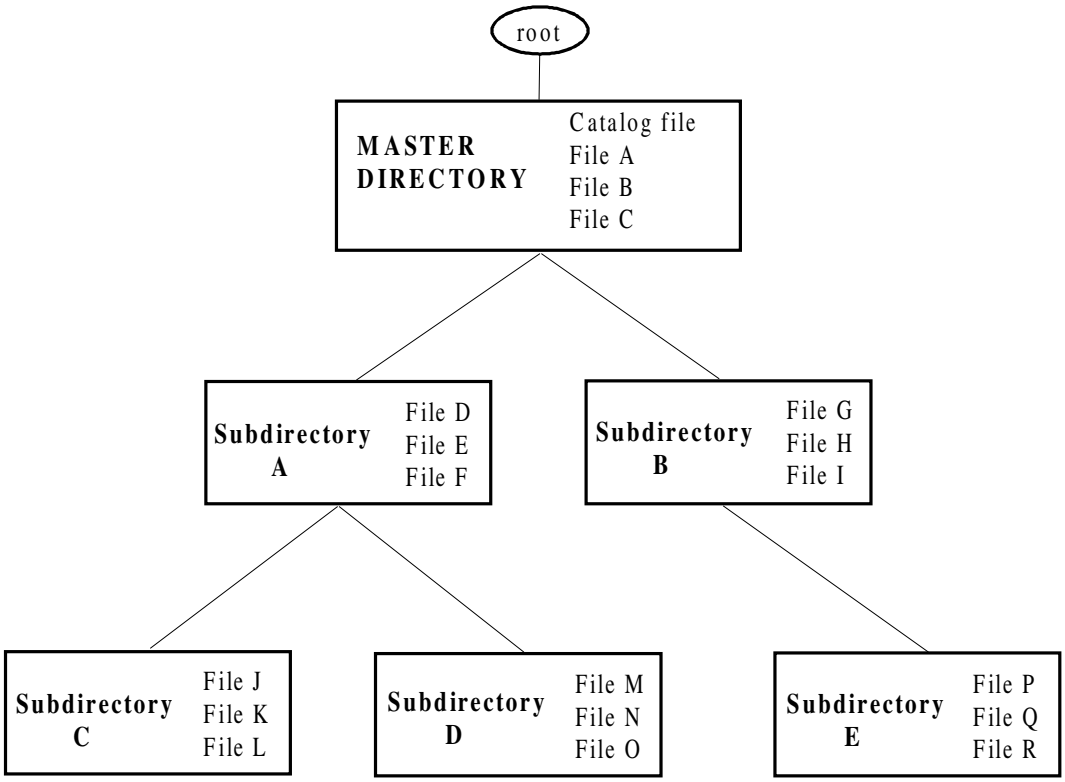

Figure 2-3. Catalog Concept

Any file whose name is registered in a catalog is called a cataloged file. A cataloged file is uniquely identified by its 'full path name', that is, its simple name prefixed by all the directory names between it and the root. All full path names therefore begin with the catalog's master directory name. If, as is recommended, the master directory has the same name as the project, then all file names in the project catalog begin with the name of the project.

2.2.1 The Project Catalog

The project catalog, or private catalog as it is also called, has three particular functions:

- It provides a directory structure in which file names are automatically prefixed with a name chosen by the System Administrator. This simplifies file management and allows users to name their files without being directly concerned with the uniqueness of these names.
- It allows users to reference files without having to specify, or even to be aware of, their physical location.
- It maintains a record of the project's file access rights.

Figure 2-3 above describes a catalog from the user's stand-point, that is, as a tree-like list of file names. However, from the system's stand-point, a catalog is in fact a BFAS file containing an index linked list of file 'descriptions'. Each file description includes:

- the file's path name (its logical identity)
- the file's media name and device class (its volume identity)
- the file's associated access rights

When referencing a (cataloged) file, one needs specify only its path name. The volume details and access rights are automatically retrieved from the file's catalog description.

An interesting feature of all catalogs is that they contain a file description of themselves. By convention a catalog file has the path name "<master-directory-name>.CATALOG", which means, in effect, that it is cataloged directly under its own master directory. When a catalog is created it contains initially one directory name (the master directory) and one file name (its own name).

2.2.2 The Site Catalog

The Site Catalog, called SITE.CATALOG, is the particular responsibility and major tool of the System Administrator. It is, in effect, the master control of all data and of all access to data in the system.

It contains three categories of information:

- The file descriptions of all site files. In keeping with the catalog naming convention, these files are prefixed with the name "SITE", for example SITE.STARTUP, SITE.QUOTA and, of course, SITE.CATALOG itself.
- The file descriptions of all project catalogs. The Site Catalog is the catalog of catalogs. It controls access to all master directories and thus indirectly to all cataloged files and other objects throughout the site.
- The project, user, billing, and station descriptions. The Site Catalog registers all user names, passwords, project names, billings, stations, environments, catalogs, startups, and so on.

2.3 Data Security

On all computer systems it is necessary to ensure that certain data cannot be accessed without authorization, either because its content is private or because unauthorized access could change its validity.

The main requirements are these:

- Data files must be protected against unauthorized access.
- The owner of the data must be able to define who can access it, and how it can be accessed.
- The System Administrator must be able to have ultimate control of the data protection mechanism.

Data security on GCOS 7 systems is ensured by the Access Rights mechanism.

2.3.1 Project Access Rights

To protect data using the access rights available on GCOS 7, we need to:

- identify who is submitting work to the system
- define which objects are protected and the access rights associated with them.

Access rights are given to projects on the following objects:

- files
- catalogs
- directories
- generation groups
- volumes
- MCS applications
- TDS transactions
- RBF stations
- environments.

The protection of these objects is ensured in a centralized manner as follows:

- The descriptions of protected objects and their access domains are stored in catalogs. A cataloged object can be accessed only through a catalog, and each time it is accessed, the type of access (READ, WRITE, etc.) is checked against the Access Control List (ACL) for the object, which is also stored in the catalog.
- The projects and their associated information (users, billings, etc.) are stored only in the Site Catalog.
- The logging of access right violations.

2.3.2 Access Rights Management

It is the System Administrator, via the SYSADMIN project, who installs, coordinates, and controls data protection in the system.

The SYSADMIN project can:

- access all objects without any control
- manage the identities of users known to the system using USER, PROJECT, and BILLING
- manage project profiles (TDS authority codes, attributes, job classes, etc.)
- catalog projects, stations, and master directories
- give access rights on objects to other projects
- make a project the owner of a master directory
- manage the list of volume names associated with a project.

2.3.3 Access Rights Violations

GCOS 7 has a number of logging and reporting features that can aid in the detection of possible threats to security in an installation. If requested, all unauthorized attempts to access protected resources are logged in the System Accounting Files SYS.ACT1 or SYS.ACT2. This information can help the installation to analyze and control its computer resources more effectively.

2.3.4 Access Rights on System Files

The SYSADMIN project is the owner of the System Catalog (SYS.CATALOG) and therefore owner of all system files. The System Administrator may subsequently set access rights for individual projects on individual system files as and when required. Alternatively, a default set of access rights can be set on all system files using the GCOS Installation and Updating Facility (GIUF), which is discussed further in Chapter 3.

2.4 Data Integrity

Not even a high quality and durable system like the DPS 7000 can completely guarantee against software or hardware failure. It does, however, provide a range of file protection and file recovery facilities which can prevent data from being lost or corrupted as a result of such failures.

File protection and recovery are provided through the combination of a checkpoint mechanism and a file journalization service. The checkpoint mechanism defines at which points in time data can be considered by the system to be in a consistent state. These checkpoints, or commitments as they are called in the TDS context, are used as points of reference whenever files with journal protection are corrupted or lost.

If a file with Before Journal protection is corrupted, it is *rolled back* to its state at the last checkpoint. If a file with After Journal protection is lost, a previously saved version can be restored and then rolled forward to its state at the last checkpoint.

2.4.1 The File Recovery Unit (FRU)

An FRU is a portion of a program delimited explicitly or in effect by two checkpoints. It takes a file or set of files from one consistent state at the beginning of its execution (first checkpoint) to another consistent state at the end of its execution (second checkpoint).

A program consists of one or more FRUs of the following type:

- an entire Batch or IOF step
- part of a Batch or IOF step between two successive checkpoints
- a TDS commitment unit
- an IQS query

All CIs (or pages) being updated by an FRU are locked against access by other FRUs. If the updating FRU terminates normally, the modified data is then committed and made accessible to the next FRU.

If the updating FRU terminates abnormally, then any physical updates since the beginning of the FRU are either rolled back or not applied, depending on whether the updates are immediate or deferred. In either case, the file is maintained in a consistent state.

2.4.2 Before Journal Protection

The Before Journal method protects user files against incomplete updates that might result from a program abort or system crash.

While an FRU is updating a file, the system writes its images before they are updated to the Before Journal. These are called the Before Images. If the updating FRU terminates abnormally, the records updated since the beginning of the FRU are overwritten by their Before Images. The file is thus rolled back to its consistent state at the beginning of the FRU.

Rollback takes place immediately the FRU aborts, or at warm restart if the system crashes. It is a dynamic process and requires no operator action. If rollback fails, this is reported in the Job Execution Report (JOR), and the incident causing the failure is logged in the SYS.ERLOG file.

2.4.3 After Journal Protection

The After Journal method protects a user file from being lost as a result of physical damage to the file's media or logical destruction of the file itself.

The After Journal contains a copy of all updates to a file since its last save date and up to its most recent checkpoint. These updated records are called After Images. If the file is destroyed, its previously saved version can be rolled forward to its most recent checkpoint by overwriting its records with their successive After Images.

To recover a lost file, the operator restores its saved version using a file restoration utility and then rolls forward the restored file using the ROLLFWD utility.

After Journalization also provides protection against faulty recovery, where immediate recovery, rollback, or rollforward cannot be executed correctly.

2.4.4 After Journal with Deferred Update Protection

Like the Before Journal method, the Deferred Update method protects files against incomplete updates that might arise from a program abort or system crash. The Deferred Update is the more efficient method, but it is only available for TDS files and must be used with After Journal protection.

When an FRU is updating a file, the system keeps the After Images in a buffer and only writes them to the After Journal when the FRU terminates successfully. The After Images are then applied from the After Journal to the file itself. In this way, a file is not updated during the FRU but *dynamically rolled forward* at the end of the FRU.

If the FRU terminates abnormally, the After Images are discarded and not written to the After Journal, and hence not applied to the file. The file therefore remains in its consistent state in at the beginning of the FRU.

2.4.5 Journalization Advanced Service (JAS)

JAS is the GCOS 7 file protection and recovery service. It comprises the Before Journal file, the After Journal directory, and facilities associated with the After Journal.

A system can have more than one JAS, and each JAS is in charge of the integrity of a specific set of files linked to it.

JAS is available in HA and non-HA environments:

- the non-HA type of JAS is called SYS JAS. This provides the traditional (i.e., non-switchable) file recovery and protection facilities.
- the HA type of JAS consists of BLUE JAS and GREEN JAS. These provide file recovery and protection services which can be taken over by a backup Member if the active Member fails.

2.4.6 The JAS Directory

Associated with each JAS is a cataloged directory containing the names, physical characteristics, and sequencing of the After Journal files.

The SYS JAS directory is cataloged under the System Catalog, and its name is SYS.JADIR. Each HA-type JAS directory is cataloged under a dedicated private catalog.

The space used in the JAS directory is a function of the number of active journal files and the number of FRU aborts. Consequently, the percentage space used by the JAS directory is continuously increasing. The System Administrator must therefore implement regular space recuperation measures which prevent an overflow on the JAS directory. This is done using MAINTAIN_JAS commands combined with file saving and journal dumping utilities.

2.4.7 Multiple After Journals

The number of After Journals is increased to three, giving the possibility to split the journalization of the files protected by the After Journal.

This feature gives a similar level of functions that the High Availability feature of GCOS 7 on a single system. Of course there is no switching survey, removing the need of using the Complex Management. The three possible journal files are referred to as the SYS JAS, the BLUE JAS and GREEN JAS (JAS standing for Journal Advanced Service). If the option "HA" is installed on the site, then this site remains controlled by the CMSC complex.

The journalization is done in the After Journal files linked to the catalog in which the files to be journalized are catalogued. A TDS step can only use one After Journal file, and a Batch or an IOF step can use one to three JAS. When one of the two private JAS is used by a TDS step, both journalizations (BEFORE and AFTER) are automatically activated.

A catalogued file is linked to one of the three JAS with the LINK command of the MNJAS utility. The catalog must be "auto-attachable". FBO files only may be attached to BLUE JAS or GREEN JAS.

Note that is not possible to use this function when TCRF is used or if the files are protected by RDDF7.

2.4.8 General Recommendations

- Files to be protected should be cataloged with JOURNAL=BOTH. Protection specified at catalog level overrides that specified at step level. This resolves any contradictory requests that might arise from steps journalizing simultaneously on the same file. Specifying BOTH ensures the maximum level of protection.
- TDS files should have After Journal protection with Deferred Updates for faster performance.
- Files with After Journal protection should be cataloged in auto-attachable catalogs, so as to avoid all risk of duplicate names.
- The BJSIMU configuration parameter and an appropriate size for the SYS.JRNAL file should be used for optimizing the Before Journal performance.
- The JAS directory should be on disk, as this provides faster performance, simpler configuration, use of the Journal Recovery Utility (JRU), and non-conflicting use of static and dynamic rollforward.
- For greater security, protected files should not be on the same volume as the JAS directory.

- The primary journals should be transferred periodically (daily even) to tape or cartridge so as to conserve sufficient reserve space for new journals and to limit the number of disks used. The JAS directory should not be allowed to exceed 80% of its full capacity. Check its fill rate regularly.
- In a non-HA environment, there should be at least one spare system disk. If the system disk is full, a tape copy should be made. In some cases, the tape copy can provide the only way to restart the system without losing files.

3. Installation and Configuration

This chapter summarizes the main concepts and utilities associated with installing, configuring and updating GCOS 7.

- Topics include:
- the GCOS 7 Domains
- PO, P2P, and RP-Configurations
- Installation media
- Phase 0 and Phase 1 installation
- Technical Status Updating
- the IUF facility
- CONFIG utility
- TAILOR utility
- System files
- Multi-system installation

Full details on all these topics are given in the *System Installation, Configuration, and Updating Guide*.

3.1 The MAIN Concepts

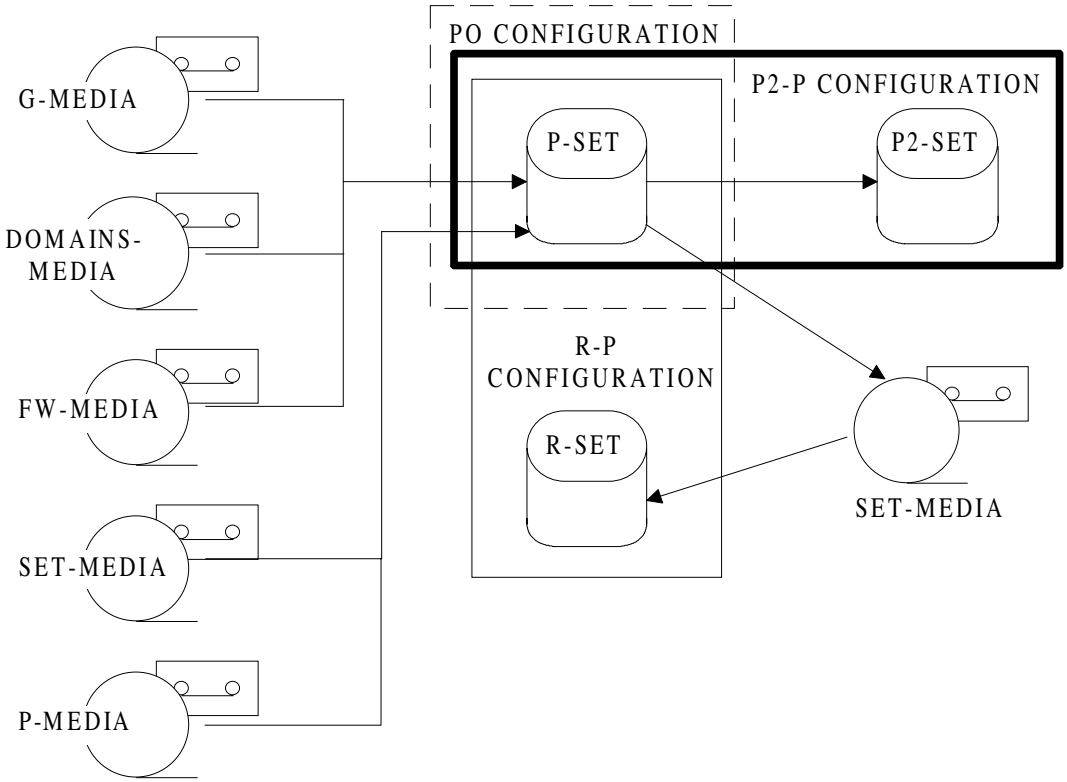

Figure 3-1. Installation Overview

3.1.1 The GCOS 7 Domains

The GCOS 7 operating system consists of five separate components, called domains. These are as follows:

- GCOS domain
- FIRMWARE domain
- OLTD domain (On-line Test and Diagnostics)
- GSF domain (GCOS 7 Service Facility)
- DSA domain (for DNS and/or CNP7 front-end communications processors)

The GCOS, OLTD, GSF, and FIRMWARE domains must be present on each working site and are always supplied. The DSA domain, however, is optional and can be delivered and installed separately.

3.1.2 The Production-Only (PO) Configuration

This consists of a single production disk set, called the P-Set.

It is the least expensive configuration, but also the least secure. For this reason, you can alternatively convert to:

- either an RP-Configuration
- or, preferably, a P2P-Configuration

3.1.3 The Production and Second Production (P2P) Configuration

This consists of two identical production disk sets, the P-Set and the P2-Set. The P-Set alternates with the P2-Set as the running set.

The non-running set is updated just before it becomes the running set. The non-running set is conserved in its non-updated form for immediate backup purposes in case of problems with the newly updated running set. When the updated running set is working correctly, and any problems have been ironed out, the non-running set can be updated in line with the running set.

3.1.4 The Reference-Production (RP) Configuration

This consists of a reference disk set (the R-Set), and a first production set (the P-Set). The R-Set is the backup set. P-Set is the running set, and the set to be updated.

The R-Set contains all the files necessary for IUF operations and is used to create or update the P-Set.

If there are problems with the updated set, the R-Set is always available for backup purposes.

As a general rule, the R-Set is only upgraded just before applying the next update to the P-Set, that is, as late as possible.

3.1.5 Installation Media

One or more of the following set of installation tapes or cartridges will have been supplied to you, depending on whether you are a New User or an Update User.

DOMAINS-MEDIA	contains the domains OLTD, DSA (for DNS and/or CNS7), FW, and GSF. In certain cases, there is a separate FW-MEDIA.
FW-MEDIA	contains the firmware files of the FIRMWARE domain.
G-MEDIA	contains the system files of the GCOS domain.
K-MEDIA	contains a software validation key.
P-MEDIA	contains a factory-prepared image of the P-Set.
SET-MEDIA	contains GCOS 7 and the associated domains for building the P-set.
STARTER-MEDIA	contains an image of the MS/D500 disk or MS/B10 disk (as appropriate), and is provided for updating GCOS 7-V3x7 systems.

3.1.6 New User

If you are installing GCOS 7 on a new machine, you are considered to be a New User. A new DPS 7000/An is delivered with GCOS 7 already installed. A new DPS 7000/xx0 is delivered with an image of the P-Set which is restored by Bull personnel.

New Users of DPS 7000/An are supplied with factory-prepared P-Set disks, containing all the system files for the GCOS, OLTD, CNS7, GSF and FW domains. P-MEDIA, DOMAINS-MEDIA, and FW-MEDIA are also supplied for backup.

New Users of DPS 7000/xx0 are supplied with a factory-prepared P-MEDIA and, for backup, with a DOMAINS-MEDIA and FW-MEDIA.

All New Users may also be supplied with a K-MEDIA.

3.1.7 Update User

If you are evolving from a GCOS 7 Release to another GCOS 7 Release, you are considered to be an Update User.

All Update Users are supplied with installation media G-MEDIA, DOMAINS-MEDIA, and FW-MEDIA for building the P-Set. V3x7 users are also supplied with a STARTER-MEDIA.

In addition, Update Users may also be supplied with a K-MEDIA or a SET-MEDIA, though some may be supplied with a P-Media instead of a STARTER-MEDIA or a SET-MEDIA.

3.1.8 Phase 0 Installation

This phase builds the P-Set.

For new DPS 7000/An systems, Phase 0 is performed at the factory. For new DPS 7000/xx0 systems, Phase 0 is performed on site by Bull personnel.

For systems updating to a new release, Phase 0 is usually performed by the System Administrator.

3.1.9 Phase 1 Installation

This phase validates the P-Set and then builds the P2-Set for a P2P-Configuration, or the R-Set for an RP-Configuration. It is applicable to all new systems and to all systems being updated.

3.1.10 Technical Statuses

A system can be updated from one Release to another or, on a more frequent basis, from one Technical Status to another.

In order that you can install a new Technical Status, the Distribution Center supplies a G-MEDIA containing new files or modules to replace existing files or modules on the system set to be updated. An accompanying Customer Service Bulletin explains the installation procedure.

For a PO-Configuration or an RP-Configuration, the set to be updated is the P-Set.

For a P2P-Configuration, either the P-Set or the P2-Set can be updated. Usually the non-running set is chosen. The other set can be updated after the new system has been running smoothly for a probationary period of time.

3.2 The MAIN Utilities

3.2.1 The IUF Facility

The Installation and Updating Facility (IUF) provides the necessary commands for installing and updating the GCOS 7 domains. It consists of:

- GIUF for installing and maintaining the GCOS domain
- FGF for installing and maintaining the FIRMWARE domain
- TDF for installing and maintaining the OLTD domain
- GSF for installing and maintaining the GSF domain
- DIUF for installing and maintaining the DSA domain

IUF supports multiple-MI software configurations in a multi-system environment.

3.2.2 The CONFIG Utility

The CONFIG utility allows the System Administrator to modify the default options, limits and thresholds delivered as standard with GCOS 7. It can be used to define/redefine such variables as:

- the maximum number of files that can be active at any one time
- the amount of memory to be dedicated to each of the major operating system dimensions
- the number of jobs that can run simultaneously.

If CONFIG is run more than once on the same system disk, the changes specified in the configuration statements are made only with respect to the original volume as delivered. Thus, if CONFIG is run a second time with only one specified change, all the changes made in the previous run of CONFIG will be cancelled and only the one latest change will be effected. This feature ensures that the status of the original system volume remains a constant reference point.

Input to CONFIG

The main items of data which CONFIG takes as input are:

- the System Resource and Status Table, which is on the system disk and defines the status of the various devices available
- the SYS.SYSTEM file on the system disk, which contains the operating system tables and the system device tables used at Initial System Load (ISL)
- a file containing a set of configuration statements which specify the characteristics of the user's configuration.

Output from CONFIG

The results produced by CONFIG comprise:

- updated system volumes
- an execution report, which indicates whether CONFIG terminated successfully or unsuccessfully, and gives a list of commands executed and parameters changed.

3.2.3 The TAILOR Utility

The function of TAILOR is to build the set of disks necessary for an operational system using, as input, either a P-Set or an R-Set.

TAILOR changes the characteristics of certain system files, defines the disks which are to support them, and suppresses the creation of some optional system files where these are not required.

Command Language

TAILOR uses a processor with a command language which allows the five system file domains to be treated independently of each other. A set of commands, help texts, and menus is available for each domain.

TAILOR commands belong to two categories:

- session commands, which determine the actions to be taken during a TAILOR session (a sequence of TAILOR commands)
- definition commands, which define the characteristics of the disks and files which make up the P-Set.

Each set of domain-related definition commands is accessed via the session command TLRGEN.

Not all the files in each domain are modifiable; some mandatory files have no selectable features.

Default Parameters

The parameters of the file definition commands supply the values which TAILOR uses to create or modify the system files.

All the command parameters have default values. If a definition command is not given for a mandatory file, the parameter default values will be applied when that file is created. The same rule applies for optional files. A file definition command is only necessary if any of the parameter default values do not match the user's requirements.

After a TLRGEN session, a set of values is obtained for each domain by merging default and user values. This set of values is stored in a permanent file and can be used as the default values for the next TLRGEN session on the same domain.

3.3 System Files

Post V5 releases support the coexistence of FBO and VBO. System files created on site may be FBO or VBO. However, system files that are delivered already loaded **must** be allocated on FBO volumes.

The following tables summarize the situation for each domain.

NOTE:
1= "DPS 7000/xx0" represents the models 2x0/3x0/4xx/5x0/7x0/8x0.

Table 3-1. FIRMWARE Domain

F = FBO
F/V = FBO or VBO
MD = Can be allocated on Mirror Disks

1 = DPS 7000/xx0
2 = DPS 7000/An
3 = DPS 7/1xx7

File Name	MD	Volume Organization	DPS 7/7000 Model
SYS.BLI		F	3
SYS.FW.*		F	1, 2, 3
SYS.FW.*.*		F	2
SYS.FW.NFTLIB		F	1, 2
SYS.FW.*.VSNi		F	2
SYS.HUB		F	1, 3
SYS.HUBG		F	1, 2, 3

Table 3-2. GSF Domain

F = FBO
F/V = FBO or VBO
MD = Can be allocated on Mirror Disks

1 = DPS 7000/xx0
2 = DPS 7000/An
3 = DPS 7/1xx7

File Name	MD	Volume Organization	DPS 7/7000 Model
SYS.GSF.BINLIB		F	1, 2, 3
SYS.GSF.SLLIB		F	1, 2, 3
SYS.GSF.SMLIB		F	1, 2, 3
SYS.GSF.UFAS		F	1, 2, 3

Table 3-3. OLTD Domain

F = FBO
F/V = FBO or VBO
MD = Can be allocated on Mirror Disks

1 = DPS 7000/xx0
2 = DPS 7000/An
3 = DPS 7/1xx7

File Name	MD	Volume Organization	DPS 7/7000 Model
SYS.HBINLIB2		F	1, 2, 3
SYS.HLMLIB2		F	1, 2, 3
SYS.HRELLIB2		F	1, 2, 3
SYS.HSLLIB2		F	1, 2, 3
SYS.SYSTEM2		F	1, 2, 3

Table 3-4. DSA Domain

F = FBO
F/V = FBO or VBO
MD = Can be allocated on Mirror Disks

1 = DPS 7000/xx0
2 = DPS 7000/An
3 = DPS 7/1xx7

File Name	MD	Volume Organization	DPS 7/7000 Model
SYS.DSABLIB		F	1, 2, 3
SYS.DSACONF		F	1, 2, 3
SYS.DSACORE		F	1, 2, 3
SYS.DSADUMP		F	1, 2, 3
SYS.DSALMLIB		F	1, 2, 3
SYS.DSALOGi		F/V	1, 2, 3
SYS.DSASLIB		F	1, 2, 3

Table 3-5. GCOS Domain

F = FBO F/V = FBO or VBO MD = Can be allocated on Mirror Disks			1 = DPS 7000/xx0 2 = DPS 7000/An 3 = DPS 7/1xx7
File Name	**MD**	**Volume Organization**	**DPS 7/7000 Model**
SITE.CATALOG		F/V	1, 2, 3
SITE.CMS_*	YES	F	1, 2, 3
SITE.HALOCK		F	1, 2, 3
SITE.HELP		F/V	1, 2, 3
SITE.IN		F/V	1, 2, 3
SITE.MIRLOG		F	1, 2, 3
SITE.STARTUP		F/V	1, 2, 3
SYS.BKST		F	1, 2, 3
SYS.BKSTi		F/V	1, 2, 3
SYS.BOOT		F	1, 2, 3
SYS.C.INCLUDE		F	1, 2, 3
SYS.CLC		F/V	1, 2, 3
SYS.CATALOG		F	1, 2, 3
SYS.ERLOG		F/V	1, 2, 3
SYS.GPL.MACLIB		F	1, 2, 3
SYS.HBINLIB		F	1, 2, 3
SYS.HCULIB		F	1, 2, 3
SYS.HELP		F	1, 2, 3
SYS.HLMLIB		F	1, 2, 3
SYS.HSLLIB		F	1, 2, 3
SYS.IN		F/V	1, 2, 3
SYS.IUF		F	1, 2, 3
SYS.JADIR	YES	F/V	1, 2, 3
SYS.JAS*	YES	F/V	1, 2, 3
SYS.JRNAL	YES	F/V	1, 2, 3,
SYS.KNODET		F/V	1, 2, 3
SYS.LIB[i]		F/V	1, 2, 3
SYS.LOGC		F/V	1, 2, 3
SYS.OUT		F/V	1, 2, 3
SYS.PVMF[i]		F/V	1, 2, 3
SYS.QM*		F/V	1, 2, 3
SYS.SITE.*		F	1, 2, 3
SYS.SPDUMP	YES	F/V	1, 2, 3
SYS.SPOOLi	YES	F/V	1, 2, 3
SYS.SWLOG	YES	F/V	1, 2, 3
SYS.SYSDUMP		F/V	1, 2, 3
SYS.SYSTEM		F/V	1, 2, 3
SYS.TVMF[i]		F/V	1, 2, 3
SYS.URCINIT		F	1, 2, 3

3.4 Multi-System Installation

The multi-system capability enables the System Administrator to install and maintain several system disks, each belonging to a different DPS 7000. Each machine and its operating system can have a different hardware and software configuration, but all the systems must be installed and maintained with the same release of GCOS 7.

One system is the master system and the other systems are slave systems. A master disk set is created on the master system and this is used to create and update disk sets for all the slave systems.

Both the master and the slave systems have fixed system disks. They do not necessarily need to be on the same site because a tape or cartridge image of a master disk set can be transported to the slave system to create the slave R-Set or P-Set.

4. Communications: Generation and Support

This chapter describes the components which implement DPS 7000 communications support, and which allow the user at a terminal to connect to any application and to use any resource in a system connected to the DPS 7000 host.

Communications support is described in terms of:

- the hardware available on the various models of the DPS 7000
- the firmware supporting the different protocols of network transmission
- the software which the user is supplied with and can develop for specific data processing requirements.

4.1 Communications Environment

GCOS 7 allows an application running on one DPS 7000 system to exchange data and control information with applications running on another system, which may or may not be of the same type. In a similar manner, terminals connected to one system can converse with another system or with terminals connected to another system.

Moreover, such intercommunication is not limited to two systems; there may be (and in fact usually is) a network of systems and terminals linked together in a pattern which is sometimes quite complex.

A system which forms part of such a network makes use of a number of internal tables which record connections, traffic patterns, protocols, statuses, and other information to make such communication possible. For GCOS 7, many of these tables are loaded and/or tailored by the Network Generation utility, NETGEN. In effect, NETGEN creates the GCOS 7 system's local view of the network.

4.1.1 Primary and Secondary Networks

A network may be primary or secondary. A primary network consists of a number of computer systems (called nodes in this context) connected together. A secondary network consists of a number of terminals connected to a node.

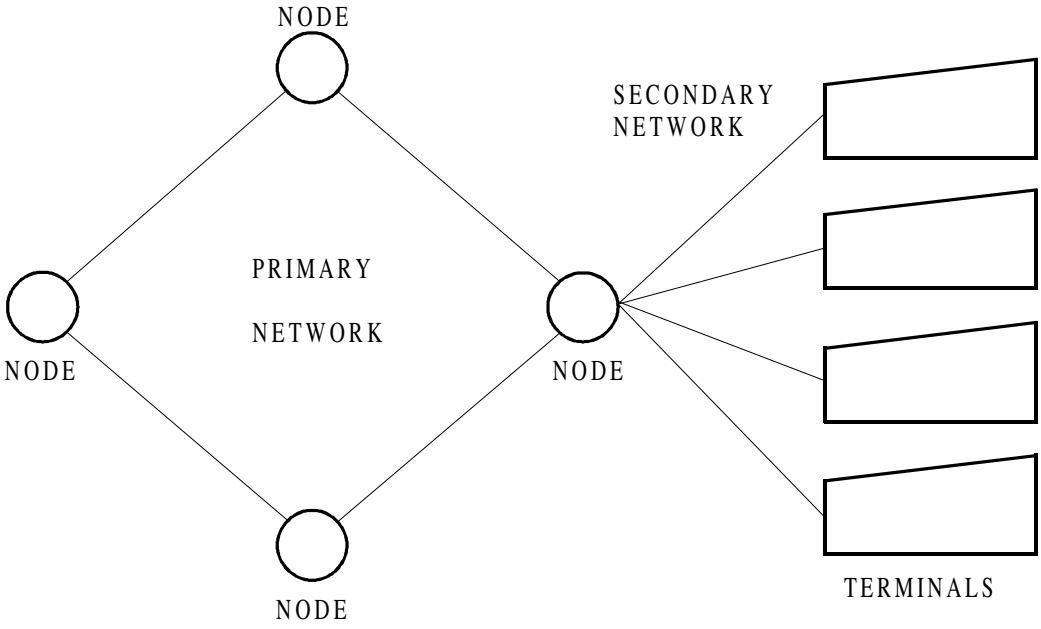

Figure 4-1. Primary and Secondary Networks

4.1.2 Front-End Processors

The communications environment always requires a front-end processor to:

- manage terminals of the local and the TRANSPAC secondary networks
- access remote systems (nodes) of the primary network

Since terminals are managed by the front-end processor, they are declared in the system generation of the front-end processor.

There are two types of front-end processor, the CNP 7 and the Datanet. Table 4-1 below summarizes which type is connectable to which DPS 7000 model.

Table 4-1. DPS 7000 Models and Front-End Processors

DPS 7000/4x5	CNP7 + Datanet + MainWay
DPS 7000/Dxx	CNP7 + Datanet + MainWay
DPS 7000/Cxx	CNP7 + Datanet + MainWay
DPS 7000/Mxx	CNP7 + Datanet + MainWay
DPS 7000/MTxx	CNP7 + Datanet + MainWay
DPS 7000/8x0	Datanet + FCP7

The communications servers used by GCOS 7 to manage the communications interface with the front-end processor are:

- TNS, for handling all communications through the ISL (Inter System Link). Communications are either through the CNP7s or through direct DPS 7000-to-DPS 7000 links.
- FEPS manages the dialog with the Datanet. There is one FEPS occurrence for each Datanet configured.
- OCS manages the dialog with the FCP7. There is one OCS occurrence for each FCP7 configured.

FPG7, a GCOS 7 interactive menu-driven application, is available to help the user declare terminals and nodes in the system generation of the front-end processors.

If the network requirements of the installation are not complicated, the average user can handle the communications setup without the assistance of the Service Center.

Some network attributes can be dynamically modified using the GCL command MDNET.

4.1.3 GCOS 7 Networking Topology

The DPS 7000 connects to the network through its PSI channel, linking:

- either a Datanet as its front-end processor
- or a controller accessing the ISL by means of a physical plug.

The controller is the SPA or LNM for a DPS 7000. The SPA (Service Processor Attachment), LNM (Local Network Manager), provide the link between the DPS 7000 and the ISL. The ISL in turn links up with another DPS 7000 either directly or via a CNP 7 or Datanet.

Figures 4-2 to 4-3 show typical examples of network configuration.

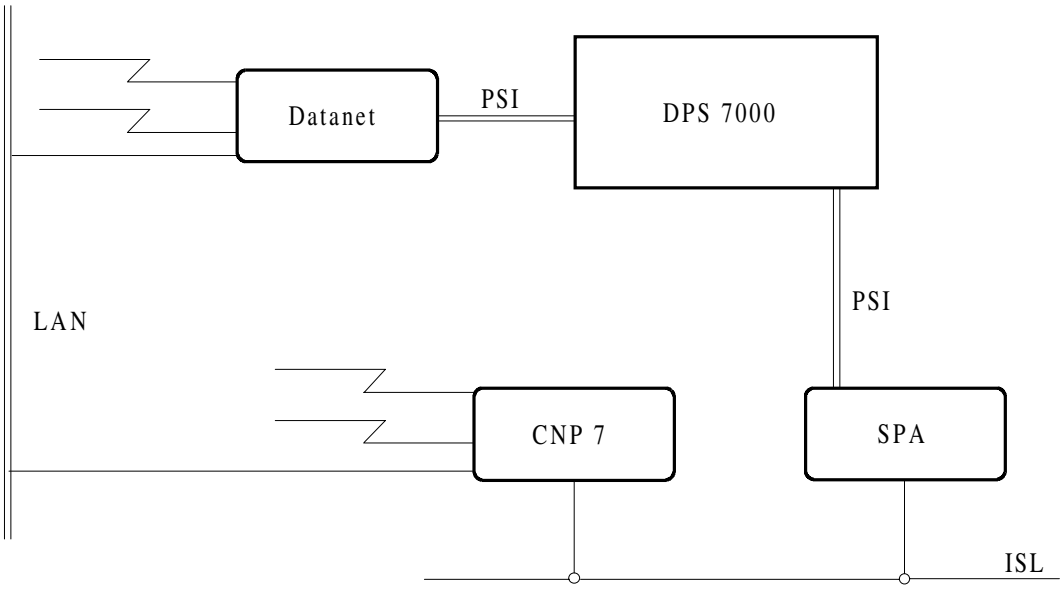

Figure 4-2. DPS 7000 with SPA and CNP 7

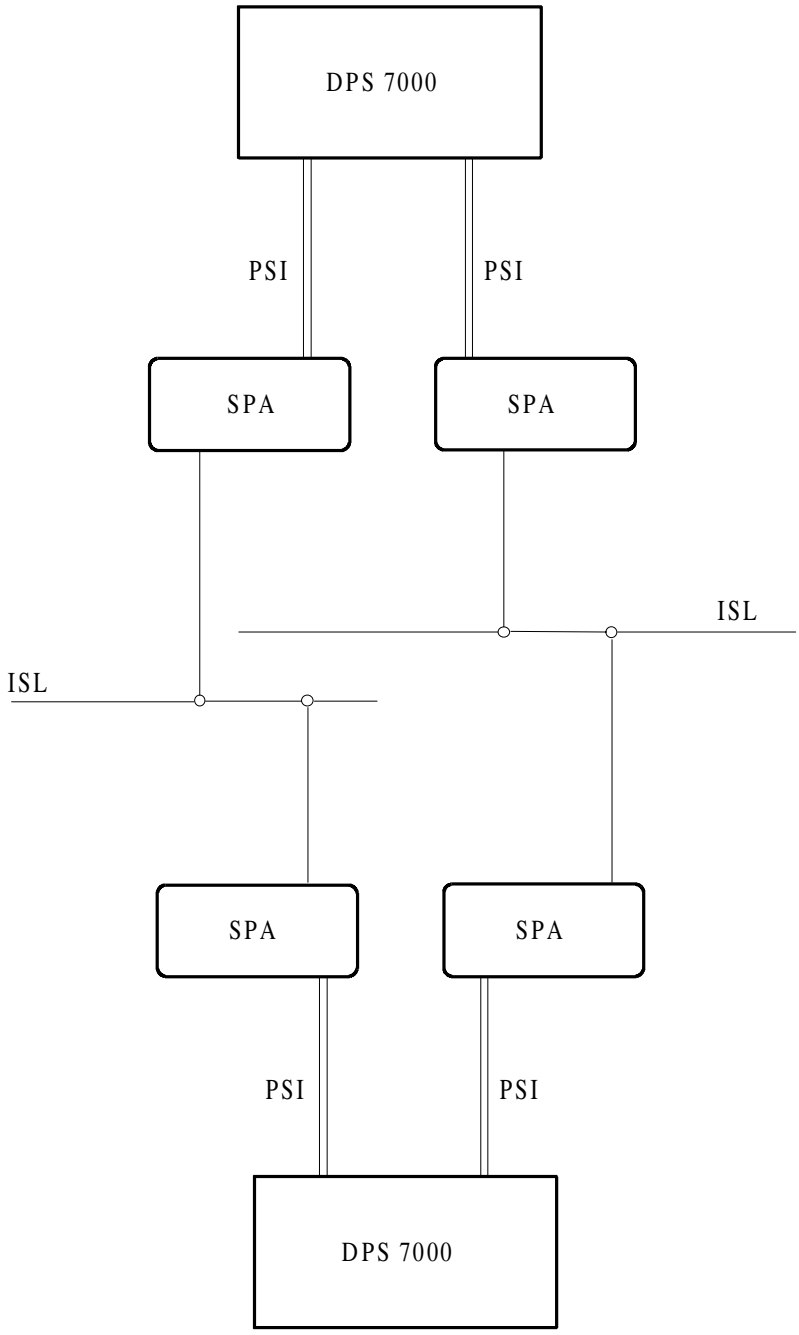

Figure 4-3. DPS 7000-to-DPS 7000

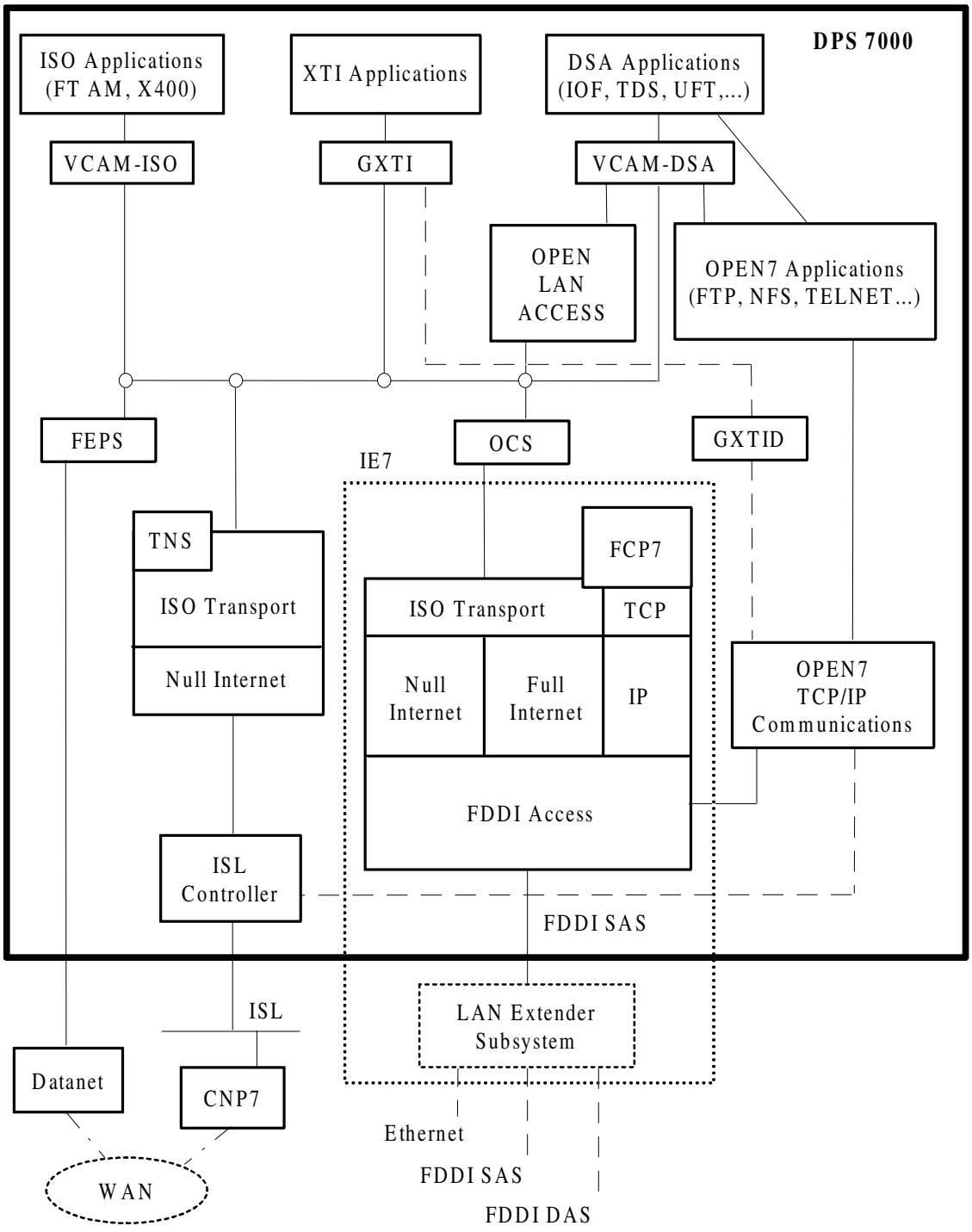

Figure 4-4. DPS 7000 Communications Architecture

4.2 Concepts of a Layered Architecture

DSA defines a layered architecture which complies with the OSI reference model of ISO.

This means that the concept of a layered architecture and the role of each layer are common to DSA and ISO.

Depending on each layer, GCOS 7 implements either the ISO standard for the layer or a proprietary DSA standard.

According to the DSA/ISO reference model, each layer within the same system provides services to and receives services from the layer immediately above or below it. Peer layers of different systems cooperate by using the same protocol.

The reference model contains seven layers.

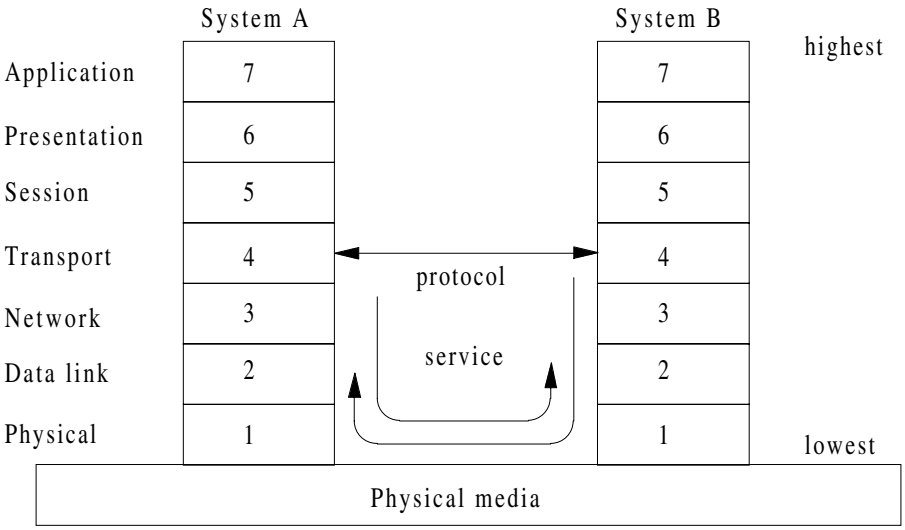

Figure 4-5. DSA/ISO Layered Architecture

4.2.1 Application Layer

The application layer provides services which are directly available to the user. Distributed processing involves the interactive use of system and network resources through DSA applications such as IOF, TDS, UFT, and DJP, or ISO applications like DMS7.

4.2.2 Presentation Layer

The presentation layer is concerned with the conversions to be applied to the contents of data exchanged between applications. For instance, ASCII-to-EBCDIC conversions pertain to the Presentation layer.

Presentation functions are often packaged with applications. Products such as FORMS and GKS ensure presentation functions by handling information displayed on screen terminals.

4.2.3 Session Layer

The session layer organizes and synchronizes the dialog between two applications. It establishes connections, manages the flow of data exchanged, and controls the turn (token) among session endpoints.

Where cooperating applications reside in different systems, the session layer uses the transport services to exchange messages between the systems.

Two applications residing in the same system can also establish a session. Such a session is "local" since it does not involve the network. The local session therefore takes direct charge of messages passing between the applications.

The software component which provides the message management in the session layer is VCAM (Virtual Communications Access Method) which is a set of sharable session control procedures.

VCAM-DSA allows DSA applications running on a DPS 7000 to communicate with those running on other Bull or non-Bull systems. VCAM-DSA handles all user requirements within a DPS 7000 environment of native systems.

VCAM-ISO allows the same for ISO applications, plus management and resynchronization of activities.

4.2.4 Transport Layer

The transport layer ensures a reliable means of transferring data between session entities of different systems by ensuring sequence control, error detection and recovery, and flow control.

Depending on the network topology, a session connection may correspond to a single end-to-end transport connection or multiple transport connections relayed in a sequence.

Communications via the ISL:

The communications management component which controls transport (and network) functions is TNS. TNS handles communications with the CNP 7 connected to the SPA or LNM via the ISL. TNS also handles communications with the MPC.

Communications via the Datanet:

FEPS only provides a pseudo-transport function since the Datanet handles all effective communications. On the DPS 7000 side, FEPS manages the Communications Controller of the DPS 7000 linked to the Datanet by the PSI. On the Datanet side, DNS establishes the transport connection between its local and the remote transport stations.

4.2.5 Network Layer

The network layer is needed whenever a transport connection uses communications link(s) established through intermediate system(s). It provides routing and relaying services necessary for maintaining such links.

4.2.6 Data Link Layer

The data link layer provides a point-to-point transfer of individual frames of data over a physical connection.

4.2.7 Physical Layer

The physical layer provides a mechanical, electrical, functional and procedural means to transmit bits.

The hardware components which provide the interfaces for the data link and physical layers are identified by CCITT and IEEE norms.

4.2.8 Networking Interfaces

The administration of the local system and its front-end processor is based on DSA objects and addressing, which remain mandatory even with ISO. The SCID (Session Control Identifier) specifies the address of the local session control and must be declared for generating the ISO network. This DSA/ISO interdependence is illustrated by the following two diagrams.

Figure 4-7 shows a DPS 6 connected to a DPS 7000 host through its CNP 7 to explain the relationship of DSA to ISO.

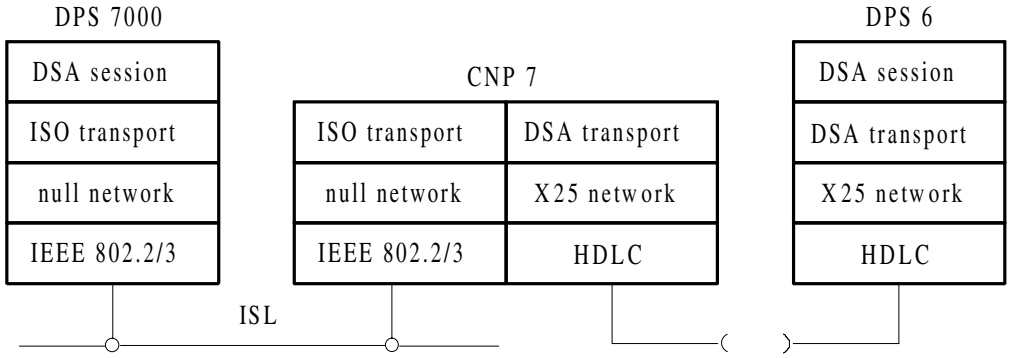

Figure 4-6. Relationship of DSA to ISO

Figure 4-8 shows a DPX/2 machine implementing UNIX communicating with the DPS 7000 using only ISO standards.

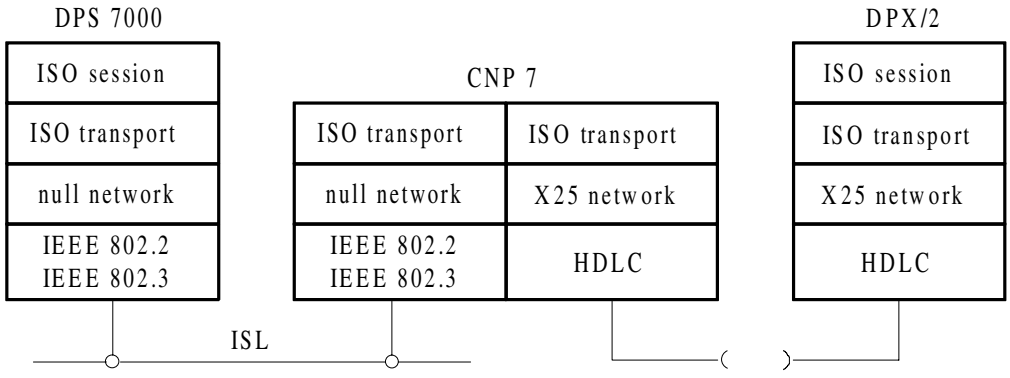

Figure 4-7. Direct Connection to UNIX Machines

4.3 How GCOS 7 Implements the DSA Layers

GCOS 7 network applications such as TDS, IOF, DJP, and UFT interface with VCAM. VCAM-DSA implements DSA200 protocol. The common VCAM interface ensures that the applications are independent of all subsequent subsystems implementing the lower DSA layers.

Below VCAM, DSA layers are distributed differently according to the type of connection.

Datanet

For a Datanet connection, the transport layer is spread between FEPS and the Datanet. The Datanet handles the real transport while FEPS handles the pseudo-transport interface specific to the DPS 7000. The pseudo-transport is limited to direct exchanges with the Datanet.

The real transport and all layers below it are implemented in the Datanet. This means that the Datanet (and not the DPS 7000) is responsible for implementing the transport and network protocols when a communication is established with a remote system.

CNP 7 and Connection via the ISL

For CNP 7 or ISL connections, the TNS communications server handles the transport and network protocols. For this reason, all the NETGEN directives defining the communications path between the DPS 7000 and its CNP 7 must be declared.

The SPA, LNM, and MPC implement the data link and physical protocols.

Direct communications established through the ISL are restricted to qualified links between DPS 7000s, namely:

- file transfers
- cooperative transaction processing using XCP1 protocol
- passthrough.

FCP7 Communications

This is new with GCOS 7-V7.

The FCP7 communications controller provides access to FDDI networks. It supports the ISO and TCP protocols and is magnaged by an OCS server.

For more information, refer to:

Network Overview, 47 A2 92UC

Network Generation, 47 A2 93UC

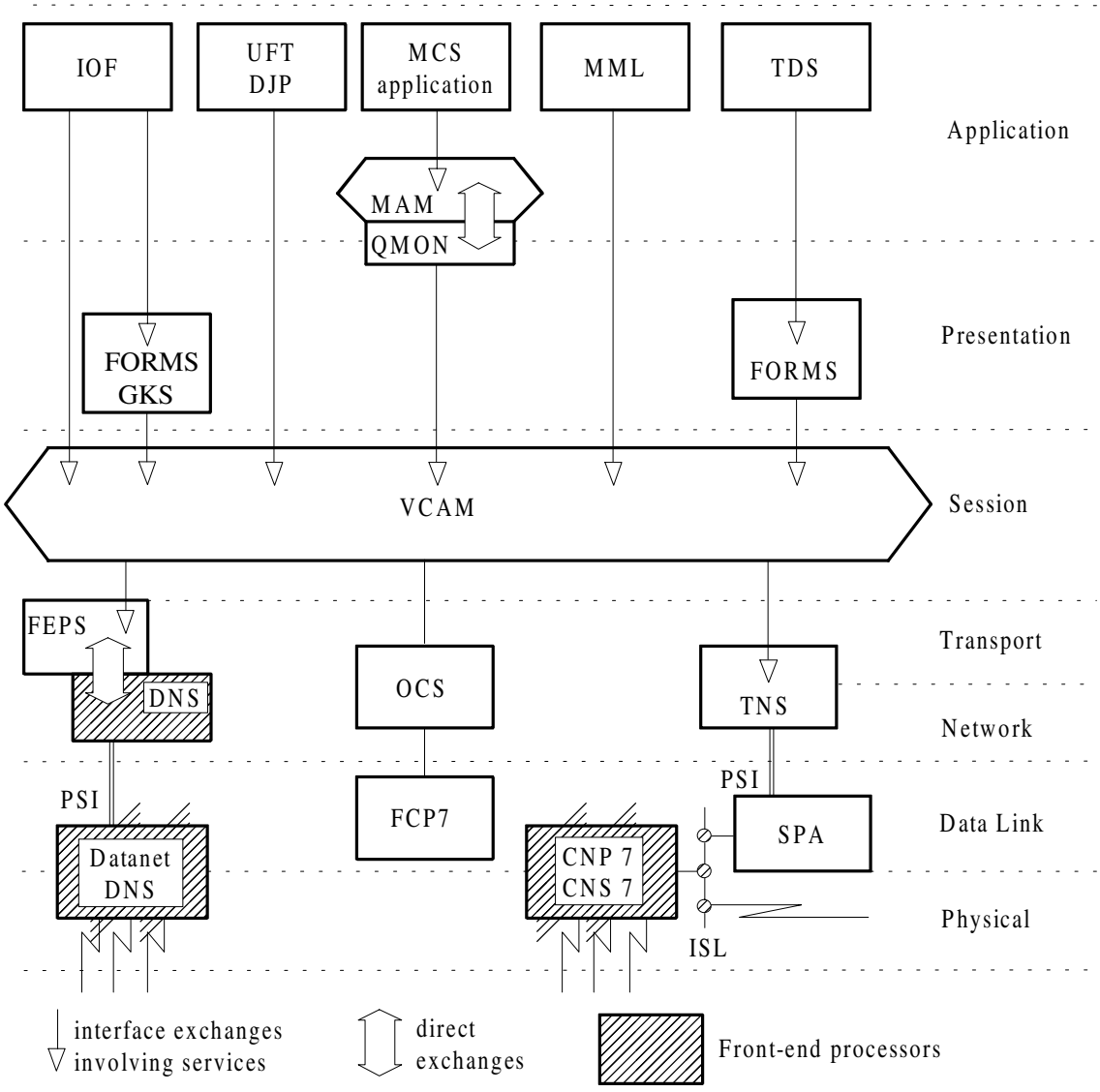

Figure 4-8. Implementation of DSA Layers

4.4 How GCOS 7 Implements the ISO Layers

VCAM implements the ISO session services and protocol according to the ISO 8326 and ISO 8327 standards.

VCAM-ISO supplies the programmatic interface and the configuration which are needed to manage the ISO session connection between a GCOS 7 application and a remote system.

ISO addressing

Each layer is accessed by the use of SAPs (Service Access Points), that is, SSAPs for the session, TSAPs for the transport, and NSAPs for the network.

The System Administrator must define:

- the ISO address of the local system
- the default values at configuration time (LSYS directive)
- the TSEL (transport selector)
- the NSAPs (a list of up to 20 Network Service Access Points)

The ISO application defines the SSEL (session selector).

The full ISO address is thus defined by the SSEL, the TSEL, and the NSAPs.

Datanet

The real ISO transport and all layers below it are implemented in the Datanet. This means that the Datanet (and not the DPS 7000) is responsible for implementing the transport and network protocols when a communication is established with a remote system.

CNP 7 and connection via the ISL

TNS implements the ISO class 4 transport protocol (ISO_DIS 8073) and a basic network layer referred to as the null subset of Internet (ISO_DIS 8473).

In order to make the connection, CNS 7 requires the full ISO address (TSEL and NSAPs). The CNP 7 chooses one among the given NSAPs, depending on the network it uses, to reach the remote system (outgoing connection).

The transport protocol is not a standard one and the connection between TNS and CNS 7 is seen as a channel connection to transfer a session-to-transport interface from the host to the front-end processor. This use of of the transport protocol is declared at generation time by the ISO:2 parameter of the RTS primitive of NETGEN.

4.5 How GCOS 7 Implements TCP/IP

A GCOS 7 subsystem called OPEN7 provides the interface between the DPS 7000 and all TCP/IP facilities.

OPEN7 provides access to:

- a DPS 7000 from a system using TCP/IP via an Ethernet network connected through the ISL.
- a DPS 7000 from a system using TCP/IP via an X25 network connected to a DPS 7000 front-end processor (CNS, DNS).
- GCOS 7 ORACLE with SQL*FORM and asynchronous terminals on some DPS 7000 models.

OPEN7 also allows:

- logon through IOF or TDS, for presentation of the local UNIX system in line or FORMS mode.
- connection to OPEN7 on a local UNIX system for presentation in character mode.
- use of X protocol (Xform) with IOF or TDS, which provides Xwindow/motif presentation of the local UNIX system, and line or forms mode for IOF or TDS.
- data transfer through:
  - IOF initiated file transfers between the remote UNIX system and GCOS 7 source library members or UFAS sequential files. This is useful for transferring data between GCOS 7 applications and UNIX systems.
  - OPEN7 initiated file transfers between the remote UNIX system and an OPEN7 file. This is useful to the System Administrator who might want to copy terminal-description files (for example) into OPEN7 of a non-UNIX system.
  - file transfers between GCOS 7/OPEN7 and a UNIX system. This is useful for saving UNIX files on DPS 7000 systems. As OPEN7 files are GCOS 7 files (with NONE organisation), they may be saved on UNIX volumes using the GCOS 7 file utilities FILSAVE and FILREST.
- file access:
  - to GCOS 7 UFAS files (FBO) from a remote UNIX system
  - to OPEN7 files which allows the System Administrator to build a file on DPS 7000 using a UNIX workstation, and afterwards use OPEN7 commands to transfer GCOS 7/OPEN7 files to a GCOS 7 source library member or UFAS sequential file.
  - to SQL*NET between two ORACLE databases

4.6 Communications Servers

The main communications functions are servers which are implemented as subsystems. Each server executes as a GCOS 7 service job, and is started and terminated by the network control operator.

4.6.1 TNS

Transport and Network Subsystem is a unique occurrence and handles all communications through the ISL. Communications either are through the CNP7s or are direct DPS 7000-to-DPS 7000 links (in the case of the MPC).

4.6.2 FEPS

Front-End Processor Support manages the dialog with the Datanet. Since each FEPS occurrence handles a single Datanet configured, FEPS is a server of multiple occurrences.

4.6.3 OCS

OCS manages the dialog with the FCP7. Since each OCS occurrence handles a single FCP7 configured, OCS is a server of multiple occurrences.

4.6.4 FECM

Front-End Control and Management administers all Datanets and CNP 7s according to the scenario defined for their functions, namely, system generation, and loading and dumping their software. Each front-end processor has its own standard FECM scenario which is factory-supplied.

4.6.5 RAEH

Remote Administrative Exchange Handler is a unique occurrence for handling all DSAC (Distributed Systems Administration and Control) sessions between the DPS 7000 host and its configured systems. The host can send commands to and receive responses from each correspondent declared. Notification of events occurring throughout the network are logged by the host.

4.6.6 QMON

Queue Monitor is a unique occurrence for managing all queues accessed by MCS applications which are user-defined communications programs. QMON cooperates with MAM to provide the interface between MCS applications and VCAM.

4.7 Network Generation

The network must first be configured before any communications session can be established. Each participating system of the network performs its own individual network generation. Each system recognizes all other systems with which it wants to communicate as its correspondents; its network generation therefore is its own particular "local" view of the network.

The most basic network consists of a DPS 7000 and its front-end processor. When a DPS 7000 system is first installed, a standard network generation for the host DPS 7000 and matching standard FECM scenario(s) for its front-end processor(s) are delivered with the GCOS 7 release.

The standard network generation represents the NETGEN version describing the minimum system resources of a DPS 7000 installation. These resources are the DPS 7000 host and its front-end processors. However, if the installation operates in the OPEN7 environment and using TCP/IP protocols, only the local system need be declared.

If the user is not concerned with remote systems of the primary network, the standard network generation is enough to operate the communications session.

NETGEN is a GCL procedure which starts the DPS 7000 network generation batch processor:

- either for modifying or adding to the standard DPS 7000 network generation delivered with GCOS 7
- or for generating a personalized network, not using the standard network generation.

4.8 System Generation For The Front-end Processor

The FPG7 utility is a simple means of configuring terminals of the secondary network, or primary network of the front-end. This utility is an interactive menu-driven application which makes the CNS7 or DNS network directives transparent to the user. All that the user has to do is to key in values for user-defined entries in each successive menu. When each menu is processed, the equivalent network directive or set of directives (CNS7 or DNS) is generated.

FPG7 is also used to create the corresponding NETGEN directives.

4.9 Network Administration Configuration

The Remote Administrative Exchange Handler (RAEH) administers the first 5 layers of DSA architecture: physical, link, network, transport, and session. It is based on the concepts and protocols common to all DSA systems, namely DPS6, DPS 7000, DPS8, and Datanet/CNP 7.

The default network administration configuration is a set of network administration directives with values typical of most DPS 7000 networks. These network directives take default NETGEN values and are invoked when the network generation is loaded.

The user can redefine any of these directives with values specific to the needs and resources of a reconfigured network.

4.10 Applications and programmatic interfaces

Applications comprise those which are delivered as packaged products and those which the user develops using standard languages and the communications programmatic interface.

4.10.1 IOF

The Interactive Operator Facility provides the user with such utilities as:

- creating, updating, and deleting source files or members containing data, language instructions, processor commands, or GCL statements
- starting and controlling batch jobs, MCS applications, and other communications services such as TDS, from any general-purpose terminal
- scanning and receiving output reports from jobs submitted
- executing interactively any application, utility, or processor.

4.10.2 TDS

Transaction Driven Subsystem is an executive for initiating, scheduling, and monitoring transactions executable in the form of sharable TPRs (Transaction Processing Routines). The TPRs are written in either COBOL or GPL.

The executive is generated by two utilities, namely:

- TP7PREP for preparing the on-line and off-line files needed for the generation
- TP7GEN for executing TDSGEN to allow the user to set parameters to the requirements of the application.

A generated version of TDS with its associated TPRs is started as a job step which:

- provides the interface for connections and disconnections, and data exchanges between the TPRs and VCAM
- dispatches messages across the TPRs according to the contents and context of the message.
- uses the Before and After Journals for data security
- uses GAC to manage concurrent file accessing
- implements the restarts after an abort or system crash.

TDS is dynamically managed by a set of TDS Master commands.

4.10.3 MICROFIT 7

This is a Micro-to-Mainframe link which enables:

- the transfer of files between the DPS 7000 and a microcomputer
- the user at the microcomputer to connect to target applications in the DPS 7000.

The microcomputer can be:

- either a Bull MICRAL or PC-like device functioning in one of two modes, namely:
  - asynchronous or synchronous mode by emulating a PC7800 model with PC7800 or Mimic software
  - synchronous mode with ATLANTIS products (SCOM, UCOM, and XCOM boards)
- or a Bull QUESTAR 400 functioning in synchronous mode by emulating a DKU7007.

4.10.4 UFT

Unified File Transfer enables the DPS 7000 to transfer a file in interactive or batch mode from or to another DPS 7000, DPS6, DPS8, QUESTAR 400, DPX 2030/5000 (SPS 7/9), DPX 2020 (QUESTAR 700), or an IBM site.

4.10.5 DJP

Distributed Job Processing enables the user at the DPS 7000 site to:

- submit a job for execution at a remote DPS 7000 site
- execute a job submitted by a remote DPS 7000 or a DPS6 functioning as a departmentalized station.

DJP, like UFT, uses the RFA (Remote File Access) protocol which conforms to the DSA standard.

4.10.6 FORMS

The MAINTAIN_FORM (MNFORM) utility is used for creating a form either interactively or in batch for the DKU7005/7007/7107/7211, IBM3270/3278/3279, MINITEL, PC7800, and VIP7804 through the SDPI interface.

A form represents a portion of the screen composed of fixed and variable text. The definition of the fixed text and the location of the variable text are defined in the form and not by the user application using the form. The presentation of data is therefore independent of the application.

Access to the form created by MNFORM is provided in COBOL under IOF and TDS.

4.10.7 GTWRITER

Generalized Terminal Writer enables a SYS.OUT report to be printed on any designated hard-copy terminal in the network.

It can be used under IOF, TDS, or in batch. The principal advantage is that a transaction can be initiated at one terminal and the results of the transaction can be received at another terminal.

4.10.8 DOF 7-PO

Programmed Operator Support enables a large number of Main Operator activities to be automated. Messages sent to the console are analyzed and responses automatically sent back.

The data interface for DOF 7-PO primitives is the Structured Record which delivers the type of command and the response required.

4.10.9 MCS

Message Control System provides access to user-defined queues. Its functions are provided by:

- MAM, which is a set of sharable queue control procedures
- QMON, which is the communications server interfacing VCAM session control procedures with MAM queue control procedures.

MCS applications are written in either COBOL or GPL and use queues for the input and output of data.

The functions provided by MCS are:

- managing queues of messages in memory and on disk files
- recovering disk queues after an incident
- synchronizing and maintaining communication between MCS applications within the same system.

4.10.10 VCAM

The Virtual Communications Access Method is the GCOS 7 programmatic interface through which applications (correspondents) can exchange data and information. Protocol and session services are provided by VCAM through a set of system primitives, which perform such tasks as:

- connection administration by defining the objects required for establishing the session
- session handling when opening and accepting, and closing the session
- dialog control when normal data, requests, and control information are sent from and received by the correspondents
- error management for the recovery of information.

4.10.11 XCP1

Extended Communications Protocol Level 1 provides the communication between different TDS applications or between TDS and IMS. If the communication is TDS-to-TDS, the session established between the user and the principal system can be extended to other systems, in a relay.

The XCP1 protocol allows the user to recover messages and to avoid the duplication of messages where retransmission occurs on recovery. The interface is implemented by a set of COBOL procedures.

Message recovery enables the transaction to re-establish the communication.

4.10.12 XCP2

Extended Communications Protocol Level 2 (Program-to-Program Communication) enables TDS transactions to cooperate in one or several DPS 7000 systems.

Cooperating transactions are TDS transactions and CICS applications. CICS applications in an SNA network are those which use IBM LU6.2 protocol interconnected through OSF to the DSA network.

CPI-C/XCP2 enables a transaction to:

- start and converse with another transaction in any system
- manage session pools and correspondents

The CONFIRMATION service of XCP2 enables the transaction to re-establish the communication.

4.10.13 AUPI

Administrative Utilities Programmatic Interface allows the user to process administrative data transmitted by systems which support DSA200 or DSA300 (AEP2) protocols.

AUPI enables the user to:

- obtain information on events, statistics, or results of command execution generated by systems supporting AEP1 and AEP2 protocols
- issue commands to systems conforming to the AEP2 protocol for monitoring network operations.

AUPI allows:

- the simplification of complex AEP encoding techniques used by DSA systems by providing a high-level view of this protocol
- user-defined applications or utilities to be independent of changes in the structure of AEP protocol.

AUPI conforms to applicable DSA standards. This means that utilities can be transported from one system to another, except when they are written is a language using system-specific features (like COBOL).

Figure 4-10 below shows the data flow between the NAD (Network Administrator) and the ASF (Administrative Storage Facility). The administrative objects and the types of AEP record are identical to those described for RAEH.

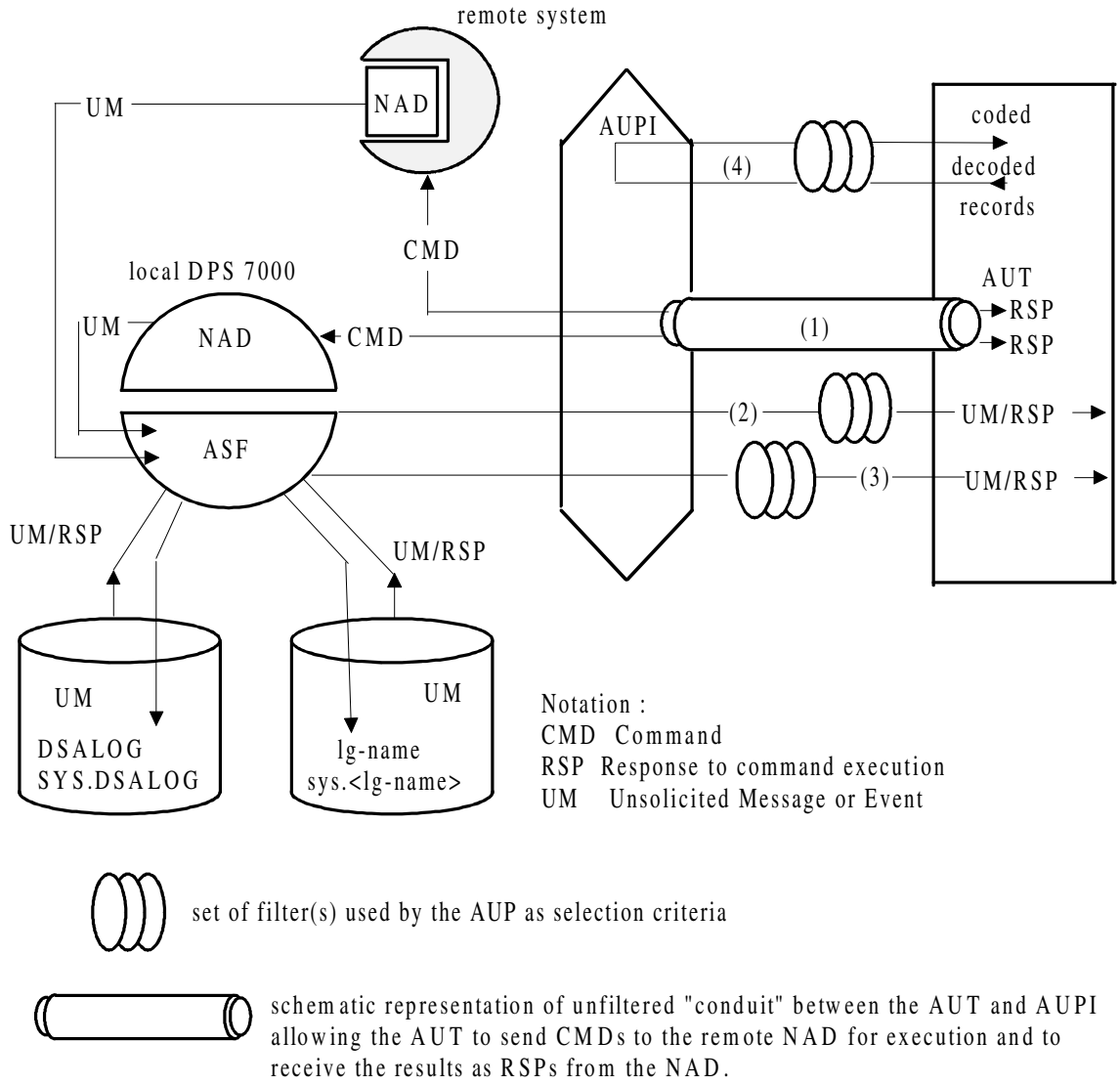

Figure 4-9. Schematic Overview of AUPI

5. System Operation

This chapter describes some of the ways in which the System Administrator can contribute to the smooth and efficient use of the system in its everyday operations. Topics include:

- recommendations on job class usage
- recommendations on file and volume naming
- a review of automated operation
- a review of coupled systems operation

5.1 Job Class Concepts and Conventions

Job classes are a means of organizing jobs into manageable groups according to their priority, profile, resource needs, and operational constraints. This simplifies job management and also leads to a more effective use of system resources.

All job classes are identified by a one-character or two-character name: A to Z or AA to ZZ. All have the following attributes:

- a scheduling priority (PRIORITY)
- an execution priority (XPRTY)
- a multi-programming limit (MAXLOAD)
- a state at system startup (STARTED or NSTARTED)

Some job classes may also have the following attributes:

- a limit to their scheduling and execution priorities (PRLIM and XPRLIM)
- a maximum elapsed time per step (ELAPTIME)
- a maximum CPU time per step (CPTIME)
- a connection to a parent job class (JCG)
- restrictions preventing operator-level modification (NSC, NMXPRTY, NMPRIO, NMLOAD)

The application programmer can override some of these job class values using:

- the PRIORITY parameter in the JCL statement $JOB
- the XPRTY, ELAPTIME and CPTIME parameters in the JCL statement $STEP or the GCL command EXEC_PG.

The system operator can:

- Use the MODIFY_LOAD command to modify the scheduling priority, execution priority, and multi-programming limit (unless defined as non-modifiable at CONFIG time).
- Use the CONNECT_LOAD and DISCONNECT_LOAD commands to connect or disconnect a job class from a job class group (JCG).

Unless specified otherwise, each job entering the system is scheduled and executed according to the scheduling priority, multiprogramming limit, and dispatching priority of its job class.

Scheduling Priority (PRIORITY)

This determines the job's position in the scheduler queue where it waits to be selected for execution. A job is placed ahead of jobs with lower priority, behind jobs with higher priority, and on a first-in-first-out basis with jobs of equal priority. Generally speaking, the higher the priority the quicker the selection process; though this would not necessarily be true if all the jobs have an equally high priority. Also, a lower priority job can be selected ahead of a higher priority job if the latter is waiting for its job class to be started or its class multiprogramming level to come down.

Dispatching Priority (XPRTY)

In simple terms, this determines a job's competitiveness for CPU time. Generally speaking, the higher the priority, the higher the ratio of CPU time to elapsed time.

Class Multiprogramming Limit (MAXLOAD)

This is the maximum number of jobs within the job class that can be executed at the same time (including suspended jobs). A job cannot start executing if the number of jobs already executing in its class is equal to the class multiprogramming limit.

5.1.1 Standard Job Classes and recommended usage

Classes A to Z, RA and RB are delivered pre-configured. By default, they are automatically initialized (started) at system startup.

A to Q	are intended for user jobs. P is the default class for Batch jobs. Q is the default class for IOF jobs.
R to Z, RA, RB	are reserved for service jobs. These are not available to user jobs.

Mono-character classes can be reconfigured with the job class attributes listed below using the CONFIG statement JOBCLASS. Classes RA and RB cannot be reconfigured.

In a multiprogramming environment like GCOS 7, it is important that the System Administrator choose an appropriate job class convention for the site and that all job submitters are aware of it. One specific means of doing this is to define a list of allowed job classes for each project. Also, major or frequent changes to job class attributes should be avoided as this makes the convention difficult to maintain and the task of the user unnecessarily difficult.

Table 5-1. Job Class Attributes and Default Values

Job Class	Default Scheduling Priority (PRIORITY)	Default Execution Priority (XPRTY)	Default Multiprog. Limit (MAXLOAD)	Recommended Usage	Comment
SERVICE JOBS					
Q	0	1	100	IOF jobs	
R	0	1	6	Readers	(1)
RA				OLTD (Job TD-MVH)	
RB				Mirror (Job REFFIL)	
S	0	0	12	Telecom (QMON, FECM, FEPS, TNS, RAEH, , OCS) IOSER ARS	(1)
T	7	2	6	Telecom (Job SERVER2)	
U	0	1	6	RBF	
V	7	9	1	System Trace (Job TRCCL)	
W	0	1	8	Output Writer (Job WRITER)	(1), (2)
X	0	2	4	JCL Translator (Job JTRA) Tds-HA Services (Jobs CMSR, RECOV, JRU)	(1), (2)
Y	7	2	6	Telecom (Jobs VCAM, GTP) Segment Server for TDS (Job JPPC)	(1), (2)
Z	0	0	4	Main Operator (Job IOF) Local Message Handler (Job LAEH)	(1)

(1) Class always stated

(2) Use IDLE state

Recommendations (see also Table 5-1):

- Classes R-Z and RA-ZZ reserved for system jobs
- Classes Q and QA to QZ for IOF jobs, with Q the default
- Classes P and AA-PZ for normal Batch jobs, with P the default
- Classes D-G for high priority (and probably unplanned) batch jobs
- Class H for user communications jobs (MAM for example)
- Class J for TDS jobs

The remaining classes (A-C, I, and K-O) are generally for Batch user jobs which have particular processing or operational requirements. For example:

- jobs which need to run as a suite
- jobs which need to run at specific times
- jobs which are intensive users of a particular resource.

If for instance the system is generally CPU-bound, it is advisable to use a lower priority class for a CPU-intensive job and a higher priority class for an I/O-intensive job. This is a useful (though static) form of system regulation.

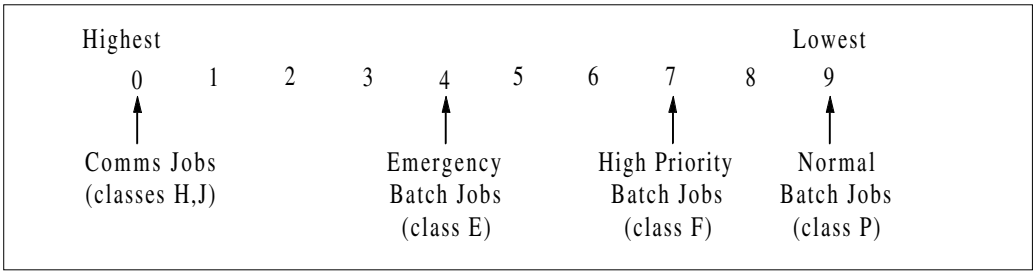

Figure 5-1. Typical Batch Job Classification

5.1.2 User-Created Job Classes

User-created job classes are two-character classes created statically by the CONFIG statement JBCLASS2, or dynamically by operator command START_LOAD.

Two-character classes (with the exception of RA and RB) can be deleted using the TERMINATE_LOAD command.

5.1.3 Job Class Group Concept

A Job Class Group is a means of organizing a number of usually similar job classes into a single manageable entity. Just as job classes are a form of job management, Job Class Groups are a form of job class management. They are generally used in the context of dimensions and automatic resource management (ARM). This is discussed further in Chapter 6.

A Job Class Group can be created via the Main Operator command CONNECT_LOAD (CNLD). Alternatively, a Job Class Group can be configured at CONFIG time using the JOBCLASS statement (mono-character) or JBCLASS2 statement (two-character).

The connection of a class to a Job Class Group can be changed using the Main Operator commands DISCONNECT_LOAD (DISLD), and CONNECT_LOAD (CNLD).

A Job Class Group consists of a 'parent' job class and one or several 'dependent' job classes. The group retains the name and attributes of the parent job class. Keep in mind that no job class group can be both a parent job class and dependent job class at the same time. If a job class group has dependents, then it cannot have a parent.

Not all job classes necessarily belong to a Job Class Group, so there may be JCGs, dependent JCs, and independent JCs.

The multiprogramming limit of the group is always equal to the multiprogramming limit of its 'parent' job class. Thus, if the total number of jobs currently executing in the group is equal to the MPL limit of the 'parent' class, no other jobs can be started anywhere in the group.

5.2 File and Volume Administration

5.2.1 Naming Conventions

The directory and catalog facilities are designed to allow users to name their files without undue concern for the uniqueness of these names. This flexibility is of course essential in a multi-user environment. However, if the System Administrator does not implement certain naming conventions, problems can arise.

For example, if a user specifies a file name which is not the full path name, and the file concerned is not in the working directory, the system will look for this file according to its own search rules. Clearly, if there is more than one file with the same name, the system may come across the wrong file first.

To avoid all possible errors of this nature and to make file and volume naming easy, the System Administrator should take the following precautions:

- Give unique names to project catalogs so that they can all be cataloged in the Site Catalog.
- Make all project catalogs auto-attachable so that one cannot forget to attach them before modifying project and file access rights.
- Give unique names to each master directory in the same catalog. Better still, create only one master directory per catalog and give this the same name as the project. With this convention, all files names belonging to a project are automatically prefixed with the name of the project.
- Give unique names to all volumes regardless of their attributes.
- Encourage users to use unique names as a matter of good practice, even for files located on different volumes.

5.2.2 Multi-Volume File Extension

Bear in mind when pre-allocating volatile files (like the SYSOUT file) that the rule for dynamic file extensions is as follows:

File extension begins on the same volume as the previous extension and continues on a new volume where the file does not already exist.

For example, suppose a file is allocated on volume A, and then extended to volume B. When volume B is full, the file can be extended to another volume but never to volume A, even if space has become available on volume A.

5.3 Automatic Operation

The duties of the operator are being increasingly automated, both:

- to reduce the amount of time the operator has to spend at the system console, and
- to ensure that the best possible action is taken in response to events.

This automation is known as the Distributed Operator Facility. It can consist of one or more of the following products:

DOF 7-MC:	the Multi-Console Facility
DOF 7-OL:	the Operator-Less Facility
DOF 7-RM:	the Remote Multiplexed Facility.
DOF 7-PO:	the Programmed Operator Support Facility.
DOF 7-SM:	the Script Manager Facility

5.3.1 The Multi-Console Facility (DOF 7-MC)

The Multi-Console Facility is an optional facility which allows one GCOS 7 system to have several system consoles, and therefore more than one Main Operator to be logged on at the same time. These operators can have either the same or different access rights.

The System Administrator decides which commands a particular operator can use, and allocates different categories of system operation to different operators, perhaps as follows:

- device operations
- batch operations
- incident handling
- monitoring system performance
- reporting network activity
- file handling

Environments and Message Filtering

Each console can be dedicated to a specialized operator function. That is, each operator works under a specific project which has a default environment or limited range of commands that can be accessed. An additional feature connected with environments is message filtering. Only those system messages associated with the function of a particular operator are sent to that operator's console.

Using the GCL commands CREATE_FILTERSET and CREATE_FILTER, a Main Operator can specify the kinds of system messages that he wishes to receive. A filter-set defined by one Main Operator can be assigned a higher priority relative to filter-sets defined by other Main Operators.

All the messages and commands generated at the various consoles are stored in a central file, which can be read and analyzed.

Using the Multi-Console Facility, system consoles can be reconfigured without interrupting the running of the system, and the failure of a system console can be circumvented.

5.3.2 The Operator-Less Facility (DOF 7-OL)

This facility is also known as unattended mode and makes available two main options, AUTO and SILENT, which reduce the amount of time the operator has to spend at a system console.

AUTO, means that a GCOS 7 system can be automatically initialized and loaded, with no manual operator intervention unless an unusual event occurs, or an answer is needed to an unexpected question.

Automatic initialization or new software loading options can be specified (using MODIFY_RESTART_OPTIONS) for the next session, when the operator shuts down the system.

When the system is to be initialized automatically, the "Predefined Replies to Questions" feature provides default replies to questions which are likely to be asked of the operator during initialization.

DOF 7-OL is not available for coupled systems.

5.3.3 The Remote Multiplexed Operator Support (DOF 7-RM)

This is an optional facility, which can be used in the IOF environment from any console which supports FORMS.

DOF 7-RM allows one DPS 7000 console to monitor the IOF activity of several GCOS 7 systems interconnected in a DSA primary network.

Up to 9 DOF 7-RM applications may exist in the same network, with each able to monitor up to 64 systems.

Note that because DOF 7-RM is an interactive application, it cannot be activated by a Main Operator.

5.3.4 Programmed Operator Support (DOF 7-PO)

DOF 7-PO is an optional software product which makes possible the automation of many routine tasks normally performed by operator personnel.

It provides a GPL or C language programming interface that allows a user application to send commands to be executed by a local or remote GCOS 7 system, and to receive messages in response.

A typical DOF 7-PO application can manage all or part of a DPS 7000 system, or several systems, performing such as:

- starting jobs for users
- incident handling
- managing the network
- monitoring system resources
- reporting network activity.

A Main operator can use a set of privileged commands to monitor and control system and network operation. On large or multi-system sites, there may be several Main Operators, each responsible for a particular aspect of system operation, for example, network administration, batch operations, or responding to system events. Each operator will then have access to a specific set of commands relating to that task. Only those system messages associated with that task will be sent to that operator's console.

A DOF 7-PO application can be written to automate operator functions. The application can then make decisions and take action based on the information it receives. For example, if the system is overloaded with too many jobs, it can hold some of them. If resources requested by a job are not available or unknown, it can cancel the job.

Such automation increases the reliability and availability of the system. It also frees the operator for other tasks.

A DOF 7-PO application may be especially useful for operating smaller DPS 7000 systems at sites where a dedicated operator is not available, or when the processing done by the system is repetitive and predictable enough to make the presence of a human operator unnecessary.

The DOF 7-PO programming interface can be used to create applications to manage OSI/DSA networks. In particular, a DOF 7-PO application can receive, in real time, all the events sent over the network.

Full details are given in the *DOF 7-PO User's Guide*.

5.3.5 The Script Manager (DOF 7-SM)

The Script Manager is an optional software product which provides an easy-to-use GCL interface for creating DOF 7-PO applications. It offers a simpler (though less powerful) alternative to the GPL interface.

The operations performed by the Script Manager are called "automata". These are activated by "servers", which wait in an infinite loop for specified events to happen, and perform pre-defined actions when these events occur. The pre-defined actions are specified by stored sequences of Script Manager commands, called "scripts". Scripts can call any standard or user-defined GCL command belonging to the MAIN and H_NOCTX domains.

Full details are given in the *DOF 7-SM User's Guide*.

5.4 Coupled Systems Operation

5.4.1 General Concept

A coupled systems environment consists of two completely independent GCOS 7 operating systems, together with their physical resources, on a site using the Coupled Systems product.

The Coupled Systems product allows certain specified disk volumes to be accessed simultaneously by jobs running on both systems. The disk controllers must be connected to and recognized by both systems.

Each of the coupled systems is complete and independent, with its own operating system and peripherals. As well as the shared volumes, each system can have its own non-shared volumes.

Each system has its own system volume, which may or may not be mounted on a sharable drive. If the system volume of System A is mounted on a sharable drive, this drive is automatically reserved at Initial System Load for System A and is protected against any access from System B.

Two main advantages of the Coupled Systems product are:

- Flexibility of operation. You can have one system operating in batch (or IOF) mode and the other in TDS mode, or both systems in batch (or IOF) mode, or both in TDS mode
- Backup security. For example, where both systems are running TDS applications, if one system fails, the other can immediately take over.

Further advantages are:

- Configuration flexibility. It is not necessary for the configurations of the two systems to be the same. Neither is it necessary to couple the same DPS 7000 models.
- Certain peripherals need exist only once for the two systems.
- Temporary space may be common between the two systems.
- A single Site Catalog can contain all the information concerning the users and projects on the site.
- Private catalogs and data files may be shared by users of both systems.

There are, however, certain constraints in the coupled systems environment:

- It is not possible to automatically share or balance the work load between the two systems.
- The two systems must have independent system files.

5.4.2 File Management

Because users of both systems have access to the files which are on shared disk volumes, user data files for the two systems need not be duplicated. Therefore the user-managed files and the data managed by the system for the user (private catalogs and the Site Catalog) need exist only once for both systems.

All cataloged files located on a shared volume can be accessed simultaneously by jobs running on both systems. Uncataloged files can be accessed alternately, by one system then the other. Both systems can allocate space for permanent and temporary files on shared volumes.

As for the sharing of files in a single system, GCOS grants access to the requested shared file only after checking that this access is compatible with those access rights already granted.

5.4.3 Data Security

Automatic protection and synchronization mechanisms operate at volume level. These mechanisms always note the change of state of the shared volumes and run the Automatic Volume Recognition (AVR) routine when a drive supporting a coupled volume changes state from STANDBY to READY. TheAVR routine is run for all transitions to the READY state in order to take into account any reconfigurations. These reconfigurations are possible even in the case of on-going I/O operations.

5.4.4 Transactional Context Recovery Facility (TCRF)

TCRF is an application which can be run in a coupled systems environment and is restricted to the System Administrator. It enables one or more transactional applications to be restarted with minimum delay on the backup system if one system fails.

TCRF works by transferring one or more transactional applications from the failed system to the backup system without interrupting processing on the backup system. It does this in the following way:

- by producing a list of all the TDS and batch files in use on the failed system, and a list of all the steps currently accessing files
- by using the journal mechanism to restore the files to a stable state
- by replacing references to the After Journal in the SYS.CATALOG of the backup system with references to the After Journal of the failed system.

5.4.5 TDS HA

Evolution from GCOS 7-V2, V3, V5

The introduction of the HA (High Availability) feature is designed to increase the availability of TDS applications by reducing system downtime.

The TDS programmer and end-user's visibility is not affected by HA.

However, the TDS administrator at an HA site is affected in the following ways:

- There are modifications to the JCL used to start TDS
- The START/STOP of a TDS application is performed through HA (CMSC) commands.

For more information, refer to:

High Availability Administrator's Guide

High Availability Concepts.

6. System Regulation

This chapter is an overview of the concepts, the techniques, and the tools for regulating your DPS 7000.

It explains in broad terms the system's own mechanism for self-regulation, and how this mechanism can be refined, under supervision by the System Administrator, to achieve specific operational and optimization objectives.

Many of these objectives are expressed in the form of parameter values called dimension attributes. The notions of a dimension and of dimension management are fundamental to system regulation and are explained later on in this chapter.

Generally speaking, there are only two reasons why the system might need regulating: either a particular operational objective has changed or there is a problem of performance.

A New Operational Objective

Whenever you need to run an application for which the normal rules of resource allocation are inappropriate (a particularly critical TDS job or an unusually large Batch job for example), you will need to intervene with dimension management commands to change these rules.

A Problem of Performance

Genuine performance problems arise directly or indirectly from poor resource distribution. In most cases the immediate problem will be one of resource saturation (i.e. bottleneck problems), for which there are usually routine "workload balancing" solutions.

However the real problem may be much more subtle, and balancing the workload might offer only a temporary solution. There may be, somewhere, an incoherence in your policy concerning a particular application or a particular job class. This is a problem of strategy, requiring perhaps a re-evaluation of specific operational and/or optimization objectives. Identifying this kind of problem becomes easier as you gain experience with the system's normal patterns of behavior, and, quite often, instinct will suffice.

6.1 The Regulation Mechanism

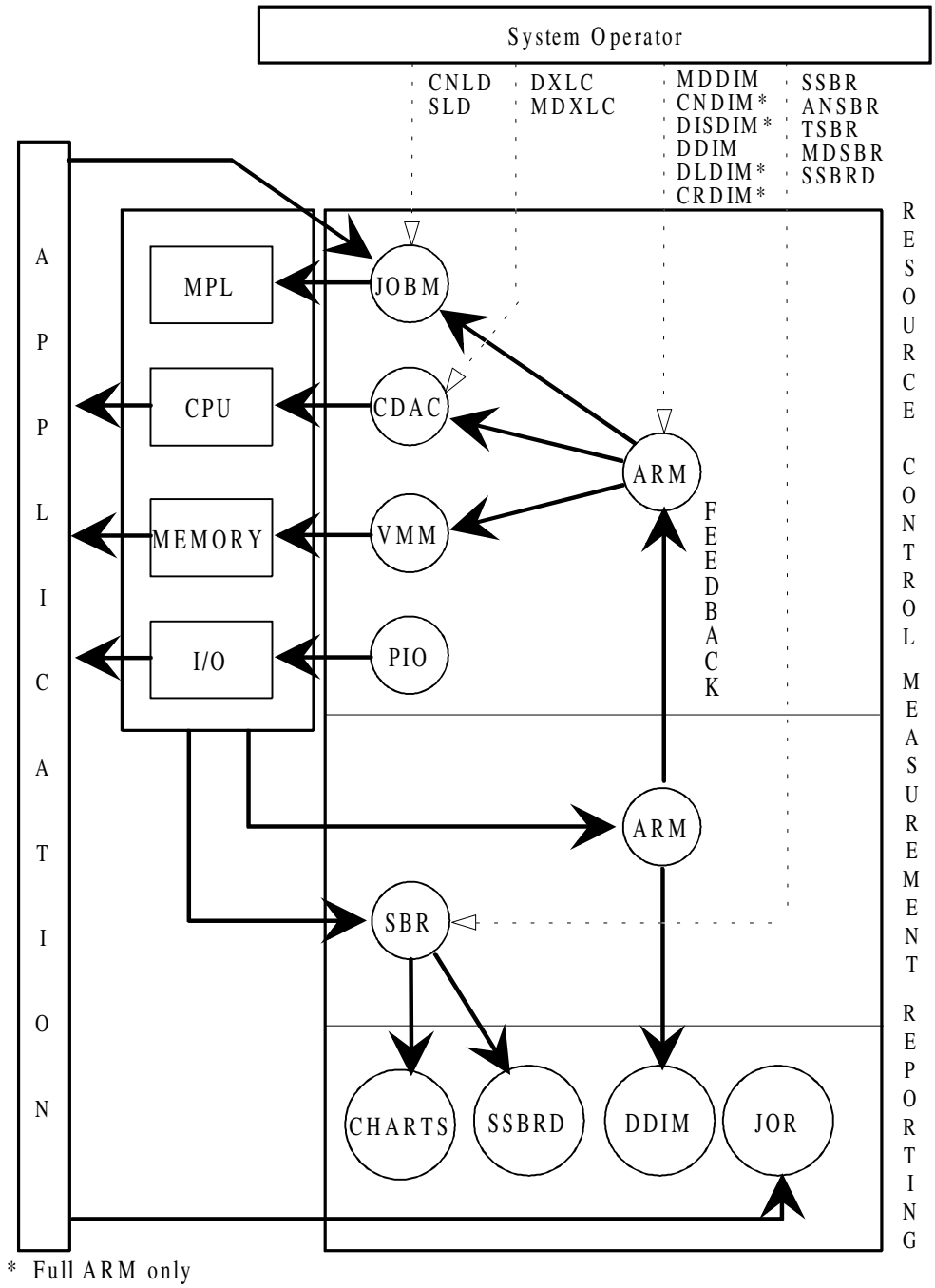

Figure 6-1. DPS 7000 Regulation Mechanism

6.1.1 Automatic Resource Manager (ARM)

At the center of the mechanism is the Automatic Resource Manager (ARM). This assimilates the operational requirements for the system into a set of specific control rules for governing memory usage, CPU usage, and multiprogramming levels. These determine working set decisions, hardware priorities, dispatching policies, and scheduling levels.

At regular time intervals, ARM assesses the impact of its control rules by measuring the current workload distribution. It then uses these measurements as feedback for adjusting and optimizing the control rules.

Figure 6-1 above gives an overview of this mechanism, and shows where and with which commands the Main Operator can intervene.

A new concept brought by the new Multi-Technology architecture is "Customer Dedicated Processors" that enables the customers to dynamically optimize processing power according to their needs for the following "privileged functions":

- GCOS 7 (Nucleus, servers & utilities),
- Oracle,
- Open 7,
- DataWarehouse 7,
- IQS,
- Y2KTEST: jobs running under Private Timer.

A DPS 7000/MT system is made up of 1 to 4 different types of processors called "DOMAINs":

- STANDARD domain, made of IPUs (Instruction Processors Units), enables to run any function, privileged or not,
- CDP1 domain, made of EPUs (Customer Dedicated Processors Type 1), enables to run all the privileged function not connected to an other domain,
- CDP2 domain, made of FPUs (Customer Dedicated Processors Type 2), enables to run one privileged function selected by the system administrator,
- CDP3 domain, made of GPUs (Customer Dedicated Processors Type 3), enables to run one other privileged function selected by the system administrator.

The processor's configuration determines which of the four domains exist.

For full information, see *The ARM User's Guide*.

6.1.2 Execution Level (XL)

The execution level (XL) is the degree of urgency computed for each process. It ranges from 0 (lowest) to 100 (highest). This value determines the Execution Level Class (XLC) for the process and hence its CPU allocation rights. With Full ARM, the average XL of a dimension also determines the dimension's memory allocation.

With Basic ARM, the XL of a process is proportional to the XPRTY of its step and is therefore a static value.

With Full ARM, the XL of a process varies according to the service rate, the RELWGHT, the BWGHT, and the XL range of its dimension. XPRTY can also be taken into account. The XL is a dynamic value, and therefore so is its XLC.

Full details on how the XL is computed are given in *The ARM User's Guide*.

6.2 Dimension Management

Dimension management is the process by which the System Administrator communicates operational and optimization objectives to the system. These objectives are expressed as parameter values called "dimension attributes", and are assimilated by ARM into specific resource allocation rules.

6.2.1 The Dimension Concept

A dimension is a set of allocation rules and policies for controlling memory utilization, CPU utilization, and multiprogramming levels.

When an object (e.g. a job class group or a load module) is assigned to a dimension, the object is subject to the allocation rules and policies of that dimension. This simplifies resource management tasks. If some objects are consuming too much or too little of the system's resources, the dimension can be modified so that its attributes are more appropriate for the objects in question.

With Full ARM, job class groups and load modules can be disconnected from their current dimension and reconnected to a more suitable one. If such a dimension does not exist, a new one can be created. This is useful when a particular application needs to be isolated and its resource allocation protected against competition with other applications.

6.2.2 The Standard Dimensions

The system is delivered with four standard dimensions, called SYS, TDS, IOF, and BATCH. They have predefined attributes and most configurations use them without any particular modification, as they provide satisfactory results in mixed and in dedicated environments.

Table 6-1. Default Attributes for the Standard Dimensions (Full ARM)

Dim Name	Assignment Criterion	RELWGHT	MPL Range	XL Range	BWGHT	PA	ICA	HO
SYS	JC=R-ZZ	100	50-50	0-100	0			
TDS	Unassigned TDS steps	100	50-50	60-99	0		1	
IOF	JC=Q-QZ	70	50-200	30-69	20			
BATCH	Default	50	2-200	10-49	30	1		1

Brief descriptions of the dimension attributes and associated concepts are given below.

6.2.3 Dimension Attributes

Assignment Criterion

This is the list of load modules, job classes, or job class groups that can be assigned to the dimension. The list is modifiable with Full ARM.

RELWGHT Attribute *(Full only)*

This is the degree of importance of the dimension, expressed as a number from 0 to 100. The higher the number, the more resources the applications running in the dimension will receive. ARM uses this value in the computation of the XL.

MPL Attribute *(Full/Basic)*

With Full ARM, this is the minimum and maximum multiprogramming level allowed for the dimension, expressed as a pair of digits from 0 to 254. Unequal bounds allows ARM to regulate the dimension's multiprogramming level. Equal bounds disables ARM's regulation.

With Basic ARM, there is no automatic regulation of the multiprogramming level, just a maximum limit specified by the dimension's MPL attribute.

XL Attribute *(Full ARM only)*

The XL range gives the minimum and maximum execution level allowed. Jobs running in dimensions whose XL ranges overlap may have to compete for resources. Jobs running in dimensions whose XL ranges do not overlap are isolated from each other and do not have to compete for resources.

PA Attribute *(Full ARM only)*

This specifies whether or not the step's XPRTY value is to be taken into account in the computation of execution levels.

BWGHT Attribute *(Full ARM only)*

This specifies the importance of system balance, expressed as a number from 0 to 100. The higher the value the more important is system balance and the less important are the needs of the dimension.

ICA Attribute *(Full/Basic ARM)*

This controls whether each step in the dimension is allocated exclusive or shared memory resources. It also enables/disables ARM's control of hardware priorities.

With Basic ARM, the ICA atribute cannot be modified.

HOLD Attribute *(Full ARM only)*

This allows/disallows automatic step suspension within the dimension.

6.3 Memory Management

6.3.1 Memory Organization

Physical Topology

Main memory is divided into 3 basic areas:

- The "resident area" which is reserved for the system firmware and for GCOS.
- The "locked area" which interfaces with the CPU and the system firmware, and whose contents cannot be transferred out of memory or even moved about within memory.
- The "swappable area" which contains loadable and relocatable software; i.e., code and data that can be transferred to and from backing store and also moved about within memory.

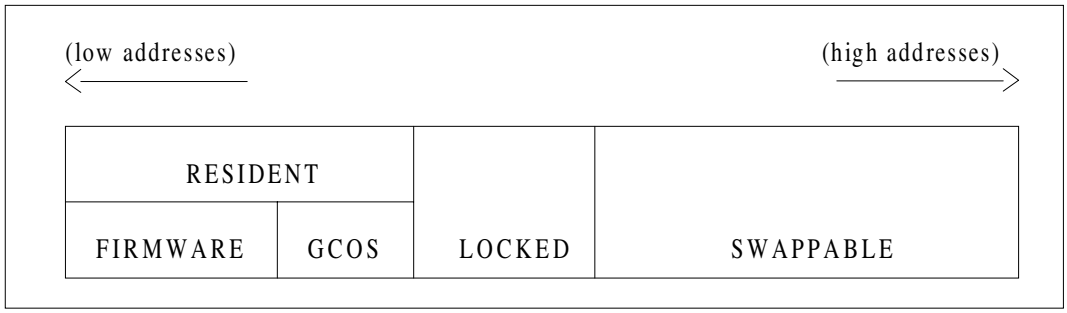

Figure 6-2. General Memory Topology

Segments and Pages

The code and data required to execute a given step is divided into logical units called segments. Segments are further divided into units of 4 Kbytes, called pages.

The transfer unit between backing store and main memory is the page. VMM does not necessarily transfer all pages of a segment at once, but only those required for the step to execute. Thus a segment may occupy less space in memory than on backing store. On the other hand, segments which are not exact multiples of 4 Kbytes are wasteful of memory space. A 256-byte segment, for example, will need a whole 4 Kbyte page in memory.

6.3.2 Memory Sharing

Certain memory segments can be shared by different steps or step processes. There are three levels of segment sharability as far as VMM is concerned:

- those that can be shared by all system steps (type 0 and type 10 segments)
- those that can be shared by different processes of a given step (type 2 and type 12 segments)
- those that cannot be shared, even by processes of the same step (type 3 segments)

VMM's decisions concerning memory sharability are in fact determined by ARM, which views memory as being divided logically into regions. It allocates these regions as follows:

- Region 0 is reserved for steps running under the SYS dimension
- one region for each dimension that has ICA=0
- one region for each step running under a dimension that has ICA=1

6.3.3 Memory Allocation

Instantaneous Working Set (IWS)

This is the total amount of memory allocated to a given step at a given moment, as measured by ARM. It consists of:

- FIXED pages; i.e., locked memory (buffers, etc)
- RU pages; i.e., swappable memory which has been recently used
- LRU pages; i.e., swappable memory which has been least recently used

The memory currently used by the step is its fixed memory plus its recently used memory.

At any moment: IWS size > USED size > FIXED size

Fixed Size Checks

ARM regularly monitors the amount of fixed memory being consumed by a step.

If this is measured to be greater than max(4*DWS, 1 Mbyte), the step is aborted with the return code CMWSOV.

DWS is specified by the JCL SIZE statement (or GCL equivalent).

Pool Memory Checks

ARM also monitors the amount of pool memory (unallocated memory) available. While this remains above 2 Mbytes, no control action is taken. VMM is free to allocate memory as and when each region requires it. There is no page aging and no rewriting to backing store. Memory is said to be in an "unconstrained state".

If, however, the amount of pool memory goes below 1 Mbyte, memory is said to be in a "constrained state". Three things happen:

- The page aging process begins. This is the VMM process of distinguishing recently used (RU) pages from the least recently used (LRU) pages, based on the page out objective supplied by ARM.
- The rewriting of the LRU pages to backing store begins.
- Limits are imposed on the number of pages that can be swapped into memory. These limits are called CWSMIN and CWSMAX. They are calculated by ARM for each region.

These actions continue until the pool memory returns to an unconstrained state (POOL > 2 Mbytes).

Further details about ARM's calculation of page out objectives, and of CWSMIN/CWSMAX, are given in the *ARM User's Guide*.

6.4 CPU Management

The CPU is allocated to a process according to the Execution Level Class (XLC) of a process.

6.4.1 Execution Level Class (XLC)

Associated with each XLC is:

- a hardware priority or set of hardware priorities (HWPRTY)
- a CPU dispatching policy (ATTR)
- a maximum CPU consumption rate (MAXCPU)

Table 6-2. XLC Attributes

XL	XLC	ATTR	MAXCPU	HWPRTY
100	0	F	100%	0-15
90-99	1	F	100%	8
80-89	2	FS	100%	9
70-79	3	FS	100%	10
60-69	4	FS	100%	11
50-59	5	O	30%	11-12
40-49	6	FS	40%	12
30-39	7	O	50%	12-13
20-29	8	O	60%	13-14
10-19	9	O	80%	14-15
0-9	10	FS	100%	15

F = Fixed, FS = Fixed Sliced, O = Optimized

6.4.2 Hardware Priorities

A step's hardware priorities are managed within the maximum and minimum limits defined by the step's XLC.

With Basic ARM and also with Full ARM where the step's dimension has ICA=1, the XLC is derived from the step's XPRTY. The XLC is therefore static, and so also are the hardware priority limits.

Where the step's dimension has ICA=0, the XLC is derived from the step's XL level computed by ARM (and thus ultimately from the dimension's RELWGHT). The XLC in this case is dynamic, and therefore so are the hardware priority limits.

Priority limits can be modified directly by the MODIFY_XL_CLASS command and indirectly through modifying dimension attributes (ICA, RELWGHT, BWGHT, and XL). Refer to the *ARM User's Guide* for details.

6.4.3 Dispatching Policies

There are three different policies for the dispatching of CPU time: the Fixed Policy, the Fixed Sliced Policy, and the Optimized Policy. These are selected according to the step's XLC.

Fixed policy

A process takes control of the CPU whenever it needs it, provided that its hardware priority is higher than that of the process (or processes in the case of multi-processors) currently executing. The process with the next highest priority uses the CPU during the times when the first process leaves it available. Furthermore, the priority that loses control of the CPU due to the intervention of a higher-priority process waits ahead of other processes having the same priority as itself.

This policy works well, but if several steps are executing with processes of the same hardware priority there is a danger that one of these may monopolize the CPU.

Fixed Sliced Policy

The same principles are applied here as with the Fixed policy, but, in addition, the CPU time is allocated equally between processes with the same hardware priority. This prevents the CPU from being monopolized by a process having the same priority as others.

Optimized Policy

Like the Fixed Sliced policy, the CPU is allocated equally between processes with the same hardware priority. In addition, a process' hardware priority is dynamically modified (within the permitted range of its XLC) according to its profile. For example, an I/O-bound process is given a higher priority if the system is generally CPU-bound.

The XLC of a process determines its hardware priority, its dispatching policy, and its maximum CPU consumption rate. The XLC is derived from the XL of the process, which is computed by ARM. The following table shows the default correspondence between the XL and XLC of a process.

Table 6-3. XLC Attributes

XL	XLC	ATTR	MAXCPU	HWPRTY
100	0	F	100%	0-15
90-99	1	F	100%	8
80-89	2	FS	100%	9
70-79	3	FS	100%	10
60-69	4	FS	100%	11
50-59	5	O	30%	11-12
40-49	6	FS	40%	12
30-39	7	O	50%	12-13
20-29	8	O	60%	13-14
10-19	9	O	80%	14-15
0-9	10	FS	100%	15

F = Fixed, FS = Fixed Sliced, O = Optimized

6.5 Backing Store Management

Virtual memory consists of the main memory plus its logical extension onto several disk volumes - these disk extensions are the backing store.

6.5.1 Backing Store Organization

The backing store storage space is subdivided into four areas:

- Paging backing store
- Library backing store
- Permanent virtual memory files backing store
- Temporary virtual memory files backing store

The paging backing store consists of files SYS.BKST, SYS.BKST1...SYS.BKST15. It is used for the pages swapped in and out of main memory.

The library backing store consists of files SYS.LIB, SYS.LIB1...SYS.LIB15. It is used for sharable modules, PLMs, and repeatable steps and checkpoints.

The permanent virtual memory files backing store consists of files SYS.PVMF, SYS.PVMF1...SYS.PVMF15. It is used for permanent backing store files required internally by the system.

The temporary virtual memory files backing store consists of files SYS.TVMF, SYS.TVMF1...SYS.TVMF15. It is used as temporary backing store by certain processors.

6.5.2 Space Allocation

The System Administrator is responsible for allocating enough space to ensure that backing store will not overflow under most operating conditions, and will not be overloaded to create a bottleneck for the system.

Each of the four areas noted above must be estimated separately; there is no way of merging them.

Some recommendations for estimating space are given in Chapter 13.

6.6 MPL Management

In systems equipped with Basic ARM, the MPL is static and modifiable only through the operator command MDDIM.

In systems equipped with Full ARM, the MPL is dynamically optimized. Its computation is based on several CPU and backing store variables used to indicate whether the system is "overused", "underused", or "normally used".

If the system is "normally used", ARM adjusts the MPLs in order to bring each dimension to the same level of contention.

If the system is "overused", ARM decreases the MPL of the dimension whose contention level is the lowest and whose MPL is not yet at its mininum.

If the system is "underused", ARM increases the MPL of the dimension whose contention level is the highest and whose MPL is not yet at its maximum.

For further details, refer to the *ARM User's Guide*.

6.7 Analysis Tools

The System Administrator has several tools for monitoring and evaluating system performance. These include:

- The System Behavior Reporter (SBR)
- The DDIM command
- The Transactional and Interactive Load Simulator (TILS)

These are outlined below.

Job Accounting Records are also useful indicators of system load and performance. These are outlined in Chapter 12 and their record descriptions are given in Appendix D.

6.7.1 System Behavior Reporter (SBR)

SBR is a tool for measuring and evaluating the use of hardware resources and the dynamic behavior of steps. It monitors system and application behavior on several levels.

The first level provides a concise report on the workload and the use of system resources. The information given includes:

- details about the number and types of programs running
- diagrams about missing pages and disk I/O activity
- diagrams about real time activity
- statistics about the CPU, memory, and device usage.

Another level gives more detailed reports containing information about the service times of devices and I/O rates. It enables the user to detect possible system inefficiency due to overuse of a particular device (bottlenecks).

Another, extended level provides still more specific information on I/O operations on backing store, system files, and the use of system services. Specific instrumentation can also provide detailed information on:

- CPU usage
- memory usage
- disk controller activity
- UFAS-EXTENDED buffer pools
- TDS behavior.

A real-time display can also be used to obtain immediate information about system behavior.

Refer to the *SBR User's Guide* for full details.

6.7.2 The DDIM Command

This command gives on-line statistics about system load and dimension activity. It also provides statistical information about High Relational Performance (HRP) processor activity. The Full ARM version gives:

- the current attributes and connections of each dimension
- the number of jobs currently running, suspended, held, rejected, or waiting in each dimension
- absolute service rates, PGTs, number of steps, input message rates, system response time, contention level, average XL for each dimension
- the resource usage of each dimension and diagnostics on whether the system is overused, normally used, or underused
- the resource usage of each step within the dimension
- the load distribution between the standard job category and the HRP job category

Refer to the *ARM User's Guide* for full details.

6.7.3 The Transactional and Interactive Load Simulator (TILS)

TILS is a utility that simulates potential TDS and/or IOF terminals on the system even though the actual network may not yet be installed or working.

It creates an artificial load on the central system by simulating a group of terminals transmitting and receiving messages. At the end of the simulation session results are printed out. These include:

- Trace information
- Terminal activity tables
- Histogram of response times
- Chronology tables with values for each minute

Refer to the *TILS User's Guide* for full details.

6.7.4 The CNFUNC, DISFUNC and DFUNC Commands

These ARM commands connect (CNFUNC), and disconnect (DISFUNC) privileged functions from given CPU resource domains (see 6.1.1). CNFUNC allows a function to use dedicated processor resources. DFUNC displays information about functions and resource domains. For details, see *The ARM User's Guide*.

6.8 Performance Problems

If the system is behaving abnormally (e.g., response times or throughput are significantly lower than they should be), this may imply the need for system regulation.

As mentioned earlier in this chapter, it is advisable to have a systematic approach to solving a performance problem. It is also essential to have a thorough understanding of the system's normal workload.

6.8.1 The Importance of Knowing Your Workload

The System Administrator should know the system's workload sufficiently well to be able to recognize normal and abnormal behavior, to predict the effects of changes in operations, applications, or usage, and to recognize typical throughput rates. He/she should readily answer such questions as:

- What is the typical number of users on the system at any time of day?
- What is the typical response time for this number of users?
- What are the peak hours of operation?
- Which regularly run jobs are intensive users of the CPU? Of memory? Of disks?
- Are there large jobs that could be running in batch?
- Which applications involve the most memory swapping?
- Are there any known system bottlenecks?

If you cannot answer these questions, you will not easily address any complex performance issues and you should spend some time observing system operation with the help of the SBR utility.

Over time you will learn the typical page fault rate for your system, the typical CPU usage, the normal memory usage, typical modes of operation, and so forth.

6.8.2 The Correct Approach to a Performance Problem

The exact approach to a performance problem naturally depends on the exact nature of the problem. There are, all the same, certain rules of thumb which can be applied to almost any problem.

Start a preliminary investigation with the following questions in mind:

- Are the response times poor?
- Is the throughput poor?
- Is the problem global or specific to a particular environment?
- Is the workload well balanced?
- Is the system overused, normally used, or underused?

The procedures for establishing answers to these questions involve a straightforward use of the DDIM and SBR analysis tools.

If this preliminary investigation does not lead to any useful conclusions, it may be necessary to embark on a major tuning exercise.

Bear in mind, though, that tuning can be costly and that it is not a cure-all. It should be regarded as a last resort solution and attempted only if you are certain that it is both justified and indeed a solution.

Tuning cannot cure the following sources of performance problems:

- Improper operation
- Unreasonable performance expectations
- Inadequate hardware configuration for the workload, such as too little memory, too few I/O channels, too few disks, and so on
- Improper device choices for the workload, such as using tapes when disks are preferable, installing disks with insufficient speed or capacity, etc.

The cost-effectiveness of a tuning exercise must always be taken into consideration. When properly done, tuning involves careful, time-consuming studies executed by skilled personnel. It may require statistics gathering and analysis, which, if carried out over long periods of time using SBR, use system resources as well as human resources.

Although these costs can be difficult to quantify, they are real and should not be overlooked.

7. System Maintenance

Bull's policy of "Predictive, Programmed, and Partner-Oriented (3P)" maintenance is designed to provide coverage for new systems and networking features.

Predictive Maintenance deals with the process of monitoring system interrupts and recovery to determine when a breakdown situation has been reached.

Programmed Maintenance concerns the inclusion of accurate preventitive actions in standard routine procedures either regularly or after the problem has been examined by support personnel.

Partner-Oriented Maintenance refers to a constructive partnership set up between customers and national Customer Service Centers (CSCs) to implement standard maintenance procedures.

7.1 Service Request Procedures

7.1.1 Customer Call Preparation

In the case of software difficulty, you should collect as much information as possible before calling the Bull Customer Service Center for assistance.

Below is a checklist of what you should send to the Service Center, depending on the type of software problem:

In all cases:

- the main console listing
- the release identifier and the version number of the software components affected.

In the case of a system crash:

- the crash identifier displayed on the main console
- the corresponding memory dump, which must be saved on disk in a DPAN dump file.

If a customer batch application or a GCOS 7 batch utility aborts:

- the abort identifier (exception code or return code)
- the Job Occurrence Report where the error appears
- the corresponding dump, if any, from the SYSOUT file.

If a TDS aborts:

- the Job Occurrence Report where the error appears
- the corresponding dump, if any, from the SYSOUT file, or saved in a DPAN dump file
- the traces, if any, which must be saved on disk.

In other cases, the Bull Service Center will ask for specific operations, information, and listings depending on the type of difficulty.

7.1.2 Call Processing

During the processing of a call, the Bull Service Center may ask you to apply an emergency correction to resolve the difficulty. This correction may already be available on site if TSUINFO files are installed on one of the system disks. In this case, the Service Center will provide the correction identifier (SER number) for the patch to be applied on the system disk.

If the correction is not available on your site, the Bull Service Center will provide it by the Remote Maintenance Service, by mail, or, in an emergency, by sending a BULL representative.

7.2 Telemaintenance

7.2.1 Telecontrol System Facility (TSF)

The Telecontrol System Facility is the main feature of Predictive Maintenance used on DPS 7000 systems.

TSF performs automatic analysis of the error logging file and displays an alert message on the system console whenever thresholds of fault rates in the system configuration are exceeded.

TSF can call the Bull Customer Service Center automatically, or on customer initiative, to transmit via a switched network information following a diagnostic. The Bull Service Center will then be able to call you and propose a solution to the problem.

The telecontrol system is described in the appropriate *Operator's Guide*.

7.2.2 Remote Maintenance Service (RMS)

RMS allows remote access to a DPS 7000 site from a Bull Technical Assistance Center (TAC), via a switched network. This access allows TAC specialists to analyze a customer's problem and, when possible, solve it during a remote session without interrupting customer operations.

There are two types of RMS session:

- RMS Hardware:

  This consists of a physical link between the DPS 7000 Service Processor and the Bull remote maintenance station.

- RMS GCOS

  This is a logical link between the DPS 7000 communications interface (DATANET) and the Bull remote maintenance station.

While an RMS session can only be initiated by Bull Service Center personnel, the customer can enable or disable the session at his discretion. The procedures used to open and close an RMS session are described in the appropriate *Operator's Guide*.

7.3 Maintenance Procedures

7.3.1 Patching

Patching a software domain consists of the application of an emergency correction to one of the system disk sets (P-set, P2-set, R-set). The procedures for each domain are described in the *System Installation, Configuration, and Updating Guide*.

An emergency correction cannot be applied to GSF.

7.3.2 Dumps

System Crash

The ISL command DUMP can be used when a system crash dump is available. The procedure to store the dump on disk is described in of the *System Operator's Guide*.

The procedure for saving a system crash dump and transferring STAR elements to tape (DP_SAVE), plus printing the dump on a line-printer (DP_PRINT), are described in the *System Operator's Guide*.

TDS Abort

The procedures to store a TDS dump on disk (DP_ALLOC), to save it on tape (DP_SAVE), to restore it from tape to disk (DP_LOAD), or to print it (DP_PRINT) are described in the *TDS Administrator's Guide*.

7.3.3 Other Aids

Main Console Listing

When a hardcopy console listing is not available, a full or partial print of the SYS.LOGC file can be made using the PRLOGC job described in Appendix B of the *System Operator's Guide*.

Release and Technical Status Identifiers

The GIUF function DISPLAY_STATUS allows you to display the current status of all domains from any system disk set. This function is documented in the *System Installation, Configuration, and Updating Guide*.

Saving Traces

The IOF commands MODIFY_DATA_MANAGEMENT_TRACE (MDDMTR) and SAVE_DATA_MANAGEMENT_TRACE (SVDMTR) are to be used when requested by the Bull Service Center.

Error Logging File and PRLOG

The contents of the system error logging file may be printed using the PRLOG job described in the *System Operator's Guide*.

8. Project/User/Billing/Station/Application Management Procedures

This chapter explains the procedures for listing and registering PROJECT/USER/BILLING/STATION/APPLICATION information in the Site Catalog. It gives the full syntax and parameter descriptions of the Site Catalog maintenance commands used to create, modify, list, move, and delete projects, users, billings, stations, and applications.

8.1 Accessing Site Catalog Maintenance Commands

The GCL command MAINTAIN_CATALOG or JCL command CATMAINT provide access to the set of Site Catalog maintenance commands.

GCL Syntax:

```
{ MAINTAIN_CATALOG }
{                  }
{ MNCAT            }

       [ COMFILE = file78 ]
- - - - - - - - - - - - - - - - - -
       [ SILENT { bool | 0 } ]
       [ PRTFILE = file78 ]

       [ OUTFILE = output-file-description ]

                 { CAT            }
                 { CAT{1|2|3|4|5} }
       [ DYNALC = {               } ]
                 { UNCAT          }
                 { TEMPRY         }

       [ ALLOCATE = (file-allocation-parameters) ]

       [ OUTDEF = (file-define-parameters) ]

       [ APPEND = {0|bool} ]
```

Parameters:

COMFILE	The file from which MAINTAIN_CATALOG commands are read. It may be an input enclosure or member of a source library, and may be temporary or permanent. The format can be DATASSF or DATA. If this parameter is omitted, commands are read from the user's terminal.
SILENT	If 1, errors are reported in PRTFILE. If 0 (default), errors are reported at the user's terminal.
PRTFILE	The printer file. If this parameter is omitted, the report is displayed at the user's terminal, and stored in the standard SYSOUT file.

OUTFILE	The description of a sequential output file used to receive the information returned by the current command. The format of the output records is described in the *GPL System Primitives*. The OUTFILE parameter supports the extended syntax for multifile tape. See the *IOF Terminal User Reference Manual Part 1*.
DYNALC	Request dynamic allocation of OUTFILE by specifying its file status: CAT cataloged CATi cataloged in one of the attached catalogs (i=1 to 5) UNCAT uncataloged TEMPRY temporary
ALLOCATE	The output-file-allocation parameters expressed as a file-allocation parameter group, as detailed in the *IOF Terminal User Reference Manual Part 2*.
OUTDEF	Additional processing parameters expressed as a file-define parameter group, as detailed in the *IOF Terminal User Reference Manual Part 2*.
APPEND	If 1, append in OUTFILE. Otherwise, OUTFILE is overwritten.

Constraints:

- SILENT=1 is meaningful only when PRTFILE is specified.
- COMFILE is mandatory in batch mode.
- When OUTDEF, ALLOCATE or DYNALC is specified, OUTFILE must also be specified.
- When APPEND is specified, OUTFILE must also be specified.
- When ALLOCATE is specified, DYNALC must also be specified.
- When OUTFILE is a library and DYNALC is used, OUTDEF must also be used to specify FILEFORM=BFAS and FILEORG=LINKQD.
- Currently, OUTFILE and the associated, parameters are only taken into account by the LIST_PROJECT, LIST_BILLING and LIST_USER processor commands (for a complete discussion, see also the *GCOS 7 Catalog Management User's Guide*).

EXAMPLES:

```
MNCAT OUTFILE=PROJ2SL..RESULT:BVU089:MS/FSA
      DYNALC=CAT ALLOCATE=(SIZE=2000 UNIT=BLOCK)
      OUTDEF=(FILEFORM=BFAS FILEORG=LINKQD);

{call the MAINTAIN_CATALOG processor; store the result in the
subfile RESULT of the PROJ2.SL library; this library is
dynamically allocated on the BVU089 disk with a size of 2000
expressed in blocks unit}
```

❑

8.2 Cataloging Project/User/Billing/Station/Application Information

When the MAINTAIN_CATALOG (or CATMAINT) command is executed from the SYSADMIN project, the System Administrator can use the following commands to list and/or update the project/user/billing/station/application information in the Site Catalog.

Table 8-1. Project/User/Billing/Station/Application Management Commands

	Project	User	Billing	Station	Application
Creating	CRP	CRU	CRB	CRS	
Modifying	MDP	MDU	MDB	MDS	
Listing	LSP	LSU	LSB	LSS	LSA
Deleting	DLP	DLU	DLB	DLS	
Moving	MVP	MVU	MVB	MVS	

Note that the MDU (MODIFY_USER) user command is also an IOF user command.

8.2.1 Things to Remember

- The project is viewed by the Site Catalog as the main entry to which users and billings are linked. Therefore, it is the first object to be created, and also the last object to be deleted.
- A user may have several projects, but only one project as a default project. A user is identified by *project-name.user-name*.
- A project may have several billings, but only one billing as its default billing. A billing is identified by *project-name.billing-name*.
- An application is automatically registered in the Site Catalog when included in the project's application list (either at project creation or at project modification).
- Project, user, billing, and station names must all follow the standard naming convention. Project, user, and billing names are a maximum of 12 characters; station names are a maximum of 8 characters.
- The project OPERATOR must contain the billing INSTALL (OPERATOR/OPERATOR/INSTALL) in the SITE.CATALOG. As an alternative, you can start the system with the "No bill check" option active.

- If a user's default project is to be changed, the default status of this project must first be annulled before giving the default status to another project. For example, the following sequence is required when changing KEVIN's default project from PROJ1 to PROJ2:

```
MDU PROJ1.KEVIN NDFLT;
MDU PROJ2.KEVIN DFLT;
```

  The same principle applies when modifying a project's default billing.

- Character strings must be delimited by single quotes if they contain blank or other non-alphanumeric characters, or if they are empty strings (). For example:

```
PASSWORD='A%%!'
JOBCLASS='()'
```

- Character strings must not begin or end with blank characters.
- Comments may be issued using the command COMM. For example:

```
CRB DEPTI.BILLING;COMM 'BILLING CENTER OF DEPT1';
```

- The star convention can be used when listing, deleting, and moving objects. For example:

`DLU *.BOB`	deletes user BOB from all projects
`LSP *`	lists all users, billings, stations, etc., associated with all projects.
`LSU *.*`	lists all user identities (project-name.user-names) in the site

8.2.2 Initial Cataloging

At system delivery the Site Catalog always contains two project names (SYSADMIN and OPERATOR) and three user names (SYSADMIN, GCOS7, and OPERATOR). Users SYSADMIN and GCOS7 can log-on to the SYSADMIN project (passwords SYS and G7 respectively) and user OPERATOR can log-on to the OPERATOR project (password OP).

There are also projects and users reserved for RMS (Remote Maintenance Service) and TSF (Telecontrol System Facility). These items are described in the manual, *System Installation, Configuration and Updating Guide*.

Use these standard user names and passwords when logging onto the system for the first time. Once alternative names have been created and validated, the standard names should be deleted as soon as possible.

The following is an example of an initial batch update to the Site Catalog:

```
 (1)  $JOB SITE-GEN, USER = SYSADMIN, PROJECT = SYSADMIN,
                  BILLING = SITE-GEN1;
 (2)  CATMAINT COMFILE = *DECK;
 (3)  $INPUT DECK;
 (4)  CRP OPER, MAIN;
 (5)  CRP FIMA;
 (6)  CRU SYSADMIN.VIP, PASSWORD = 'SECRET';
 (7)  CRB SYSADMIN.SITE-GEN2;
 (8)  CRU FIMA.SUPERMAN, NDFLT, PASSWORD = 'ZORRO';
 (9)  CRU FIMA.WONDERWOMAN, PASSWORD = 'WOW!'
 (10) CRU FIMA.WHIZZKID;
 (11) CRB FIMA.BILL1, DFLT;
*(12) CRB FIMA.BILL2;
 (13) CRU SYSADMIN.SUPERMAN, NDFLT;
 (14) VAL;
 (15) $ENDINPUT;
 (16) $ENDJOB;
```

This job creates:

- a MAIN operator project called OPER (4)
- three billings: SITE-GEN2, BILL1 and BILL2 (7, 11 and 12)
- two projects: OPER and FIMA (4 and 5)
- four users: VIP, SUPERMAN, WONDERWOMAN and WHIZZKID (6, 8, 9 and 10)

The SYSADMIN project has SITE-GEN2 as its default billing. The project FIMA has BILL1 as its default billing and BILL2 as an alternative billing.

The user VIP has the password SECRET and default access to project SYSADMIN.

The user SUPERMAN has the password ZORRO and access to projects SYSADMIN and FIMA, but no default project. The user WONDERWOMAN has the password 'WOW!' and default access to project FIMA. The user WHIZZKID has default access to project FIMA, but no password.

* Note that when the CRB command is specified without the DFLT option, the system assumes that the billing is the default billing for the project. In this case, however, because project FIMA already has a default billing (command 11), a warning message is sent. The billing BILL2 is created, but is not designated as the default billing for FIMA.

8.2.3 Subsequent Cataloging

After creating an initial set of projects, users, and billings, the System Administrator should log-on to the SYSADMIN project under a user name created in the previous session and delete the standard names SYSADMIN and GCOS7.

At the same time, or as and when required, the System Administrator can create new projects/users/billings/stations and modify, move, or delete existing ones.

For example, a batch update to the Site Catalog might go as follows:

```
(1)  $JOB SITE-MANLIB, USER = VIP;
(2)  CATMAINT COMFILE = *LIBDECK;
(3)  $INPUT LIBDECK;
(4)  DLU SYSADMIN.SYSADMIN;
(5)  DLU SYSADMIN.GCOS7;
(6)  MDU SYSADMIN.VIP, PASSWORD = 'TOPSECRET';
(7)  DLU *.SUPERMAN;
(8)  DLU FIMA.*;
(9)  DLB FIMA.*;
(10) DLP FIMA;
(11) $ENDINPUT;
(12) $ENDJOB;
```

- The standard user names SYSADMIN and GCOS7 are deleted (4 and 5)
- The System Administrator (VIP) has changed his own password to TOPSECRET (6)
- User SUPERMAN is deleted from all projects (7)
- The FIMA project is deleted, though only after all its users and billings are deleted (8, 9, and 10)

8.3 Validating the Site Catalog

There are three validation commands:

- the command VAL, which activates checks on the Site Catalog
- the command NVAL, which inhibits checks on the Site Catalog
- the command LCS, which lists the present state of the Site Catalog

These deal only with the project/user/billing/station information. All other objects in the Site Catalog may be created and accessed regardless of the validation procedure.

8.3.1 The VAL command

This checks the consistency of the project/user/billing/station in the Site Catalog. It must be executed before access rights are set.

```
VAL    [ { NBILLCHK | BILLCHK } ]

       [ CATALOG = { 0 | 1 | 2 | 3 | 4 | 5 } ] ;
```

NOTES:

1. If BILLCHK is specified, the project/user/billing information is checked. If NBILLCHK is specified, only the project/user information is checked.
2. The VAL command is rejected if there is no user under the SYSADMIN project.
3. The VAL BILLCHK command is executed only if there is at least one user and at least one billing under the SYSADMIN project.
4. The VAL NBILLCHK command is rejected if there is no user under the SYSADMIN project.

8.3.2 The NVAL Command

The NVAL command suspends checking of the project/user/billing/station information. The system continues nonetheless to use this information.

```
NVAL     [ CATALOG = { 0 | 1 | 2 | 3 | 4 | 5 } ] ;
```

CAUTION:
NVAL command on the running SITE.CATALOG is reserved for system maintenance or installation.

While this command is being performed, no new connections to IOF are allowed. To restore SITE.CATALOG validity, you can start a new IOF session on the local system console only if all IOF users are logged off.

8.3.3 The LCS Command

This lists the current state of the Site Catalog.

```
LCS      [ CATALOG = { 0 | 1 | 2 | 3 | 4 | 5 } ] ;
```

8.4 Project Description Commands

8.4.1 Command Syntax

CREATE_PROJECT command

```
CRP   project-name

      [JOBCLASS={(class[,*][,*class][,class*][,**]...)|*[,**]|**}]

      [ { STD | NSTD } ]

      [ { MAIN | NMAIN } ]

      [ { STATION | NSTATION } ]

      [ { RMS | NRMS } ]

      [ APPLIST=(application-name[/tdscode] ...) ]

      [ STTNLIST=(station-name ...) ]

      [ JOBCLASS={(class[,*] ...) | * } ]

      [ DFLTOUTC = output-class ]

      [       {(star-volume-name...)         } ]
      [ MSVOL={                              } ]
      [       {(volume-name[:volume-name]...)} ]

      [ { TAPEVOL } {(star-volume-name...)          } ]
      [ { MTVOL   }={                               } ]
      [ { CTVOL   } {(volume-name[:volume-name]...)} ]

      [ MASTDIR ]

      [ MSTUPB = { SITE | PROJECT | USER | EMPTY } ]

      [ MSTUPI = { SITE | PROJECT | USER | EMPTY } ]

      [ OSTUPB = { PROJECT | USER | EMPTY } ]

      [ OSTUPI = { PROJECT | USER | EMPTY } ]

      [ CATALOG = { 0 | 1 | 2 | 3 | 4 | 5 } ]

      ;
```

MODIFY_PROJECT command

```
MDP   project-name

      [JOBCLASS={(class[,*][,*class][,class*][,**]...)|*[,**]|**}]

      [ { STD | NSTD } ]

      [ { MAIN | NMAIN } ]

      [ { STATION | NSTATION } ]

      [ { RMS | NRMS } ]

      [ { APPLIST=(application-name[/tdscode] ...) } ]
      [ { ADDAPPL=(application-name[/tdscode] ...) } ]
      [ { DELAPPL=(application-name[/tdscode] ...) } ]

      [ { STTNLIST=(station-name ...) } ]
      [ { ADDSTTN=(station-name ...)  } ]
      [ { DELSTTN=(station-name ...)  } ]

      [ JOBCLASS={(class[,*] ...) | * } ]

      [ DFLTOUTC = output-class ]

      [ {       {(star-volume-name...)         }    } ]
      [ { MSVOL={                              }    } ]
      [ {       {(volume-name[:volume-name]...)}    } ]
      [ {                                           } ]
      [ {          {(star-volume-name...)        } } ]
      [ { ADDMSVOL={                             } } ]
      [ {          {(volume-name[:volume-name]...)} } ]

      [ {{ TAPEVOL } {(star-volume-name...)         }    } ]
      [ {{ MTVOL   }={                            }    } ]
      [ {{ CTVOL   } {(volume-name[:volume-name]...)}    } ]
      [ {                                                } ]
      [ {{ ADDTAPEVOL } {(star-volume-name...)         } } ]
      [ {{ ADDMTVOL   }={                              } } ]
      [ {{ ADDCTVOL   } {(volume-name[:volume-name]...)} } ]

      [ MASTDIR ]

      [ MSTUPB = { SITE | PROJECT | USER | EMPTY } ]

      [ MSTUPI = { SITE | PROJECT | USER | EMPTY } ]
```

```
      [ OSTUPB = { PROJECT | USER | EMPTY } ]
      [ OSTUPI = { PROJECT | USER | EMPTY } ]
      [ CATALOG = { 0 | 1 | 2 | 3 | 4 | 5 } ]
      ;
```

LIST_PROJECT command

```
LSP   { * | project-name }
      [ { USER | NUSER } ]
      [ { STATION | NSTATION } ]
      [ CATALOG = { 0 | 1 | 2 | 3 | 4 | 5 } ] ] ;
```

If the star convention is used to assign job classes, the LSP command lists all the job classes individually for projects created or modified before V9 TS9662. It lists simply what has been typed in for projects created or modified in V9 TS9662 and post ones.

DELETE_PROJECT command

```
DLP   { * | project-name }
      [ CATALOG = { 0 | 1 | 2 | 3 | 4 | 5 } ] ;
```

MOVE_PROJECT command

```
MVP   project-name
      [ INCAT = { 0 | 1 | 2 | 3 | 4 | 5 } ]
      [ OUTCAT = { 0 | 1 | 2 | 3 | 4 | 5 } ]
      [ REPLACE ]
      [ NUPGRADE ] ;
```

If the project to be moved is a pre-V5 project and NUPGRADE is not specified, the project is assigned a default IOF job class equal to Q. If Q was previously assigned as the project's default batch job class, then this is forced to P.

8.4.2 Parameters and Constraints

project-name	Simple name of 12 characters maximum
STD	Project is a standard operator project (default value for CRP command).
NSTD	Project is not standard operator project
MAIN	Project is a main console operator project
NMAIN	Project is not a main console operator project (default value for CRP command).
STATION	In CRP/MDP: Project is a station operator project
NSTATION	In CRP/MDP: Project is not a station operator project (default value for CRP command).
RMS	Project is a remote maintenance project (must have MAIN attribute).
NRMS	Project is not a remote maintenance project (default value for CRP command).
APPLIST	List of application names (IOF, tds-name, DJP, UFT, etc.) to which the project has right of access. See "Notes" below. An application name follows catalog standard naming conventions (8 characters maximum). The first application name is the default application of the project. Up to 10 application names may be given per CRP or MDP command. APPLIST=() specified in an MDP command removes the list of applications from the project.
ADDAPPL	List of application names to be added to the existing list. Up to 10 application names may be given per MDP command. If the existing list is empty, the first application specified by ADDAPPL is the default application for the project.
DELAPPL	List of application names to be deleted from the existing list. Up to 10 application names may be given per MDP command.

tdscode	A string of up to 8 hexadecimal characters defining the different transactions the project may access when using the given TDS application. There is no default value.
STTNLIST	List of stations to which the project may be linked. The first station name is its default station. Station name follows catalog standard naming conventions (8 characters maximum). Up to 10 station names may be given per CRP or MDP command. STTNLIST=() specified in an MDP command removes the list of stations from the project. The station list of the project SYSADMIN must be either empty or contain at least the station MAIN. If the name ALL_RSYS (see *UFT User's Guide*) is specified as a station name, it must be specified either alone or second and last in the list. Note also that ALL_RSYS can be specified only in STTNLIST (never in ADDSTTN for example).
ADDSTTN	List of stations to be added to the existing list. Up to 10 station names may be given per MDP command. If the existing list is empty, the first station specified by ADDAPPL is the default station for the project.
DELSTTN	List of stations to be deleted from the existing list. Up to 10 station names may be given per MDP command.
JOBCLASS	List of job classes assigned to the project for job submission. The first class in the list is the project's default Batch job class. The second class in the list is the project's default IOF job class. Up to 26 job class names may be given per CRP or MDP command. A class name consists either of a single character chosen among the 26 alphabetic characters (from A to Z inclusive) and the * character, or of two characters, each of them independently chosen among the 26 alphabetic characters and the * character (from AA to ZZ inclusive, or from A* to Z* inclusive, or from *A to *Z inclusive, or **).

For example:

('KP', *) is equivalent to ('KP', 'Q', 'A',..., 'P', 'R',... 'Y')

('P','**') is equivalent to ('P', 'Q', 'AA', 'AB', ...,"AZ', 'BA', 'BB', ..., 'BZ', ..., 'ZA', 'ZB', ..., 'ZZ')

('P', 'Q', '*', '**') is equivalent to ('P', 'Q', 'A', ..., 'O', 'R', ..., 'Z', 'AA', 'AB', ..., 'AZ', 'BA', 'BB', ..., 'BZ', ..., 'ZA', 'ZB', ..., 'ZZ')

If one or several single character job classes are specified after a *, they will not be taken into account again; the same for the two character job classes specified after a ** job class, because the character * represents 26 single character classes, the character ** represents 676 (26x26) two character classes and these are the maximum that may be given in a command; i.e., 702 (26x27) classes.

JOBCLASS=(AB, *, A, B, **, BP, MX, PI) is equivalent to JOBCLASS=(AB, *, **).

Note: You can never specify explicitly more than 26 different classes in a CRP or MDP command.

Default values for the CRP command are P and Q; i.e. JOBCLASS=('P','Q').

The list of job classes specified in a MDP command replaces completely the existing job class list.

If the job class assigned to a project is a job class group, all classes connected to this job class group are dynamically assigned to the project.

ATTENTION: If the project being modified is a pre-V5 project, it cannot be attributed two-character job classes.

DFLTOUTC	Default Output Class for the project. Output Class is a letter from A to Z inclusive. Futhermore, DFTLOUTC=@ may be used in the MODIFY_PROJECT command to suppress the Output Class previously registered in the catalog at project level. This parameter allows you to select an Output Class for all the Output which the Submitter has belonging to a Project. If the Output class is not explicitly specified, or is not specified in the catalog at project level, the system uses a default Output Class. This default is the Output Class which was specified at Station level, or the value given for MAIN station at CONFIG time. If one or several job classes are specified after a *, they will not be taken into account because * represents 26 classes, and this is the maximum that may be given in a command. JOBCLASS=(AB, * , AC) is equivalent to JOBCLASS=(AB, *).
MSVOL/TAPEVOL	List of volumes to be assigned to the project. MSVOL specifies disk volumes, and TAPEVOL (alias MTVOL or CTVOL) specifies tape and/or cartridge volumes. The volume list is used for matching the media name specified by the project in the VOLPREP utility. The VOLPREP step will not be performed if the match is unsuccessful. Up to 10 volume names, name ranges, or star names can be given per CRP or MDP command. A volume name is a string of up to 6 characters (padding with blanks occurs automatically if necessary). It may contain alphanumeric characters, hyphens (-), and underscores (_). If it contains any other characters it must be protected by quotes; e.g., 'AB+:C;'. An asterisk (*) which is part of a protected string is regarded as an ordinary character, and not as the star character.

A volume range is two volume names separated by a colon (:). The volume range includes all volume names alphabetically comprised between the lower bound (left) and the upper bound (right), both inclusive. The lower bound must be alphabetically less than or equal to the upper bound.

A star name is an unprotected string containing one or more asterisks (*). The * represents any character or sequence of characters, including none. For example, K*14 will match all volume names which begin with K and end with 14. The star convention cannot be used in a volume range (e.g., K5*:K89 will be rejected).

ADDMSVOL/ADDTAPEVOL
: List of volumes to be added to the existing list. ADDMSVOL specifies disk volumes, and ADDTAPEVOL (alias ADDMTVOL or ADDCTVOL) specifies tape and/or cartridge volumes.

MASTDIR
: Indicates that a master directory having the same name as the project name is to be cataloged. The given project will have the OWNER access right on it.

MSTUPB/MSTUPI
: Mandatory startup in batch/interactive mode. Default value is SITE.

OSTUPB/OSTUPI
: Optional startup in batch/interactive mode. Default value is PROJECT.

USER/NUSER
: If NUSER is specified, the list of Users belonging to this project is not displayed. The default value is USER, which displays a list of users.

STATION/NSTATION in LSP:
: If NSTATION is specified, the list of Stations allowed for this project is not displayed. The default value is STATION, which displays the list of stations for the project.

CATALOG/INCAT/OUTCAT
: Ranks in ATTACH of working Site Catalog(s), ranging from 0 to 5. Default value is 0 (current Site Catalog).

REPLACE
: Must be used in the MVP command if the project being moved already exists in the output catalog.

NUPGRADE
: Inhibits the upgrading of a pre-V5 project to a V6 project. If not specified, the upgrade will take place.

NOTES:

1. A user having the STD attribute in his project description may use JCL and GCL freely in interactive mode; i.e., both directives and processor-specific commands. A user having the NSTD attribute can use only directives.
2. A project with the MAIN attribute is automatically assumed to have NSTD, even if STD was specified in the Site Catalog. Several users may simultaneously have the MAIN attribute; they will all receive messages sent to MAIN unless explicit filters are set up.
3. A project with the RMS attribute is automatically assumed to have MAIN and NSTD. Moreover, users of this project are authorized to connect to a system whose SITE.CATALOG has not yet been validated (see the VAL command).
4. A STATION operator is a special IOF user registered in SITE.CATALOG as belonging to a project having the STATION attribute. He is responsible for the first station in the project's station list. If the project has more than one user-id associated with it in the Site Catalog, the first user to log on becomes the station operator and initiates startup for the station. Any other users who later log on to that station become ordinary IOF operators and initiate only their own project's startup file.

   Only one STATION operator may be logged on at a given time on a given station. Only a single mailbox exists for this project.
5. At system delivery time the projects SYSADMIN and OPERATOR exist in the Site Catalog. They may never be deleted (even if the command DLP * is issued).
6. A project can be deleted only if it has no user or billing attached to it, or if it has no protected object associated with it in the Site Catalog (i.e., if all its access rights in the Site Catalog have been removed).
7. If a station specified in a station list does not already exist, it is created with the values of the MAIN station.
8. In both batch and interactive mode, mandatory and optional startups must be different.
9. The MVP command moves a project with all its dependent objects (i.e. users, billings, applications, environments, lists of volumes, stations, and sites).
10. In order that project users can log on to IOF and TDS applications, the project's APPLIST must include the standard name IOF and the relevant tds-names. These are checked irrespective of the value given for the CHKPW parameter of the CONFIG statement SECOPT. Access to other cataloged applications (DJP, UFT, RPMOS, SOW, etc.) is controlled only if CHKPW=YES.

11. To access UFT, DJP, PMOS applications, or to start a GTP (Generalized Transfer Processor) with SOW ... SPOOL command (when CHKPW=YES), the server site project must include the names UFT, DJP, RPMOS or SOW in its APPLIST.

    For further information on security and access to UFT and DJP, see UFT User's Guide and DJP User's Guide.

8.5 User Description Commands

8.5.1 Command Syntax

CREATE_USER command

```
CRU   project-name.user-name

      [ PASSWORD = char12 ]

      [ { DFLT | NDFLT } ]

      [ CATALOG = { 0 | 1 | 2 | 3 | 4 | 5 } ] ;
```

MODIFY_USER command

```
MDU   project-name.user-name

      [ OPSW=char12 ]

      PASSWORD=char12

      [ { DFLT | NDFLT } ]

      [ CATALOG = { 0 | 1 | 2 | 3 | 4 | 5 } ] ;
```

LIST_USER command

```
      { *.*                   }
      { *.user-name           }
LSU   {                       }
      { project-name.*        }
      { project-name.user-name }

      [ { STATION | NSTATION } ]

      [ CATALOG = { 0 | 1 | 2 | 3 | 4 | 5 } ] ;
```

DELETE_USER command

```
      { *.*                    }
      { *.user-name            }
DLU   {                        }
      { project-name.*         }
      { project-name.user-name }

      [ CATALOG = { 0 | 1 | 2 | 3 | 4 | 5 } ] ;
```

MOVE_USER command

```
      { project-name.*         }
MVU   {                        }
      { project-name.user-name }

      [ INCAT = { 0 | 1 | 2 | 3 | 4 | 5 } ]

      [ OUTCAT = { 0 | 1 | 2 | 3 | 4 | 5 } ]

      [ REPLACE ] ;
```

8.5.2 Parameters and Constraints

project-name.user-name	Simple names of 12 characters maximum.
OPSW	The current password to be changed. This parameter is not required if a SYSADMIN user is the requester.
PASSWORD	The new password. This is a string of up to 12 characters. It must be enclosed in single quotes if it contains any blanks or other non-alphanumeric characters, e.g., PASSWORD='%;-*:'. The password must not begin or end with a blank character. The password is an attribute of the user and is not project dependent. If the user has already been given a password, this can be altered using the MDU command. There is no default value.
DFLT	The project is a default project of this user. Default value for the CRU command.
NDFLT	The project is not the default project of this user.

STATION/NSTATION	If NSTATION is specified, the list of Stations allowed for the Project is not displayed. The default value is STATION, which displays the list of stations.
CATALOG/INCAT/OUTCAT	Ranks in ATTACH of working Site Catalog(s), ranging from 0 to 5. Default value is 0 (current Site Catalog).
REPLACE	Must be used in the MVU command if the user being moved already exists in the output catalog.

NOTES:

1. Users in the SYSADMIN project can never be deleted (even if a DLU *.* command is issued).
2. If a user changes his own password, he must ensure that this password is the same on all the systems he may use belonging to the same network.
3. When SECUR'ACCESS (A Bull DPS7 access control product) is used, CREATE_USER, MODIFY_USER and MOVE_USER commands cannot be used on the running SITE.CATALOG. Instead use the special SECUR'ACCESS commands for this purpose.

8.6 Billing Description Commands

8.6.1 Command Syntax

CREATE_BILLING command

```
CRB   project-name.billing-name

      [ CREDIT=dec8 ]

      [ { DFLT | NDFLT } ]

      [ CATALOG = { 0 | 1 | 2 | 3 | 4 | 5 } ] ;
```

MODIFY_BILLING command

```
MDB   project-name.billing-name

      [ CREDIT=dec8 ]

      [ { DFLT | NDFLT } ]

      [ CATALOG = { 0 | 1 | 2 | 3 | 4 | 5 } ] ;
```

LIST_BILLING command

```
      { *.*                      }
      { *.billing-name           }
LSB   {                          }
      { project-name.*           }
      { project-name.billing-name }

      [ { STATION | NSTATION } ]

      [ CATALOG = { 0 | 1 | 2 | 3 | 4 | 5 } ] ;
```

DELETE_BILLING command

```
      { *.*                          }
      { *.billing-name               }
DLB   {                              }
      { project-name.*               }
      { project-name.billing-name    }

      [ CATALOG = { 0 | 1 | 2 | 3 | 4 | 5 } ] ;
```

MOVE_BILLING command

```
      { project-name.*               }
MVB   {                              }
      { project-name.billing-name    }

      [ INCAT = { 0 | 1 | 2 | 3 | 4 | 5 } ]

      [ OUTCAT = { 0 | 1 | 2 | 3 | 4 | 5 } ]

      [ REPLACE ] ;
```

8.6.2 Parameters and Constraints

project-name.billing-name	Simple names of 12 characters maximum.
CREDIT	Maximum CREDIT allowed to the billing in number of billing units. Default value is 99 999 999.
DFLT	This billing is the default billing of the project. Default value in CRB command.
NDFLT	This billing is not the default billing of the project.
STATION/NSTATION	If NSTATION is specified, the list of stations allowed for the Project Billing is not displayed. The Default value is STATION, which displays the list.
CATALOG/INCAT/OUTCAT	Ranks in ATTACH of working Site Catalog(s), ranging from 0 to 5. Default value is 0 (current Site Catalog).

REPLACE	Must be used in the MVB command if the billing being moved already exists in the output catalog.

NOTE:

Billings belonging to the SYSADMIN project can never be deleted (even if a DLB *.* command is issued).

8.7 Station Description Commands

8.7.1 Command Syntax

CREATE_STATION command

```
CRS   station-name

      [ DFLTOUTC = output-class ]

      [ DFLTPRTY = (output-class/priority ...) ]

      [ PRTDFDVC = device-class ]

      [ FBANNER = { 0 | 1 | 2 } ]

      [ EBANNER = { 0 | 1 } ]

      [ BANCHAR = char2 ]

      [ SITELIST = (site-name ...) ];

      [ CATALOG = { 0 | 1 | 2 | 3 | 4 | 5 } ] ;
```

MODIFY_STATION command

```
MDS   station-name

      [ DFLTOUTC= output-class ]

      [ DFLTPRTY = (output-class/priority ...) ]

      [ PRTDFDVC = device-class ]

      [ FBANNER = { 0 | 1 | 2 } ]

      [ EBANNER = { 0 | 1 } ]

      [ BANCHAR = char2 ]

      [ { SITELIST }                    ]
      [ { ADDSITE  } = (site-name ...) ]
      [ { DELSITE  }                    ]

      [ CATALOG = { 0 | 1 | 2 | 3 | 4 | 5 } ] ;
```

LIST_STATION command

```
LSS   { * | station-name }
      [ CATALOG = { 0 | 1 | 2 | 3 | 4 | 5 } ] ;
```

DELETE_STATION command

```
DLS   { * | station-name }
      [ CATALOG = { 0 | 1 | 2 | 3 | 4 | 5 } ] ;
```

MOVE_STATION command

```
MVS   station-name
      [ INCAT = { 0 | 1 | 2 | 3 | 4 | 5 } ]
      [ OUTCAT = { 0 | 1 | 2 | 3 | 4 | 5 } ]
      [ REPLACE ] ;
```

8.7.2 Parameters and Constraints

station-name	Simple name of 8 characters maximum. Use the conventional name ALL_RSYS (All Remote systems) if you want to designate all the remote system names not registered in the catalog.
DFLTOUTC	This character gives the default output class of deliveries of jobs received by the printer of the station. Default value is that given for MAIN station at CONFIG time.
DFLTPRTY	This parameter gives the default priority to be attached to each output class. Output-class is a letter from "A" to "Z" inclusive, priority is from 1 to 7. Default values are those declared for the MAIN station at CONFIG time. Note also that the media for the output classes are the same as those of the MAIN station.
PRTDFDVC	Describes the printer default device class. The default value is that given for the MAIN station at CONFIG time.

FBANNER	Gives the number of front banners to be printed at beginning of the deliveries. Permitted values are 0 (no banners), 1 (1 banner), or 2 (2 identical banners). The default value is that given for the MAIN station at CONFIG time.
EBANNER	Gives the number of end banners to be printed at the end of the deliveries. Permitted values are 0 (no banners) or 1 (1 banner). The default value is that given for MAIN station at CONFIG time.
BANCHAR	Gives the 2 characters to be used when printing the banners. Special characters (blanks included) must be protected by quotes (e.g., ':;'). The default value is that given for the MAIN station at CONFIG time.
SITELIST	List of sites to which the station may be mapped. The first site name is its default site. Site name follows the catalog standard naming conventions (8 characters maximum). Up to 10 site names may be given per CRS or MDS command. If this parameter is missing in a CRS command, the station will be given a unique default site, with the name as the station (Except the ALL_RSYS station, which uses MAIN as the default site). If the name ALL_RSYS (see *UFT User's Guide*) is specified as a site name, it must be specified either alone or second *and* last in the list. Note also that ALL_RSYS can be specified only in SITELIST (never in ADDSITE for example).
ADDSITE	List of sites to be added to the existing list. Up to 10 site names may be given per MDS command.
DELSITE	List of sites to be deleted from the existing list. Up to 10 site names may be given per MDS command.
CATALOG/INCAT/OUTCAT	Ranks in ATTACH of working Site Catalog(s), ranging from 0 to 5. Default value is 0 (current Site Catalog).
REPLACE	Must be used in the MVS command if the station being moved already exists in the output catalog.

NOTES:

1. A station can be created in two ways:

   The recommended method is to enter the station description using the CRS command, and then specify it's name in the station list of the project concerned using the CRP or MDP commands.

   Alternatively, you can use the CRP or MDP commands first to establish the name of the station in a project's station list, and then use the CRS command to create the station. If you do not use the CRS command, the station is automatically created with the values of a MAIN station.

2. The first site name specified in parameter ADDSITE will become the default site of the station if its site list is empty.

3. SITELIST=() specified in a MDS command removes the list of sites from the station.

4. The mapping of a station to a site is done at log on time. If the site list of a station is empty, the station will be mapped to the site with the same name as the station.

5. A CRS or MDS command issued on the MAIN station can include only the parameter SITELIST or ADDSITE. These parameters must be specified in such a way that the resulting site list of the MAIN station either becomes empty or has the site MAIN as its default site.

6. Once created, the MAIN station can never be deleted (even if DLS * is issued).

7. Using the DLS command makes the station obsolete; it is still known, however, and is deleted only at the next cold or clean restart.

8. When using the DJP facility, we recommend that you declare a station with the name HOSTID, as this will attach it to all projects using DJP.

8.8 Application Description Commands

The only application description command available is the LIST_APPLICATION (LSA) command. This displays which projects contain the specified application in their application list (as defined by the APPLIST parameter of the CRP or MDP commands).

8.8.1 Command Syntax

LIST_APPLICATION command

```
LSA  NAME = application-name
       [ CATALOG = { 0 | 1 | 2 | 3 | 4 | 5} ] ;
```

8.8.2 Parameters and constraints

NAME — An application name has the following format:

```
<simple-name>[/<mask>]
```

A simple name is a string of up to 8 characters maximum.

If the application is a TDS application then its simple name (8 characters maximum) may be followed by a slash (/) and a mask (8 hexadecimal digits maximum and right-padded with zeros). This mask is interpreted as a set of authority codes (see the *TDS Administrator's Guide*). It enables you to list only projects whose tdscode contains at least one authority code matching an authority code given by the mask.

CATALOG — This specifies the rank, in the ATTACH_CATALOG command, of the Site Catalog on which the command is to operate. It ranges from 0 to 5, with 0 the default value (running Site Catalog).

EXAMPLES:

```
LSA IOF;
```

Lists all projects that have IOF in their application list.

```
LSA TDS1;
```

Lists all projects (and their tdscode) having TDS1 in their application list.

```
LSA TDS1/82;
```

Lists only the projects (and their tdscode) which have TDS1 in their application list with at least the authority code 0 or the authority code 6.

❑

NOTE:

A mask set to "00000000"X is the same as no mask. This means that the three following commands are equivalent:

```
LSA TDS1/00000000;
LSA TDS1/FFFFFFFF;
LSA TDS1;
```

8.8.3 Error Messages

The error messages are the standard ones in MAINTAIN_CATALOG (warnings or fatal error messages about syntax or semantic).

8.9 Types of Error Message

Three types of error messages are produced with MAINTAIN_CATALOG:

- Syntax Error Messages
- Warning Messages (*)
- Abort Messages (***)

Asterisks indicating the severity of errors are printed with warning and abort messages:

*	= execution continues (warning)
***	= execution stops (abort)

9. Environment Management Procedures

This chapter describes the standard GCL command environments delivered with GCOS 7. It explains the procedures for adapting these environments and creating new ones.

9.1 Overview

9.1.1 Environment Structure

The contents of each environment may be seen as the set of relationships between the environment object itself, as stored in the SITE.CATALOG, and the commands stored in the system library SYS.HBINLIB or user libraries. Defining an environment consists of listing which commands may be accessed from the environment and, among them, which ones are to be presented on the menu screens. This definition may be done for each existing domain.

In order to avoid excessively long definitions for each environment, and also to make updating the environments easier, the commands are grouped into families of commands.

In the sample structure below, environment E1 gives access to the command families 1 and 3. Family 1 contains the commands 1, 2, and 3. Family 3 contains the command 4. Therefore a user operating under environment E1 is granted access to commands 1, 2, 3, and 4.

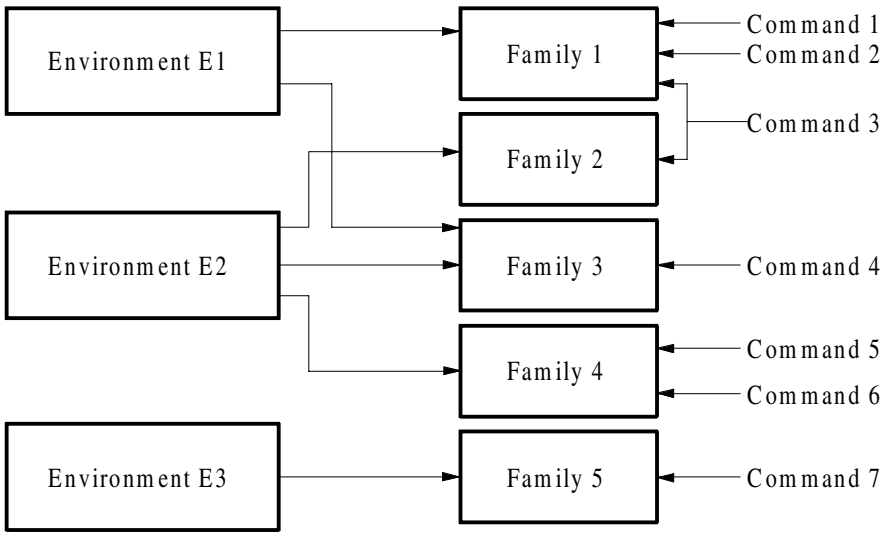

Figure 9-1. Example of an Environment Structure

9.1.2 Command Accessibility and Visibility

Each environment gives access to one or more families. There may be up to 256 families of commands. They are numbered from 1 to 256. Each command is declared as belonging to one or more families and as being presented on the menu screens or hidden. The setting of access rights is done by means of the ACCESS and HIDE parameters of the PROC command. Families do not have the concrete representation in the system (either in catalog structures or in the binary libraries) that the environments and commands have. Their only purpose is to help the System Administrator to create and modify the environments.

There is, therefore, no command for defining families. Their implicit definition results from both the creation of the environments and the setting of access rights on the commands. This is shown by the directions of the arrows in Figure 9-1.

An environment created in the SITE.CATALOG may be viewed as a mask of 256 bits, each of which controls the access to a specific family of commands numbered from 1 to 256.

Each procedure definition contains an access mask of 256 bits (specified by the ACCESS parameter) which controls the membership of each family, and a hide mask also of 256 bits (specified by the HIDE parameter) which controls the presence on the menu screens for each family for which the procedure is accessible (HIDE is ignored in a procedure where ACCESS=0).

The following example demonstrates the rules for controlling the accessibility and visibility of the GCL commands through the environment mechanism.

EXAMPLE:

The declaration of command A is:

```
PROC      NAME = A
          ...
          ACCESS = (j,k)
          HIDE = ([i,]j)
          ...;
```

❑

The result of this declaration is the creation of a command whose access and hide masks are as shown in Figure 9-2 below.

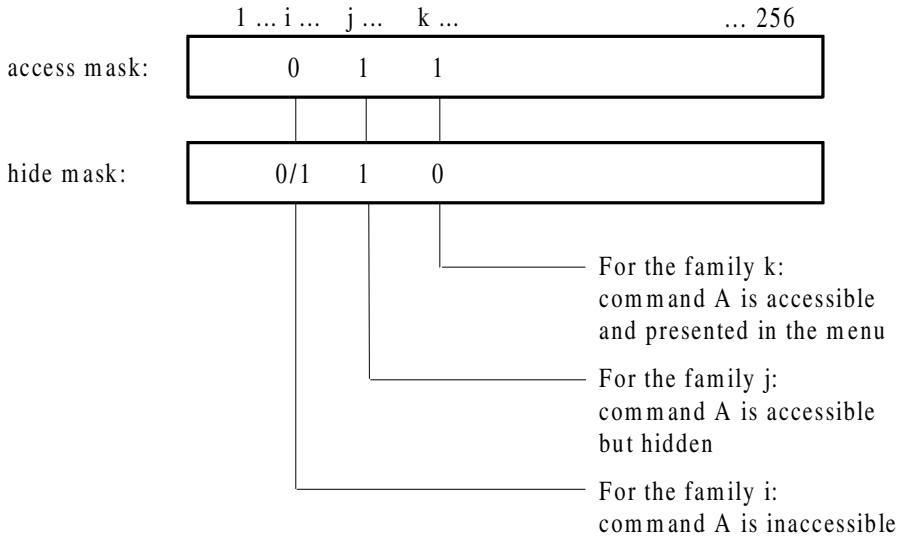

Figure 9-2. Access and Hide Masks of a Command

The behavior of a command in a specified user environment is summarized by the following:

- If the environment grants access to a family, and that family can access a visible command, then the environment also has access to this visible command (access mask = 1, hide mask = 0).
- If the environment grants access to a family, and that family can access a hidden command, then the environment also has access to this hidden command (access mask = 1, hide mask = 1).
- In the previous example, the command A is hidden in an environment which grants access to family j but not k.
- If the environment does not grant access to a family, then it cannot access any command (visible or hidden) to which the family has access (access mask = 1, hide mask = 0 or 1).
- In the previous example, the command A is inaccessible in an environment which grants access to neither family j nor family k.

Environments are not the only means of controlling the accessibility of a command. The operator rights associated with the user's project also affect project accessibility and visibility of the GCL commands. This is discussed later in this section.

Commands that are associated with applications that have not been installed are not accessible.

9.1.3 Modifying Command Accessibility/Visibility

The access rights on a command consist of the ability to use that command from a given environment, and also to make it visible or hidden when menus are used.

9.1.3.1 User Commands

Initially, the access rights are defined at the same time as the GCL command itself, using the ACCESS and HIDE parameters of the PROC statement. This is the normal method when creating user commands which are generally stored in private binary libraries that the user inserts in his own binary search path. This is also the method used by the System Administrator when creating site range commands in the system library SYS.HBINLIB. To set the appropriate access rights on his own private commands, the user has to apply to the System Administrator to be given the numbers of the families that he can use for his own purpose.

9.1.3.2 Standard Commands

Unlike the user commands, special means for setting the access rights on the standard commands (that is, those that are delivered with the system in the system library SYS.HBINLIB) are provided exclusively for the System Administrator. These standard commands cannot be modified, not even by the System Administrator, and their contents may not be printed. The only noteworthy exception is for modifying or printing certain superficial parameters of the command, including their access and visibility attributes.

The MODIFY_ACCESS (MDA) command of the MAINTAIN_COMMAND processor allows the System Administrator to redefine the access rights of the standard commands.

The full format of this command is given in the *IOF Terminal User's Reference Manual*. In terms of setting access and hide masks, the format is usually as follows:

```
MODIFY_ACCESS     PROC = { star62 | (name31 ...) }

                  ....

                  [ ACCESS = (nnn[-nnn]...) ]

                  [ ACCESS_MODE = { ADD | DL | RPL } ]

                  [ HIDE = (nnn[-nnn]...) ]

                  [ HIDE_MODE = { ADD | DL | RPL } ]

                  ....
```

The ACCESS parameter is a list of families to be added to or deleted from the procedure, or which replaces the current list in the procedure, depending on the value specified by ACCESS_MODE. By default the list is deleted.

The HIDE parameter is a list of families for which the procedure is to be hidden or no longer hidden, or which is to replace the current list in the procedure, depending on the value of HIDE_MODE. The default value (ADD) means the procedure is to be hidden.

The effects of successive MODIFY_ACCESS commands are cumulative. This is shown with the following example in which the contents of the access mask and the hide mask (described above) are printed.

NOTE:

The MODIFY_ACCESS command inserts the specified modifications into the SYS.HBINLIB file, but these modifications do not become effective until the next ISL, at which time they are copied into the SYS.SPOOL file. Consequently, any redefinition of access rights must be validated by:

- either a RESTART CLEAN plus RESTORE

  or

- any kind of restart followed by the loading of H_SM13 or the use of the SPOOL command at ISL.

EXAMPLE:

```
Masks          Effects

PROC A                    1 1 1 1 1 ...   A is accessible
   ACCESS=(1,2,3,4,5);    0 0 0 0 0 ...   and visible for
                                          families 1 to 5

MDA  PROC=A               1 0 0 1 1 ...    A is
     ACCESS=(2,3)         0 0 0 0 1 ...    - visible for
     HIDE=5;                                  families 1, 4
                                           - hidden for
                                              family 5
                                           - inaccessible
                                              for families
                                              2 and 3
MDA  PROC=A               1 0 0 1 1 ...    A is
     HIDE=4;              0 0 0 1 1 ...    - visible for
                                              family 1
                                           - hidden for
                                              families
                                              4 and 5
                                           - inaccessible
                                              for families
                                              2 and 3
MDA  PROC=A               1 0 0 1 1 ...    A is  as above
     HIDE=3;              0 0 1 1 1 ...    HIDE is ignored
```

❑

The System Administrator is also provided with a command which lists all the parameters of the selected procedures that may be modified by the MODIFY_ACCESS command, namely: access rights, operator rights and priorities. This command is the LIST_ACCESS command; its format is:

```
LIST_ACCESS
                              { name31 ... }
                 PROCEDURES = { star62     }
                              { *          }

                [ DOMAINS = star62 ] ;
```

Listing Current Status

The LIST_ACCESS command displays an array that indicates the current status of the commands for each family from 1 to 256. A plus sign (+) means that the procedure is accessible and presented on the menu screens; a minus sign (-) means that it is accessible but hidden; a blank character means that the command is not accessible.

In the previous example, a LIST_ACCESS command following each MDA command would have displayed the new access rights of the command A, as described on the right-hand column of the previous example.

The effect of the MODIFY_ACCESS command is to patch into the specified procedures the given restrictions on the previous access rights.

Resetting Inital Status

The RESET command permits the System Administrator to reset the original access rights (and priorities) of the specified commands. The format of the RESET command is:

```
RESET     PROCEDURES = { name31 ...| star62 }

          [ DOMAIN = name31 ] ;
```

This command cancels the effects of all previous MODIFY_ACCESS commands and therefore restores the original access rights (and priorities) of the commands in the standard library SYS.HBINLIB.

The RESET command should be used systematically in any startup sequence for installing environments, and also used prior to any access right setting.

The use of the MODIFY_ACCESS command by the System Administrator to specify restrictions on access rights is not dangerous, as the RESET command permits him to restore their initial values.

9.1.4 Creating and Deleting Environments

The ENVT command of the MAINTAIN_COMMAND processor creates or modifies an environment in the SITE.CATALOG. It can be used only by the System Administrator.

The format of the command is:

```
ENVT    ENVIRONMENT = name12

        FAMILIES = (dec3[-dec3] ... )

        [ MODE = { CR | ADD | DL } ] ;
```

The FAMILIES parameter specifies the list of families to which the environment is to be granted access. In the MODE parameter, CR (the default) creates or recreates the environment with the specified family numbers, ADD adds the specified family numbers to the current definition of the environment, and DL deletes the specified family numbers from the current definition of the environment.

The DELETE_ENVT command of the MAINTAIN_COMMAND processor deletes an existing environment. It can be used only by the System Administrator.

The format of the command is:

```
DLENVT  ENVIRONMENT = name12 ;
```

The LIST_ENVT command of the MAINTAIN_COMMAND processor displays the list of the families to which the specified environment(s) have access.

The format of this command is:

```
LSENVT  ENVIRONMENTS = star24 ;
```

9.1.5 Linking Environments to Projects

The PROJ command of the MAINTAIN_COMMAND processor sets the relationships between the projects and the environments. It can be used only by the System Administrator. The format of this command is:

```
PROJ        PROJECT = name12

            ENVIRONMENTS = (name12 [name12] ... )

            [ MODE = { CR | ADD | DL } ] ;
```

The first name in the list is used as the default environment for the project.

If you specify ENVIRONMENTS=0, all the relationships between the given project and its previously attached environments are deleted, and the project is granted access to the unnamed environment 0 which grants access to all commands.

The environments must have been created (using the ENVT command) before attaching them to the projects (using the PROJ command).

The LIST_PROJ command of the MAINTAIN_COMMAND processor lists all the relationships between the projects and the environments.

The format of this command is:

```
LSPROJ       PROJECTS = star24 ;
```

All the environments that are attached to the selected project(s) are listed.

9.1.6 Operator Rights and Environments

Three types of operator are recognized by the GCL command language:

- the MAIN operator (value 1)
- the IOF operator (value 5)
- the STATION operator (value 6)

Specific commands are provided for the user who has a specific operator right. These commands are provided in addition to the commands that are normally accessible in his working environment.

For example, a user that is a station operator, may both have the role of a station operator and develop programs within the PROGRAM_PREP environment, without having to switch from one environment to another.

In the following description of the standard environments, the user is assumed to be an IOF operator (and also to have the GCL right, value 4). For each environment the described screen presentation and list of hidden commands apply to this category of user.

For a specific operator (station operator), all the commands that he has to be granted access to are accessible and presented on the first screens of his menus, regardless of the environment he is working in. The commands are not sorted using the environment mechanism, but via the operator access rights and priority defined by the GCL management.

Thus, the station operator working in the PROGRAM_PREP environment will be:

- presented with the commands of the PROGRAM_PREP environment (see below).
- granted access to the hidden commands of the PROGRAM_PREP environment (again, see below) listed in the following paragraph
- granted access to all the specific station control directives which will be presented in the first screens of his menu.

This is performed by gathering all the station operator directives in the standard family 2, which is made accessible from all the standard environments.

The station operator commands have been assigned priority 15.

The standard presentation of the menu of a user who is a station operator and works in PROGRAM_PREP is shown below:

```
       PROGRAM_PREP
------------------------------
    1 FORCE_OUTPUT
    2 FORCE_USER_REQ
    3 MODIFY_CONFIGURATION OC
    4 MODIFY_STATION RESCUE_OFF     station operator commands
    5 MODIFY_STATION RESCUE_ON      (PRTY 15)
    6 START_INPUT_READER
    7 START_OUTPUT_WRITER
    8 TERMINATE_INPUT_READER
    9 TERMINATE_OUTPUT_WRITER
------------------------------
2/4 1 PREPARE_PROGRAM_BATCH
    2 PREPARE_PROGRAM_INT
                                    (PRTY=50)
    3 COBOL

       .
       .
       .
       etc.     (refer to PROGRAM_PREP visibility)
```

The means for defining a specific operator environment are described in Appendix B.

9.1.6.1 Setting Operator Rights

The operator rights are set at procedure creation using the OPACC and OPHID parameters of the PROC statement. As for families, these two options may be represented as two 8-bit masks (one access mask and one hide mask). For example, specifying:

- OPACC = (5,6)
- OPHID = 5

in a PROC statement means that the command is accessible only for a user that works under a project which has operator rights 5 and/or 6, that is, either a station operator or an IOF operator (or both). The command is hidden, however, for a user who is an IOF operator only.

Remember that these access controls apply only after the ones associated with the families. In the above example, the user cannot access the command if he works under an environment which does not give access to the command, even if he is a station operator. If he works under an environment for which the command is hidden, he can access it if he is a station operator or an IOF operator, however, the command is not presented on the menu screens.

The default operator rights for the standard commands are given below (OPHID is always 0, that means the command is not hidden).

9.1.6.2 Domain IOF

User Category	OPACC	Commands
GCL or IOF operator	(4,5)	MODIFY_OUTPUT_PARAMETERS MPRTLIB MWENVT MWINLIB MWLIB
	4	the other commands

9.1.6.3 Domain H_NOCTX

User Category	OPACC	Commands
Main operator or Station operator	(1,6)	FORCE_OUTPUT FORCE_USER_REQ MODIFY_CONFIGURATION MODIFY_STATION MODIFY_INPUT_READER START_OUTPUT_WRITER TERMINATE_INPUT_READER TERMINATE_OUTPUT_WRITER
Main or IOF or Station operator	(1,5,6)	DISPLAY_ERROR_LOG DISPLAY_REQUEST
IOF operator	5	the other commands

9.1.6.4 Domain MAIN

User Category	OPACC	Commands
Main operator	1	all commands except network commands (operator rights attached to network commands cannot be modified via MDA)

9.1.6.5 Other Domains

User Category	OPACC	Commands
GCL	4	all commands

9.1.7 Command Priorities

The default presentation of the menu screens is a list of commands in alphabetical order. A maximum of 18 commands can be presented on each screen.

For a domain that contains a lot of commands (such as IOF, H_NOCTX, or MAINTAIN_LIBRARY) this default presentation does not provide easy command selection. One might prefer, for example, to have the more frequently used commands gathered on the first screen, and the more specialized commands presented on the following screens.

This can be done with the basic mechanism of priorities. Each GCL command may be assigned a value from 0 to 255, named PRTY, which controls the position of the command on the menu screens. Commands which have lower numbers (higher priorities) are presented before commands which have higher numbers (lower priorities). Commands with the same priority are presented in alphabetical order.

A blank line is inserted in the menu each time the priority jumps to or over the next multiple of 5. For example, blank lines are inserted between commands of priorities 2 and 5, 2 and 9, 9 and 10, but not between priorities 5 and 9.

A page skip occurs each time the priority jumps to or over the next multiple of 50. For example, a page skip occurs between commands of priorities 2 and 52, 49 and 50, 49 and 250, but not between priorities 50 and 99.

The following example shows the presentation of a domain that contains commands with different priority values (notation Pi, Qi and Ri identifies commands with a priority equal to i).

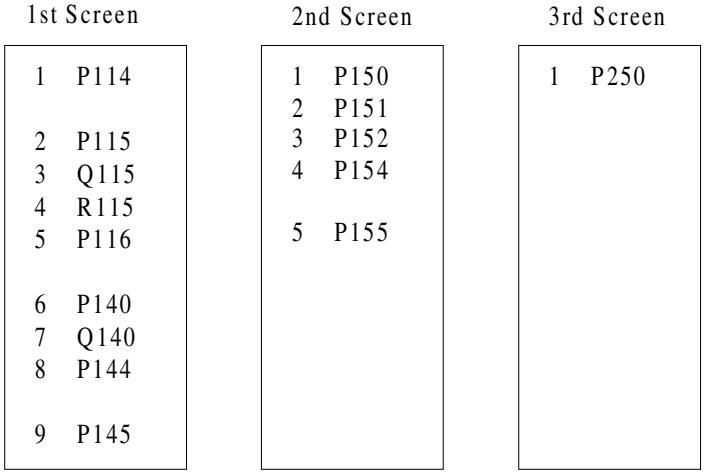

Figure 9-3. Command Priorities (Example)

The priorities given to a command are, unlike access rights, not dependent on the families.

The initial value of the priority is set using the PRTY parameter of the PROC statement. The default value is zero.

The System Administrator can use the MODIFY_ACCESS command of the MAINTAIN_COMMAND processor (PRTY parameter) to change the priority of the commands stored in the system library SYS.HBINLIB.

The LIST_ACCESS command lists the current priority value.

The RESET command restores the initial priority.

All these commands are fully described in *IOF Terminal User's Reference Manual*.

The initial priority values for the standard commands are listed below:

- The normal value of the priority for all standard commands is 50. Exceptions are given below.
- In the H_NOCTX and MAIN domains (directives): all the directives are given the priority 100, with the exception of TUTORIAL, which is given priority 200.

9.2 Standard Environments

IMPORTANT:
The standard environments below are meant to be used as general guides and not for a particular case.

9.2.1 Presentation

Seven standard environments are provided. They are described below.

9.2.1.1 PROGRAM_PREP Environment

The PROGRAM_PREP environment is adapted to users who develop COBOL programs interactively. A selection of the most commonly used commands is made in order to reduce the number of menu screens to three. These screens contain mainly the commands to edit, compile, link, and debug COBOL programs, and to control jobs and outputs.

Other commands are accessible, but hidden in this main environment. They may be called from a subenvironment (secondary environment) of PROGRAM_PREP named BG_PROG_PREP. You can switch to this secondary environment by using the BACKGROUND_PROGRAM_PREP command. This secondary environment presents on the menu screens a set of complementary tools such as MAINTAIN_COMMAND, MAINTAIN_FORM and some data management utilities.

Two special commands are integrated in the PROGRAM_PREP environment. These commands edit, compile, link and execute a set of programs. One command (PREPARE_PROGRAM_INT) links that sequence of operations in interactive mode, the other (PREPARE_PROGRAM_BATCH) in batch mode.

9.2.1.2 TDSAPPL_PREP Environment

The TDSAPPL_PREP (TAP) environment is designed to enable the user to develop and test COBOL TPRs in batch or interactive mode.

In order to give uniform visibility to the user of the standard environments, most of the elements concerning the development of TPRs as such are shared with PROGRAM_PREP.

As far as testing the TPRs is concerned, the main objective of TAP is to conceal as much as possible from the user those system elements specific to the TDS monitor, in particular the integration of a newly generated TPR into a TDS in execution.

To achieve this, information must be made available to the different TDS users, giving the status of the monitor (active or not) and the search path used to localize the TPRs. Only one file is used for passing this information.

The commands used to run the utilities associated with TDS application preparation are provided in the subenvironment BG_TDS_PREP. The BACKGROUND_TDSAPPL_PREP command is used to switch to this subenvironment. A return to the TDSAPPL_PREP environment can be made with the RETURN_TDSAPPL_PREP command.

9.2.1.3 PROJ_MANAGER Environment

The PROJ_MANAGER environment presents on four menu screens the main commands to handle files attached to a project, operate on directories, manage libraries, define new GCL commands, and create documentation.

Other functions can be performed in line mode or using the two lines at the bottom of the menu screen, by entering hidden commands. A list of the hidden commands may be obtained by using the BACKGROUND_PROJ_MANAGER command.

9.2.1.4 FILE_VOLUME Environment

The FILE_VOLUME environment makes available all the commands relating to files, catalogs and volumes that are not reserved exclusively for the System Administrator. These commands may be used to create, delete, modify and maintain files, catalogs and volumes. A selection of the most frequently used commands is presented on three menu screens.

The whole list of commands relating to files, catalogs and volumes may be obtained by using three different background environments named BG_FILE, BG_CATALOG, and BG_VOLUME. All the commands available under these background environments are also accessible directly in the FILE_VOLUME environment by entering them in line mode (hidden commands).

9.2.1.5 SYSADMIN Environment

The SYSADMIN environment makes available all the commands that exist in the system library SYS.HBINLIB. A selection of the most frequently used commands is presented on four screens. These commands are used to define users, projects and billings in the SITE.CATALOG, to install the system, to tailor the system files, and to create documentation.

The System Administrator may enter hidden commands on the last two lines of the menu screens, or by invoking the special background environments BG_FILE_, BG_CATALOG_, and BG_VOLUME_ as in the FILE_VOLUME environment.

9.2.1.6 MAIN_FULL Environment

The MAIN_FULL environment provides the MAIN operator full access to all commands stored in the standard library SYS.HBINLIB.

9.2.1.7 MAIN_REDUCED Environment

The MAIN_REDUCED environment provides the MAIN operator all the commands needed for day-to-day operation of the system, with the exception of those commands which modify system-wide parameters. The command MODIFY_DIMENSION, for example, is not accessible from this environment.

9.2.2 Standard Environment Components

Together with the environment object itself, other components are installed in the SITE.CATALOG or system libraries to provide a consistent entity. This section describes briefly the different components that are or may be part of a standard environment.

9.2.2.1 Environment Object

The environment object itself must be defined and stored in the SITE.CATALOG, as described previously in this section, using the ENVT command of the MAINTAIN_COMMAND processor. This command specifies the name of the environment and defines a list of families to which the environment grants access.

9.2.2.2 Access Rights and Priorities

The access rights and priorities of the standard commands must be implemented in the system library SYS.HBINLIB using the MODIFY_ACCESS command of the MAINTAIN_COMMAND processor. These access rights, applied to the whole set of commands, define implicitly the standard families, as described above.

9.2.2.3 Special Commands

Special commands, called non-standard commands (because they are not part of the SYS.HBINLIB delivery) may be created to make the set of commands presented to the user conform to the needs of the environment user's class. Such commands are used in the PROGRAM_PREP environment.

9.2.2.4 Help Texts Associated with the Environments

Help texts that are associated with the environments have to be provided. Three levels of Help text exist. They are:

- **domain level** (the help text is displayed by entering a question mark in the select field of the menu)
- **command level**
- **parameter level**

If the current domain is IOF (identified by an "S:" prompt in line mode), the Help text which is displayed at domain level is that of the current environment, if it exists. Thus creating a new environment means that associated help texts must be written.

9.2.2.5 Help Texts Associated with Non-standard Procedures

Help texts associated with non-standard procedures also have to be provided, both at command level and at parameter level.

For details about Help text generation, refer to the *IOF Programmer's Manual*.

9.2.2.6 Source Subfiles

Source subfiles may be needed. Some complex mechanisms, such as synchronization between a batch job and the interactive session that submitted it, need source subfiles to be associated with GCL procedures. Several such subfiles are needed with the commands PREPARE_PROGRAM_INT and PREPARE_PROGRAM_BATCH of the PROGRAM_PREP environment, and PREPARE_TPR_INT and PREPARE_TPR_BATCH of the TDSAPPL_PREP environment. They contain JCL sequences invoked by the EJR command, and commands in source format invoked from COMFILE options or the ALTER_INPUT directive.

9.2.2.7 Component Generation

All these components (environment objects, access rights and priorities, special procedures, Help texts, and source subfiles) are generated in the appropriate system files (SITE.CATALOG, SYS.HBINLIB, SITE.HELP, and SYS.HSLLIB) by using a unique JCL sequence submitted in batch.

This JCL sequence and its use is described later in this section.

This JCL must be maintained by the System Administrator, who may add new components of the same type as those listed above (environment objects, source subfiles, etc.) and the specific commands necessary for the site (such as PROJ for setting the relationships between projects and environments).

9.2.3 Standard Families

The definitions of standard environments require definitions of standard families.

Ninety-nine standard families have been reserved for the definition of the standard environments. The families (numbered 100 to 256) may be either used by the System Administrator, for defining new environments, as described later, or assigned to users for their own needs. In both cases, the System Administrator is the only person in charge of the use of family numbers.

The complete list of standard families is given in Appendix A. Each family is assigned a number, a symbolic name (which is optional), and two lists of commands, one list containing the visible commands for the family and the other the hidden commands.

The same family may contain commands that belong to different domains, and a command may belong to one or more families.

The standard families, together with other non-standard families, are the items with which the System Administrator can build additional environments.

Examples of the use of standard and users' families are given in Appendix B.

9.2.4 Standard Command Priorities

As described above, all the commands are assigned a priority that controls their position on the menu screens. There is one priority value only for each command (not dependent on the families, as for accessibility or visibility).

As for access rights, priorities are assigned to commands of any domain. But, for the domains that do not contain a lot of commands, the initial priorities of the commands are not modified, because the alphabetical order generally is the best. Alphabetical order is the default for commands of the same priority.

The largest domains are IOF (system-level commands), H_NOCTX (directives), and the MAINTAIN_LIBRARY domains. For the other domains, the initial priorities (always 50, except for the TAILOR domain) are not changed in the standard environments generation.

9.2.4.1 Domain IOF

Commands not appearing in the following list have a priority of zero.

Priority	Command
50	EDIT
	MAINTAIN_TDSGEN
	PREPARE_PROGRAM_BATCH
	PREPARE_PROGRAM_INT
	PREPARE_TPR_BATCH
	PREPARE_TPR_INT
51	BUILD_SYSTEM
	MAINTAIN_CATALOG
	MAINTAIN_MIGRATION
	MAINTAIN_QUOTA
55	BIND_CU
	COBOL
	EXEC_PG
	FSE
	LINK_PG

```
56    |  BUILD_LIBRARY
      |  CLEAR_LIBRARY
      |  DELETE_LIBRARY
      |  MAINTAIN_LIBRARY
      |
58    |  MODIFY_DISK
      |  RESTORE_DISK
      |  SAVE_DISK
      |
61    |  PREPARE_DISK
      |  PREPARE_DISKETTE
      |  PREPARE_TAPE
      |  PREPARE_VOLUME
      |
66    |  CLEAR_VOLUME
      |  LIST_VOLUME
      |  MAINTAIN_VOLUME
      |
70    |  CONVERT_FORM
      |  CREATE_HELP_TEXT
      |  DUMPJRNL
      |  MAINTAIN_COMMAND
      |  MAINTAIN_FORM
      |  MAINTAIN_JAS
      |  ROLLFWD
      |
75    |  LIST_ACL
      |  MODIFY_ACL
      |
80    |  BUILD_FILE
      |  CLEAR_FILE
      |  COMPARE_FILE
      |  COPY_FILE
      |  CREATE_FILE
      |  DELETE_FILE
      |  LIST_FILE
      |  LOAD_FILE
      |  MERGE_FILE
      |  MODIFY_FILE
      |  PRINT_FILE
      |  RESTORE_FILE
      |  SAVE_FILE
      |  SORT_FILE
      |  SORT_INDEX
      |
81    |  CREATE_LINK
      |  DELETE_LINK
      |  LIST_LINK
      |
```

```
 85   |  CREATE_GEN
      |  DELETE_GEN
      |  LIST_GEN
      |  MODIFY_GEN
      |  SHIFT_GEN
      |
 90   |  CREATE_DIR
      |  DELETE_DIR
      |  LIST_DIR
      |
 91   |  CREATE_DK_FILE
      |  CREATE_MT_FILE
      |  CREATE_SYSTEM_FILES
      |  MAINTAIN_DATA_BASE
      |  REORG_DATA_BASE
      |
 95   |  BACKGROUND_PROGRAM_PREP
      |  BACKGROUND_CATALOG_
      |  BACKGROUND_FILE_
      |  BACKGROUND_VOLUME_
      |
 96   |  INIT_TDS_MGT_FILE
      |
 97   |  BACKGROUND_PROJ_MANAGER
      |  RETURN_FILE_VOLUME_
      |  RETURN_PROGRAM_PREP
      |  RETURN_PROJ_MANAGER
      |  RETURN_SYSADMIN
      |  RETURN_TDSAPPL_PREP
      |
100   |  MAINTAIN_FILE
      |  MODIFY_FILE_STATUS
      |
105   |  COMPARE_FILESET
      |  COPY_FILESET
      |  CREATE_FILESET
      |  DELETE_FILESET
      |  EXPAND_FILESET
      |  LIST_FILESET
      |  LOAD_FILESET
      |  PRINT_FILESET
      |  RESTORE_FILESET
      |  SAVE_FILESET
      |
```

110	ATTACH_CATALOG
	COPY_CATALOG
	CREATE_CATALOG
	DELETE_CATALOG
	LIST_CATALOG
	MODIFY_CATALOG
	RESTORE_CATALOG
	SAVE_CATALOG
	LIST_FILE_SPACE
	MODIFY_FILE_SPACE
113	CREATE_QUOTA_FILE
	DELETE_QUOTA_FILE
	INVALIDATE_QUOTA_FILE
	LIST_QUOTA
	MODIFY_QUOTA_FILE
	VALIDATE_QUOTA_FILE
116	LIST_CATSPACE
	MODIFY_CATSPACE
145	RETURN_FILE_VOLUME
	RETURN_SYSADMIN
150	BACKGROUND_CATALOG
	BACKGROUND_FILE
	BACKGROUND_VOLUME
160	FILL_MIRROR
	START_MIRROR
175	DEBUG
	SCANNER
215	DPRINT

9.2.4.2 Domain H_NOCTX

The commands not appearing in the following list have a priority of zero.

Priority	Command
15	FORCE_OUTPUT
	FORCE_USER_REQ
	MODIFY_CONFIGURATION
	MODIFY_STATION
	START_INPUT_READER
	START_OUTPUT_WRITER
	TERMINATE_INPUT_READER
	TERMINATE_OUTPUT_WRITER

```
120     |   DISPLAY_IO_CACHE
        |   MODIFY_IO_CACHE
        |   START_IO_CACHE
        |   START_IO_SERVER
        |   TERMINATE_IO_CACHE
        |   TERMINATE_IO_SERVER
        |
160     |   ENTER_JOB_REQ
        |   ENTER_FILETRANS_REQ
        |   ENTER_LIBTRANS_REQ
        |
165     |   CANCEL_JOB
        |   HOLD_JOB
        |   RELEASE_JOB
        |
170     |   CANCEL_OUTPUT
        |   HOLD_OUTPUT
        |   RELEASE_OUTPUT
        |
171     |   CANCEL_TERMINAL_OUTPUT
        |   HOLD_TERMINAL_OUTPUT
        |   RELEASE_TERMINAL_OUTPUT
        |
176     |   CANCEL_USER_REQ
        |
180     |   BYE
        |
200     |   ALTER_INPUT
        |
205     |   DISPLAY_CONFIGURATION
        |   DISPLAY_DEVIATION_TIME
        |   DISPLAY_JOB
        |   DISPLAY_LOAD
        |   DISPLAY_OUTPUT
        |   DISPLAY_OUTPUT_PARAMETERS
        |   DISPLAY_SECURITY_OPTIONS
        |   DISPLAY_TERMINAL_OUTPUT
        |   DISPLAY_TRACE
        |   DISPLAY_USER_REQ
        |
210     |   DISPLAY_PROFILE
        |   MODIFY_PROFILE
        |
212     |   CANCEL_DUMP
        |   DISPLAY_DUMP
        |   LIST_DUMP
        |
```

215	DPRINT
	LET
	MAIL
	SEND
220	TUTORIAL

9.2.4.3 Domain MAINTAIN_LIBRARY_SL

The commands not appearing in the following list have a priority of zero.

Priority	Command
50	COMPARE
	COPY
	CREATE
	DELETE
	DISPLAY
	EDIT
	FSE
	LIB
	LIST
	PRINT
	RENAME
	RENUMBER
55	QUIT
100	INFILE
	INLIB1
	INLIB2
	INLIB3
	OUTFILE
115	RESTORE
	SAVE

9.2.4.4 Domain MAINTAIN_LIBRARY_CU

The commands not appearing in the following list have a priority of zero.

Priority	Command
50	COPY
	DELETE
	DISPLAY
	LIB
	LIST
	PRINT
60	INFILE
	INLIB1
	INLIB2
	INLIB3
	OUTFILE
70	RESTORE
	SAVE
80	QUIT

9.2.4.5 Domains MAINTAIN_LIBRARY_LM and _BIN

The commands not appearing in the following list have a priority of zero.

Priority	Command
50	COPY
	DELETE
	DISPLAY
	LIB
	LIST
	PRINT
	RENAME
60	INFILE
	INLIB1
	INLIB2
	INLIB3
	OUTFILE
70	RESTORE
	SAVE
80	QUIT

9.2.5 PROGRAM_PREP Environment

Presentation of Commands

The program preparation environment is composed of a main environment named PROGRAM_PREP and a subenvironment named BG_PROG_PREP.

The BACKGROUND_PROGRAM_PREP command is used to switch from PROGRAM_PREP to BG_PROG_PREP. A return to the main environment can be done with the RETURN_PROGRAM_PREP command.

The set of accessible commands is the same in both environments. In other words, the end-user will use the BACKGROUND_PROGRAM_PREP command only if he wishes to be presented with menu screens that contain commands to run the utilities associated with program preparation, such as file command and form management. The more experienced user will avoid switching continuously from one environment to the other, by entering the required commands in line mode.

The menu screens for both environments are presented below.

```
          PROGRAM_PREP          |      BG_PROG_PREP
--------------------------------|-------------------------
1/3
 1  PREPARE_PROGRAM_BATCH       |  1  MAINTAIN_COMMAND
 2  PREPARE_PROGRAM_INT         |  2  MAINTAIN_FORM
 3  BIND_CU                      |  3  BUILD_FILE
 4  COBOL                       |  4  CREATE_FILE
 5  EXEC_PG                      |  5  LIST_FILE
 6  FSE                         |  6  LOAD_FILE
 7  LINK_PG                      |  7  PRINT_FILE
 8  MAINTAIN_LIBRARY            |  8  SORT_FILE
 9  LIST_FILE                   |  9  SORT_INDEX
10  PRINT_FILE                  | 10  SHIFT_GEN
11  BACKGROUND_PROGRAM_PREP     | 11  RETURN_PROGRAM_PREP
 -------------------------------|-------------------------
2/3
 1  ENTER_JOB_REQUEST           |  1  ENTER_JOB_REQUEST
 2  CANCEL_JOB                  |  2  CANCEL_JOB
 3  HOLD_JOB                    |  3  HOLD_JOB
 4  RELEASE_JOB                 |  4  RELEASE_JOB
 5  CANCEL_OUTPUT               |  5  CANCEL_OUTPUT
 6  HOLD_OUTPUT                 |  6  HOLD_OUTPUT
 7  RELEASE_OUTPUT              |  7  RELEASE_OUTPUT
 8  DEBUG                       |  8  BYE
 9  SCANNER                      |
                                |
10  BYE                         |
--------------------------------|-------------------------
```

```
3/3
 1  ALTER_INPUT                |  1  ALTER_INPUT
 2  DISPLAY_CONFIGURATION      |  2  DISPLAY_CONFIGURATION
 3  DISPLAY_JOB                |  3  DISPLAY_JOB
 4  DISPLAY_LOAD               |  4  DISPLAY_LOAD
 5  DISPLAY_OUTPUT             |  5  DISPLAY_OUTPUT
 6  DISPLAY_PROFILE            |  6  DISPLAY_PROFILE
 7  MODIFY_PROFILE             |  7  MODIFY_PROFILE
                               |
 8  DPRINT                     |  8  DPRINT
 9  LET                        |  9  LET
10  MAIL                       | 10  MAIL
11  SEND                       | 11  SEND
                               |
12  TUTORIAL                   | 12  TUTORIAL
-------------------------------|-------------------------
```

Hidden Commands

The commands that are not accessible are related to file and volume management, or operations which are strictly reserved to the System Administrator. The non-accessible commands are listed below.

```
CLEAR_VOLUME            MAINTAIN_CATALOG
COPY_CATALOG            MAINTAIN_VOLUME
CREATE_CATALOG          MODIFY_ACL
CREATE_DIR              MODIFY_CATALOG
CREATE_DK_FILE          MODIFY_CATSPACE
CREATE_GEN              MODIFY_DISK
CREATE_LINK             MODIFY_FILE_SPACE
CREATE_MT_FILE          MODIFY_FILE_STATUS
CREATE_SYSTEM_FILES     MODIFY_GEN
DELETE_CATALOG
DELETE_DIR              PREPARE_DISK
DELETE_FILE             PREPARE_DISKETTE
DELETE_FILESET          PREPARE_TAPE
DELETE_GEN              PREPARE_VOLUME
DELETE_LIBRARY          RESTORE_CATALOG
DELETE_LINK             SAVE_CATALOG

CREATE_QUOTA_FILE       MODIFY_QUOTA_FILE
DELETE_QUOTA_FILE       MAINTAIN_MIGRATION
INVALIDATE_QUOTA_FILE   MAINTAIN_QUOTA
MODIFY_IO_CACHE         VALIDATE_QUOTA_FILE
```

Non-Standard Commands

We describe here the non-standard procedures and their associated items (source subfiles, Help texts, etc.) that are delivered with the PROGRAM_PREP environment.

Two user-visible procedures are provided:

```
PREPARE_PROGRAM_INT      (abbreviation PPI)
PREPARE_PROGRAM_BATCH    (abbreviation PPB)
```

These two procedures automatically chain the phases of program preparation and debugging in interactive and Batch mode respectively.

The use of these procedures is documented in the Help texts which are available at each level (environment, command and parameter). The PREPARE_PROGRAM_INT (PPI) command is used to edit, compile, link, and execute a set of programs, in interactive mode.

The PREPARE_PROGRAM_BATCH (PPB) command is used to edit, compile (in batch mode), link (in interactive mode), and execute (in interactive or batch mode) a set of programs.

At each step (compilation, linking, execution), a prompt screen is displayed and different options are proposed, according to the results of the previous steps.

For example, if the linking is successful, you are asked if you want to:

- execute the program (default option)
- edit a program
- quit the command.

If the linking has failed, you are asked if you want to:

- edit a program (default option)
- link again
- call the SCANNER utility
- print the results of the linking on a printer
- quit the command

Each screen corresponds to a hidden procedure called by the PPI (or PPB) command.

9.2.5.1 PREPARE_PROGRAM_INT

The normal sequence of the PPI command is:

- editing then compiling all programs
- linking
- executing the load module.

It is possible, however, to proceed directly with the linking or the execution (LINK and EXECUTE options of PPI).

1. If you select the LINK option, you are prompted with the screen of the SELECT_YOUR_OPTION_LAST_PG procedure, which asks for confirmation of the LINK option.
2. If you select the EXECUTE option, you are prompted with the screen of the EXEC_PG procedure, which asks you for the execution parameters.
3. If you select the default option (edit and compile a set of programs), after editing and compiling a program, according to the severity of the compilation and the results of the previous steps, you are prompted with a screen of one of the following procedures:

   If the program is not the last of the list:

```
SELECT_YOUR_OPTION
SELECT_YOUR_OPTION_        (compilation successful,
SELECT_YOUR_OPTION___                   severity 0)

COMPILATION_SUCCESSFUL     (compilation successful,
COMPILATION_SUCCESSFUL__             severity 1 or 2)

COMPILATION_FAILED         (compilation failed,
COMPILATION_FAILED_                  severity 3 or 4)
```

   If your program is the last of the list:

```
SELECT_YOUR_OPTION_LAST_PG_  (compilation successful,
                                        severity 0)

SELECT_YOUR_OPTION_LAST_PG   (compilation successful,
                                       severity 1 or 2)

COMPILATION_FAILED_          (compilation failed,
                                       severity 3 or 4)
```

   For example, SELECT_YOUR_OPTION asks you if you want to:

   – edit the next program
   – skip to the next program
   – link the main program
   – quit the command.

COMPILATION_SUCCESSFUL asks in addition if you want to see the result of the compilation (to correct the severity 1 or 2), or to use the SCANNER utility.

After linking, you are prompted with a screen from one of the following procedures, depending on the result of the link:

```
SELECT_YOUR_OPTION-    (link successful, severity 0)
LINK_SUCCESSFUL       (link successful, severity 1 or 2)
LINK_FAILED            (link failed, severity 3 or 4)
```

If you select EXECUTION, you are prompted with the screen of EXEC_PG, which asks for the execution parameters. After execution, the prompt screen of SELECT_YOUR_OPTION___ asks you if you want to:

– re-execute the program
– return to the PPI command
– quit the command.

Two other procedures, COMPILATION_RESULT and LINK_RESULT allow you to see the result of the last compilation or link.

In order to permit the user to define the context in which compilation and linkage will occur, other hidden procedures are used:

Compiling:

```
    calls                calls
PPI ----> COBOL_OPT_PPI ----> COBOL_OPT_PPI_NAME
```

In the COBOL_OPT_PPI procedure, the parameters of COBOL are declared. In the COBOL_OPT_PPI_NAME procedure, these parameters are assigned. This procedure can be replaced by a user procedure.

Linking:

```
    calls               calls
PPI ----> LINK_PG_OPT_ ----> LINK_PG_OPT_NAME
```

In the LINK_PG_OPT_ procedure, the parameters of LINK_PG are declared. In the LINK_PG_OPT_NAME procedure, these parameters are assigned. This procedure can be replaced by a user procedure.

Associated Help Texts

Helps texts are available at each level (command and parameter). You request a Help text by entering a "?" in the corresponding field. All the names of these texts are prefixed with H_ENVT_.

Associated Subfiles

The PPI command uses the following subfiles:

- C_PGP_FSE_MODIFY
  which starts the MODIFY command of the Full Screen Editor (FSE) when you want to edit a program
- C_PGP_INIT_CR_SEV2 and C_PGP_EDIT_CR_SEV2
- C_PGP_INIT_CR_SEV3 and C_PGP_EDIT_CR_SEV3
- C_PGP_INIT_CR_SEV4 and C_PGP_EDIT_CR_SEV4
  which write in files the result of the compilation, to permit you to consult this result when you select the COMPILATION_RESULT option.
- C_PGP_INIT_LR and C_PGP_EDIT_LR
  which writes in a file the result of the linking to permit you to consult this result when you select the LINK_RESULT option.

9.2.5.2 PREPARE_PROGRAM_BATCH

The normal sequence of the PPB command is:

- editing
- submitting the compilation in batch mode
- checking the compilation results
- linking in Interactive mode
- execution in Batch or Interactive mode.

It is, however, possible to proceed directly with the linking or execution (LINK and EXECUTE options of PPB).

1. If you select the LINK option, you are prompted with the screen of the SELECT_YOUR_OPTION_ procedure, which asks for confirmation of the LINK option.
2. If you select the EXECUTE option, you are prompted with the screen of the SELECT_YOUR_OPTION___ procedure, which asks you if you want to execute your program.
3. If you select the default option (edit and compile a set of programs), The Full Screen Editor is first started, then the Batch command file is submitted. For this, other hidden procedures are called:

```
    calls                calls
PPB ----> EJR_COB_OPT_PPB ----> EJR_COB_OPT_NAME
```

In the EJR_COB_OPT_PPB procedure, the parameters of the ENTER_JOB_REQ directive are declared. In the EJR_COB_OPT_NAME procedure, these parameters are assigned. A user procedure can be substituted for the EJR_COB_OPT_NAME procedure.

Compilations are executed in Batch mode. So according to the state of the job (compilation is in scheduling or has not terminated, compilation has finished and was successful, compilation has finished and failed), you are prompted with a different screen. The different screens ask you if you want to edit the next program, to wait for the result of the compilations, to check the result of the compilations, etc.

The procedures are:

```
    SELECT_YOUR_OPTION
or  SELECT_YOUR_OPTION_
or  SELECT_YOUR_OPTION__
```

If you select the CHECK option, you are prompted with the screen of one of the following procedures, depending on the result of the compilation:

```
    COMPILATION_SUCCESSFUL
or  COMPILATION_FAILED
or  SELECT_YOUR_OPTION_
or  COMPILATION_SUCCESSFUL_
or  COMPILATION_FAILED_
```

If you select the LINK option, two procedures are called:

```
    calls              calls
PPB ----> LINK_PG_OPT_ ----> LINK_PG_OPT_NAME
```

In the LINK_PG_OPT_ procedure, the parameters of LINK_PG are declared. In the LINK_PG_OPT_NAME procedure, these parameters are assigned. A user procedure can be substituted for the LINK_PG_OPT_NAME procedure.

Following the linking, you are prompted with a screen from one of the following procedures, depending on the result:

```
    SELECT_YOUR_OPTION___ (link successful, severity 0)
or  LINK_SUCCESSFUL      (link successful, severity 1 or 2)
or  LINK_FAILED           (link failed, severity 3 or 4).
```

Two other procedures, COMPILATION_RESULT_PPB and LINK_RESULT, allow you to look at the result of the compilation or link.

If you select the EXECUTE option in Interactive mode, you are prompted with the screen of the EXEC_PG_ procedure, which asks for the execution parameters. You are then prompted with the screen of the SELECT_YOUR_OPTION___ procedure, which asks if you want to reexecute the program.

If you select the EXECUTE option in Batch mode, two procedures are called:

```
    calls                    calls
PPB ----> EJR_EXEC_OPT_PPB ----> EJR_EXEC_OPT_NAME
```

In the EJR_EXEC_OPT_PPB procedure, the parameters of the ENTER_JOB_REQ directive are declared. In the EJR_EXEC_OPT_NAME procedure, these parameters are assigned. A user procedure can be substituted for the EJR_EXEC_OPT_NAME procedure. Then, according to the state of the job, you are prompted with the screen of either the EXECUTION-WAITING procedure or EXECUTION_TERMINATED procedure.

Associated Help Texts

Help texts are available at each level (command and parameter). You request a Help text by entering a "?" in the corresponding field. All the names of these texts are prefixed with H_ENVT_.

Associated Subfiles

In the PPB command the following subfiles are used:

- C_PGP_FSE_MODIFY
  which executes the MODIFY command of Full Screen Editor (FSE) when you want to edit a program.
- C_PGP_INIT_SYNCHRO
  which initializes the files in which the results of the compilations will be written.
- C_PGP_INIT_LR and C_PGP_EDIT_LR
  which write in a file the result of the linking, so that you can consult this result when you select the LINK_RESULT option.

The C_PGP_JCL_COBOL subfile is used in the EJR_COB_OPT_PPB command. This subfile uses other subfiles. They are:

- COBOL_OPT_PPB_NAME
  which contains the COBOL step. This subfile can be replaced by a user subfile.
- C_PGP_EDIT_CR_NOLISTING
- C_PGP_EDIT_CR_PPB_SEV2
- C_PGP_EDIT_CR_PPB_SEV3
- C_PGP_EDIT_CR_PPB_SEV4
  which write the results of the compilations in files.

The C_PGP_JCL_EXEC subfile is used in the EJR_EXEC_OPT_PPB command. This subfile uses another subfile named EXEC_OPT_PPB_NAME, which contains the execution step. The EXEC_OPT_PPB_NAME subfile can be replaced by a user subfile

9.2.5.3 Command Personalization

During the execution of the commands PREPARE_PROGRAM_INT and PREPARE_PROGRAM_BATCH the user is asked to enter values that define the context in which updates, compilation and linking will occur. Examples of such options are library definitions, program names, or output listing requests.

In order to reduce the length of the dialog, only a few parameters are entered interactively at the terminal. Other options are arbitrarily taken inside the procedures that are called in the PPI and PPB contexts.

Of course, these options may be statically redefined by the System Administrator, who is allowed to modify or redefine any component of the standard environments. He will not, however, be able to define options to fit every end-user working under the standard environments.

For this reason, any user can replace the standard options by his own. Global variables are defined which may be initialized by the user (for example, at user or project startup level) with the name of the private procedure or JCL member which is to be executed in place of the standard one. This concerns the compile, link and execution options.

The options which may replace the standard ones are:

```
-  the interactive compile options     }  for PREPARE_PROGRAM_INT
-  the interactive link options        }  context
```

and:

```
-  the compile options                 }
-  the EJR options of compile          }
-  the interactive link options        }  for
                                       }  PREPARE_PROGRAM_BATCH
-  the Batch execution options         }  context
-  the EJR options of Batch execution  }
```

The following schema shows the parameterization features in both PPI and PPB contexts:

Table 9-1. PPI and PPB Options - PROGRAM_PREP Environment

	PPI	PPB
COBOL parameters	*G_PGP_COBOL_OPT_PPI (COBOL_OPT_PPI_NAME) (procedure)	*G_PGP_COBOL_OPT_PPB (COBOL_OPT_PPB_NAME) (JCL)
COBOL EJR parameters	-------	*G_PGP_EJR_COBOL_OPT (EJR_COB_OPT_NAME) (procedure)
LINK parameters	*G_PGP_LINK_PG_OPT (LINK_PG_OPT_NAME) (procedure)	
Execution parameters	(entered at the terminal with EXEC_PG_ procedure)	*G_PGP_EXEC_OPT_PPB (EXEC_OPT_PPB_NAME) (JCL)
Execution EJR parameters	-------	*G_PGP_EJR_EXEC_OPT (EJR_EXEC_OPT_NAME) (procedure)

The names preceded by an asterisk are the names of GCL global variables. Each of these may contain the name of a procedure or JCL member to be executed in place of the standard one. All these global variables are declared NAME (with an implicit length 31).

The second name enclosed in parentheses is the name of the "standard" procedure (or JCL member) which is to be executed if the associated variable has no assigned value.

For instance, the user who includes in his startup sequence:

```
GLOBAL  G_PGP_COBOL_OPT_PPI  NAME;
LET  G_PGP_COBOL_OPT_PPI  MY_COBOL_OPT;
```

will execute the MY_COBOL_OPT procedure when PPI is called, before the COBOL compiler is invoked. This user procedure will contain the assignment of the user-specific

COBOL options in a format which is described below. To personalize your COBOL options you then have to:

1. Create a procedure which initializes the values of the option that are different from the default ones (given below)
2. Put the binary library in which this procedure is stored into your binary search path
3. Declare the global variable G_PGP_COBOL_OPT_PPI and store in it the name of your procedure.

When the specified global variable is not declared, the standard procedure COBOL_OPT_PPI_NAME is called from SYS.HBINLIB which assigns predefined values to the COBOL option.

NOTE:

All the names in the schema refer to the procedure names or variables that contain user procedure names, except the ones related to the COBOL option and execution option in a batch context (PPB).

That is: COBOL_OPT_PPB_NAME and EXEC_OPT_PPB_NAME are JCL members stored in SYS.HSLLIB, and G_PGP_COBOL_OPT_PPB and G_PGP_EXEC_OPT_PPB are global variables which may contain the names of JCL members stored in a user library.

Interactive Cobol Options

```
Default procedure   : COBOL_OPT_PPI_NAME

User procedure name : contained in the global variable
                      G_PGP_COBOL_OPT_PPI
```

The names, characteristics, and default values of the (local) variables which may be set in the user procedure are listed below:

NAME	TYPE	LENGTH	DEFAULT
LIST	BOOL	1	1
CLIST	BOOL	1	1
EXPLIST	BOOL	1	0
MAP	BOOL	1	0
XREF	BOOL	1	1
DCLXREF	BOOL	1	0
DEBUG	BOOL	1	1
DEBUGMD	BOOL	1	-

```
LEVEL        NAME      4   VALUES=(L64,NSTD,  L64
                           L62,CBC,CBX,ANSI,
                           DSA,NBS1,NBS2,
                           NBS3,NBS4)
OBSERV       BOOL      1                        1
DIAG         NAME      3   VALUES=(IN,AFT,BEF)  IN
SILENT       BOOL      1                        0
CARDID       BOOL      1                        -
CASEQ        BOOL      1                        1
CKSEQ        BOOL      1                        1
SUBCK        BOOL      1                        -
CODAPND      BOOL      1                          0
DDLIST       BOOL      1                        0
EXPSIZE      BOOL      1                          -
TEMP         NAME      4                        -
OBJNAME                1   VALUES=(0,1,A,B)      1
SUBOPT       BOOL      1                        0
LSEV         NAME      1   VALUES=(F,0)          F
DSEGMAX      DEC       4   VALUES=>0             -
PSEGMAX      DEC       4   VALUES=>0             -
OBJLIST      BOOL      1                          0
DICLIST      BOOL      1                          0
DUMP         NAME      4   VALUES=(DATA,ALL,NO)  -
```

EXAMPLE:

Assume all the above default values are your COBOL options, except MAP (you require MAP=1) and DCLXREF (you require DCLXREF=1), you then must:

1. Define a procedure:

```
PROC  MY_COBOL_OPT  HIDE=-1  ... ;
   LET  MAP  1;
   LET  DCLXREF  1 ;
ENDPROC ;
```

2. Initiate your startup with:

```
MWINLIB  BIN  MY_BINLIB;
GLOBAL   G_PGP_COBOL_OPT_PPI  NAME;
LET      G_PGP_COBOL_OPT_PPI  MY_COBOL_OPT;
```

❑

NOTE:

MY_BINLIB is the binary library in which the procedure MY_COBOL_OPT has been stored.

Interactive Link Options

```
Default Procedures  : LINK_PG_OPT_NAME

User Procedure Name : contained in the global variable
                      G_PGP_LINK_PG_OPT
```

The names, characteristics, and default values of the (local) variables which may be set in the user procedure are listed below:

```
NAME       TYPE   LENGTH                          DEFAULT

COMFILE    FILE     78                              -
COMFAC     BOOL      1                               -
SYSDEF     LIB      78                             -
SRCHLIB    NAME      5  VALUES=(FIRST,LAST)   FIRST
DUMP       NAME      4  VALUES=(DATA,ALL,NO)      -
```

COBOL EJR Options

```
Default Procedure  : EJR_COB_OPT_NAME

User Procedure Name: contained in the global variable
                     G_PGP_EJR_COBOL_OPT
```

The names, characteristics, and default values of the (local) variables which may be set in the user procedure are listed below:

```
NAME          TYPE   LENGTH                    DEFAULT

CLASS         NAME      1  VALUES=&A              -
PRIORITY      DEC       1  VALUES=0<=*<=7        -
HOLDOUT       BOOL      1                         -
DEST          CHAR     17                         -
HOLD          BOOL      1                         -
REPEAT        BOOL      1                         -
```

Batch Cobol Options

```
Default JCL Subfile   : COBOL_OPT_PPB_NAME

User JCL Subfile Name : contained in the global variable
                        G_PGP_COBOL_OPT_PPB
```

The COBOL step call is stored in the COBOL_OPT_PPB_NAME subfile of the SYS.HSLLIB library. Its format is:

```
COBOL  SOURCE = &1
       INLIB  = (&2 &3)
       CULIB  = (&4 &5)
       PRTLIB = (&6 &7)
       LEVEL  = L64;
```

The input parameters (&1 to &7) have been computed from the user options entered at the terminal (PPB).

This subfile may be replaced by a user subfile whose name is to be stored in the G_PGP_COBOL_OPT_PPB variable (declared NAME). This subfile is assumed to belong to the user SL library declared when entering the PPB command (SLLIB parameter)

EXAMPLE:

1. Define a source member MY_BATCH_COBOL_OPT that contains the following, and store it in your private source library.

```
COBOL  SOURCE = &1
       INLIB  = (&2 &3)
       CULIB  = (&4 &5)
       PRTLIB = (&6 &7)
       LEVEL  = L64
       LIST  CLIST  XREF  MAP;      (your options)
```

2. Initiate your startup with:

```
GLOBAL  G_PGP_COBOL_OPT_PPB  NAME;
LET  G_PGP_COBOL_OPT_PPB  MY_BATCH_COBOL_OPT;
```

❑

Batch Execution Options

```
Default JCL Subfile   : EXEC_OPT_PPB_NAME

User JCL subfile name : contained in the global variable
                        G_PGP_EXEC_OPT_PPB
```

The basic JCL is stored in the EXEC_OPT_PPB_NAME subfile of the SYS.HSLLIB library. Its format is:

```
STEP &1 (&2  &3);
ENDSTEP;
```

The input parameters (&1, &2, and &3) design the load module name and LM library. They have been computed from the user options entered at the terminal (PPB).

This subfile may be replaced by a user subfile whose name is to be stored in the G_PGP_EXEC_OPT_PPB variable (declared NAME). This subfile is assumed to belong to the user SL library declared when entering the PPB command (SLLIB parameter)

EXAMPLE:

1. Define a source member MY_EXEC_OPT that contains the following, and store it in your private source library.

```
STEP &1 (&2  &3)
     CPTIME = 1000
     LINES = 1000
     DEBUG = MY_DBFILE ...
ASSIGN ... ;
ASSIGN ... ;
ENDSTEP;
```

2. Initiate your startup with:

```
GLOBAL  G_PGP_EXEC_OPT_PPB  NAME;
LET  G_PGP_EXEC_OPT_PPB  MY_EXEC_OPT;
```

❑

Batch Execution EJR Options

```
Default Procedure   : EJR_EXEC_OPT_NAME

User Procedure Name : contained in the global variable
                      G_PGP_EJR_EXEC_OPT
```

The names, characteristics, and default values of the (local) variables which may be set in the user procedure are listed below:

```
NAME          TYPE   LENGTH                  DEFAULT

CLASS         NAME      1  VALUES=&A            -
PRIORITY      DEC       1  VALUES=0<=*<=7       -
HOLDOUT       BOOL      1                        -
DEST          CHAR     17                       -
HOLD          BOOL      1                       -
REPEAT        BOOL      1                       -
```

9.2.6 TDSAPPL_PREP Environment

Presentation of Commands

The TDS application preparation environment is composed of a main environment named TDSAPPL_PREP and a subenvironment named BG_TDS_PREP.

The BACKGROUND_TDSAPPL_PREP command is used to switch from TDSAPPL_PREP to BG_TDS_PREP. You can return to TDSAPPL_PREP by using the RETURN_TDSAPPL_PREP command.

The set of accessible commands is the same in both environments. In other words, the end-user will use the BACKGROUND_TDSAPPL_PREP command only if he wishes to be presented with menu screens that contain commands to run the utilities associated with TDS application preparation. The more experienced user will avoid switching continuously from one environment to the other, by entering the required commands in line node.

The menu screens for both environments are presented below.

```
        TDSAPPL_PREP            |      BG_TDS_PREP
--------------------------------|------------------------
1/3
 1  PREPARE_TPR_BATCH           |  1  MAINTAIN_COMMAND
 2  PREPARE_TPR_INT             |  2  MAINTAIN_FORM
                                |
 3  COBOL                       |  3  BUILD_FILE
 4  EXEC_PG                     |  4  CREATE_FILE
 5  FSE                         |  5  LIST_FILE
 6  LINK_PG                     |  6  LOAD_FILE
 7  MAINTAIN_LIBRARY            |  7  PRINT_FILE
                                |  8  SORT_FILE
 8  MAINTAIN_FORM               |  9  SORT_INDEX
                                |
 9  LIST_FILE                   | 10  SHIFT_GEN
10  PRINT_FILE                  |
                                | 11  RETURN_TDSAPPL_PREP
11  BACKGROUND_TDSAPPL_PREP     |
 -------------------------------|------------------------
```

```
2/3
 1  ENTER_JOB_REQUEST          |  1  ENTER_JOB_REQUEST
                               |
 2  CANCEL_JOB                 |  2  CANCEL_JOB
 3  HOLD_JOB                   |  3  HOLD_JOB
 4  RELEASE_JOB                |  4  RELEASE_JOB
                               |
 5  CANCEL_OUTPUT              |  5  CANCEL_OUTPUT
 6  HOLD_OUTPUT                |  6  HOLD_OUTPUT
 7  RELEASE_OUTPUT             |  7  RELEASE_OUTPUT
                               |
 8  DEBUG                      |  8  BYE
 9  SCANNER                     |
                               |
10  BYE                        |
-------------------------------|-------------------------
3/3
 1  ALTER_INPUT                |  1  ALTER_INPUT
                               |
 2  DISPLAY_CONFIGURATION      |  2 DISPLAY_CONFIGURATION
 3  DISPLAY_JOB                |  3  DISPLAY_JOB
 4  DISPLAY_LOAD               |  4  DISPLAY_LOAD
 5  DISPLAY_OUTPUT             |  5  DISPLAY_OUTPUT
                               |
 6  DISPLAY_PROFILE             |  6  DISPLAY_PROFILE
 7  MODIFY_PROFILE             |  7  MODIFY_PROFILE
                               |
 8  DPRINT                     |  8  DPRINT
 9  LET                        |  9  LET
10  MAIL                       | 10  MAIL
11  SEND                       | 11  SEND
                               |
12  TUTORIAL                   | 12  TUTORIAL
-------------------------------|-------------------------
```

Hidden Commands

The commands that are not accessible in the TDSAPPL_PREP environment are related to file and volume management, or operations which are strictly reserved to the System Administrator. The non-accessible commands are listed below.

```
CLEAR_VOLUME            MAINTAIN_CATALOG
COPY_CATALOG            MAINTAIN_VOLUME
CREATE_CATALOG          MODIFY_ACL
CREATE_DIR              MODIFY_CATALOG
CREATE_DK_FILE          MODIFY_CATSPACE
CREATE_GEN              MODIFY_DISK
CREATE_LINK             MODIFY_FILE_SPACE
CREATE_MT_FILE          MODIFY_FILE_STATUS
```

```
CREATE_SYSTEM_FILES     MODIFY_GEN
DELETE_CATALOG
DELETE_DIR              PREPARE_DISK
DELETE_FILE             PREPARE_DISKETTE
DELETE_FILESET          PREPARE_TAPE
DELETE_GEN              PREPARE_VOLUME
DELETE_LIBRARY          RESTORE_CATALOG
DELETE_LINK             SAVE_CATALOG

CREATE_QUOTA_FILE       MODIFY_QUOTA_FILE
DELETE_QUOTA_FILE       MAINTAIN_MIGRATION
INVALIDATE_QUOTA_FILE   MAINTAIN_QUOTA
MODIFY_IO_CACHE         VALIDATE_QUOTA_FILE
```

9.2.6.1 Overview of PTI and PTB Procedures

The use of the PTI and PTB procedures, and those called from them, is documented in the Help texts which are available at each level (environment, command, and parameter). The following subsections are intended for those interested in understanding and modifying commands of the standard TDSAPPL_PREP environment.

Whether the TPRs are developed in interactive mode (PREPARE_TPR_INT) or in batch (PREPARE_TPR_BATCH), tests are made under the control of the same GCL procedure TPR_TEST. This procedure accesses information regarding the TDS through a load module which manages concurrent accesses.

The following schema shows how this is done:

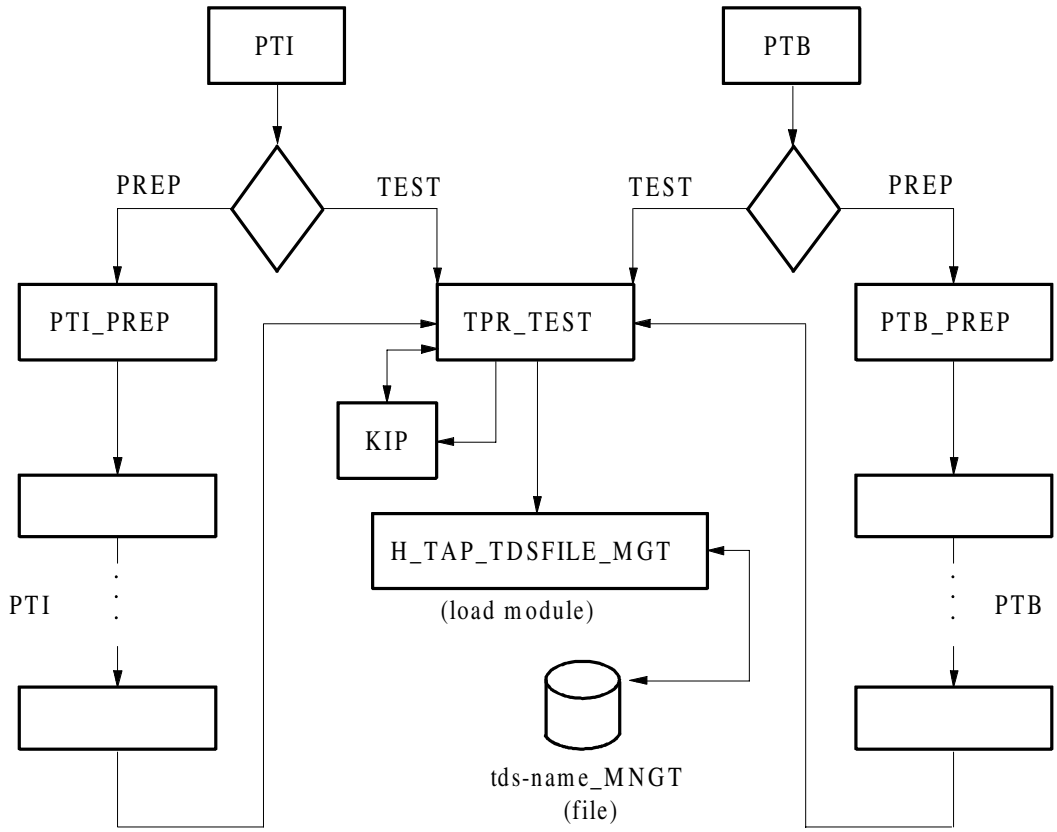

Figure 9-4. PTI and PTB Procedures

For more information about the TDS management file and its associated items (H_TAP_TDSFILE_MGT load module and INIT_TDS_MGT_FILE procedure) refer to the PROJ_MANAGER environment description below.

9.2.6.2 Elements of Communication

The test mechanism is based on elements supplied by the user and stored in global variables.

These variables are either initialized in a project or user startup sequence, or entered via a GCL procedure (KEY_IN_PARA, KIP).

```
G_TAP_TDS  (CHAR 4)         contains the name of the
                             TDS application.

G_TAP_TDS_CLASS  (CHAR 1)  contains the TDS execution class.

G_TAP_WORK_CLASS  (CHAR 1) contains the LIBMAINT execution class.

G_TAP_JCLLIB  (LIB 78)     contains the
                            characteristics of the
                            library in which the JCL
                            member activating the TDS
                            is to be found.

G_TAP_JCLMB  (CHAR 12)     contains the member name.

G_TAP_SM  (CHAR 4)         contains the name of the
                            SM in which the user TPRs
                            are linked (TPR, TPR1, ...TPR99).

G_TAP_SM1  (LIB 78)        contains the
                            characteristics of the
                            first SM library of the
                            JCL command JOBLIB.

G_TAP_SM2  (LIB 78)        contains the
                            characteristics of the
                            second SM library of the
                            JCL command JOBLIB.

G_TAP_USR_TST  (CHAR 12)   contains the name used to
                            connect the user to TDS.

G_TAP_PSW_TST  (NAME 12)   contains the password
                            associated with this user.
```

Other global variables are used for communication between procedures (not end-user visible):

```
G_TAP_UA  (CHAR 72)        contains the information
                            present in the test
                            management file.
```

```
G_TAP_MAST_OP  (CHAR 12)   contains the name of the
                            user who started TDS.

G_TAP_USER_LOCK  (CHAR 12) contains the name of the
                            last user to have modified
                            the file.

G_TAP_SMLIB_ORDER  (DEC 1) contains an indicator of
                            the search path of the TPRs.

G_TAP_SM1_EFN  (CHAR 12)   contains the EFN of the
                            first library in the JCL
                            command JOBLIB.

G_TAP_SM2_EFN  (CHAR 12)   contains the EFN of the
                            second library in the JCL
                            command JOBLIB.

G_TAP_USERID  (CHAR 12)    contains the user
                            identification.
```

Apart from the global variables described above, TPR_TEST uses the following:

```
G_TAP_COM  (CHAR 3)        contains the name of the
                            calling procedure (PTI or PTB).

G_TAP_STP  (CHAR 8)        contains the name of a
                            member created in
                            SYS.HSLLIB where the
                            command "tds-name M STOP"
                            will be stored
                            (cf. COMF_TAP_STP subfile).

G_TAP_EJT  (CHAR 12)       contains the name of a
                            member created in
                            SYS.HSLLIB where the
                            EJR command to start TDS
                            will be stored
                            (cf. COMF_TAP_EJT subfile).

G_TAP_EJV  (CHAR 11)       contains the name of a
                            member created in
                            SYS.HSLLIB where the
                            EJR command to start
                            VMAINT will be stored
                            (cf. COMF_TAP_EJV subfile).
```

```
G_TAP_EJLV  (CHAR 15)      contains the name of a
                            member created in
                            SYS.HSLLIB where the
                            EJR command to start
                            LIBMAINT and VMAINT
                            will be stored
                            (cf. COMF_TAP_EJLV subfile).

G_TAP_CPT  (CHAR 11)       contains the name of a
                            member created in
                            SYS.HSLLIB where the
                            CONNECT_APPLICATION
                            command to connect to TDS
                            will be stored
                            (cf. COMF_TAP_SWxxx subfile).

G_TAP_WAIT  (DEC 5)        contains the time in
                            milliseconds between two
                            attempts to access the TDS
                            session log file if it is busy.

G_TAP_CONNECT  (BOOL)      contains an indicator for
                            direct connection to TDS
                            (i.e., with no SWITCH).
```

9.2.6.3 Differences with regard to PROGRAM_PREP Environment

Differences Between PTI_PREP and PPI

- the keyword LMLIB is a synonym of SMLIB.
- there is an additional keyword (SM) to take, if necessary, the name of the SM where the user TPRs will be linked.
- there is a call for the compilation and linking of specific procedures (COBOL_OPT_PTI_ and LINK_TPR_OPT_) with dynamic construction of the name of the COMFILE of the LINKER.
- at the EXECUTION label, PTI is started with the TEST option.

Differences Between PTB_PREP and PPB.

- the keyword LMLIB is a synonym of SMLIB.
- there is an additional keyword (SM) to take, if necessary, the name of the SM where the user TPRs will be linked.

- there is a call for the compilation and linking of specific procedures (EJR_COB_OPT_PTB and LINK_TPR_OPT_) with dynamic construction of the name of the COMFILE of the LINKER.
- at the EXECUTION label, PTB is started with the TEST option.

NOTE:
All other internal mechanisms of PPI and PPB are unchanged.

Call of the COBOL Compiler and the LINKER in PTI_PREP.

- PTI_PREP calls the compiler via the COBOL_OPT_PTI_ procedure. This procedure, before the actual CALL to COBOL, takes the compilation parameters either from a GCL procedure supplied by the user (the name of the GCL procedure is found in the global variable G_TAP_COBOL_OPT_PTI), or from a standard procedure (COBOL_OPT_PTI_NAME).
- PTI_PREP calls the LINKER via the LINK_TPR_OPT_ procedure. This procedure, before the actual CALL to the LINKER, takes the parameters either from a GCL procedure supplied by the user (the name of the GCL procedure is found in the global variable G_TAP_LINK_TPR_OPT), or from a standard procedure (LINK_TPR_OPT_NAME).

Call of the COBOL Compiler and the LINKER in PTB_PREP.

- The call of the COBOL compiler is done in the same way as in PPB.
- Within the procedure itself is found a call to EJR_COB_OPT_PTB which in its turn makes a call, either to a user procedure, if its name is found in the global variable G_TAP_EJR_COBOL_OPT, or to a standard procedure

  (EJR_COB_TPR_OPT_NAME) whose role is to supply the keywords intended to receive the EJR parameters (execution class, listing to be held or not, etc.).
- The LINKER is started in interactive mode in the same way as in PTI_PREP.

9.2.6.4 Command Personalization

As for the PROGRAM_PREP environment, you have the facility to personalize your compile or link options. Global variables are defined which may be initialized by the user (for instance, in the startup sequence) with the name of the private procedure or JCL member which is intended to be executed in place of the standard one.

The options which may be substituted for the standard ones are:

```
-  the interactive compile options }
-  the interactive link options    }  for PREPARE_TPR_INT context
```

and:

```
-  the compile options            }
-  the EJR compiler options       }  for PREPARE_TPR_BATCH context
-  the (interactive) link options }
```

Only the TPR preparation phase is concerned (PTI_PREP, PTB_PREP). The following schema shows the parameterization features in both PTI and PTB contexts:

Table 9-2. PPB and PPI Options - TDSAPPL_PREP Environment

	PTI	PTB
COBOL parameters	*G_TAP_COBOL_OPT_PTI (COBOL_OPT_PTI_NAME) (procedure)	*G_TAP_COBOL_OPT_PTB (COBOL_OPT_PTB_NAME) (JCL)
COBOL EJR parameters	-------	*G_TAP_EJR_COBOL_OPT (EJR_COBOL_TPR_OPT_NAME) (procedure)
LINK parameters	*G_TAP_LINK_TPR_OPT (LINK_TPR_OPT_NAME) (procedure)	

The names preceded with an asterisk are the names of GCL global variables. Each of these may contain the name of a procedure (or JCL member) to be executed in place of the standard one. All these global variables are declared NAME (with an implicit length 31).

The second name enclosed in parentheses is the name of the "standard" procedure (or JCL member) which is to be executed if the associated variable has no assigned value. For example, the user who initiates in his startup sequence:

```
GLOBAL  G_TAP_COBOL_OPT_PTI  NAME;
LET  G_TAP_COBOL_OPT_PTI  MY_COBOL_OPT;
```

will execute the MY_COBOL_OPT procedure when PTI is called, before the COBOL compiler is invoked in place of the COBOL_OPT_PTI_NAME procedure. This user procedure will contain the assignment of the user-specific COBOL options in a format which is described below. To personalize your COBOL options you then have to:

1. Create a procedure which initializes the values of the option that are different from the default ones (given below)
2. Put the binary library in which this procedure is stored into your binary search path
3. Declare the global variable G_TAP_COBOL_OPT_PTI and store in it the name of your procedure.

When the specified global variable is not declared, the standard procedure COBOL_OPT_PTI_NAME is called from SYS.HBINLIB which assigns predefined values to the COBOL option.

NOTE:

All the names in the schema refer to the procedure names or variables that contain user procedure names, except the ones related to the COBOL option in a batch context (PTB_PREP).

That is: COBOL_OPT_PTB_NAME is a JCL member stored in the SYS.HSLLIB library, and G_TAP_COBOL_OPT_PTB is a global variable which may contain the name of a JCL member stored in a user library.

Interactive COBOL Options

```
Default Procedure  : COBOL_OPT_PTI_NAME

User Procedure Name: contained in the global variable
                     G_TAP_COBOL_OPT_PTI
```

The names, characteristics and default values of the (local) variables which may be set in the user procedure are listed below:

NAME	TYPE	LENGTH		DEFAULT
LIST	BOOL	1		1
CLIST	BOOL	1		1
EXPLIST	BOOL	1		0
MAP	BOOL	1		0
XREF	BOOL	1		1
DCLXREF	BOOL	1		0
DEBUG	BOOL	1		1
DEBUGMD	BOOL	1		-
LEVEL	NAME	4	VALUES=(L64,NSTD, L62,CBC,CBX,ANSI, DSA,NBS1,NBS2, NBS3,NBS4)	L64
OBSERV	BOOL	1		1
DIAG	NAME	3	VALUES=(IN,AFT,BEF)	IN
SILENT	BOOL	1		0
CARDID	BOOL	1		-
CASEQ	BOOL	1		1
CKSEQ	BOOL	1		1
SUBCK	BOOL	1		-
CODAPND	BOOL	1		0
DDLIST	BOOL	1		0
EXPSIZE	BOOL	1		-
TEMP	NAME	4		-
DICLIB	LIB	78		-
OBJ	NAME	1	VALUES=(0,1,A,B)	1
SUBOPT	BOOL	1		0
LSEV	NAME	1	VALUES=(F,0)	F
DSEGMAX	DEC	4	VALUES=>0	-
PSEGMAX	DEC	4	VALUES=>0	-
OBJLIST	BOOL	1		0
DICLIST	BOOL	1		0
DUMP	NAME	4	VALUES=(DATA,ALL,NO)	-

EXAMPLE:

Assume all the above default values are your COBOL options, except XREF (you require XREF=0) and DCLXREF (you require DCLXREF=1), you then must:

1. Define a procedure:

```
PROC  MY_COBOL_OPT  HIDE=-1  ... ;
LET  XREF  0 ;
LET  DCLXREF  1 ;
ENDPROC ;
```

2. Initiate your startup with:

```
MWINLIB  BIN  MY_BINLIB;
GLOBAL   G_TAP_COBOL_OPT_PTI  NAME;
LET      G_TAP_COBOL_OPT_PTI  MY_COBOL_OPT;
```

   where MY_BINLIB is the binary library in which the procedure MY_COBOL_OPT has been stored.

❑

Interactive LINK Options

```
Default Procedures : LINK_TPR_OPT_NAME

User Procedure Name: contained in the global variable
                     G_TAP_LINK_TPR_OPT
```

The names, characteristics, and default values of the (local) variables which may be set in the user procedure are listed below:

NAME	TYPE	LENGTH		DEFAULT
SM	BOOL	1		1*
COMFILE	FILE	78		-
COMFAC	BOOL	1		-
SYSDEF	LIB	78		-
SRCHLIB	NAME	5	VALUES=(FIRST,LAST)	FIRST
DUMP	NAME	4	VALUES=(DATA,ALL,NO)	-

* In the default procedure (LINK_TPR_OPT_NAME), the SM variable is set to 1. If a user procedure is substituted for this later, the SM variable is set to 0, if not assigned in the user procedure.

Cobol EJR Options

```
Default   Procedure: EJR_COBOL_TPR_OPT_NAME

User Procedure Name: contained in the global variable
                     G_TAP_EJR_COBOL_OPT
```

The names, characteristics, and default values of the (local) variables which may be set in the user procedure are listed below:

```
    NAME          TYPE    LENGTH                    DEFAULT

    CLASS         NAME       1  VALUES=&A              -
    PRIORITY      DEC        1  VALUES=0<=*<=7         -
    HOLDOUT       BOOL       1                           1
    DEST          CHAR      17                         -
    HOLD          BOOL       1                         -
    REPEAT        BOOL       1                         -
```

Batch COBOL Options

```
Default JCL subfile:   COBOL_OPT_PTB_NAME

User JCL subfile name: contained in the global variable
                       G_TAP_COBOL_OPT_PTB
```

The COBOL step call is stored in the COBOL_OPT_PTB_NAME subfile of the SYS.HSLLIB library. Its format is:

```
COBOL SOURCE = &1        INLIB  = (&2 &3)
      CULIB  = (&4 &5)
      PRTLIB = (&6 &7)
      LEVEL  = L64;
```

The input parameters (&1 to &7) have been computed from the user options entered at the terminal (PTB_PREP).

This subfile may be replaced by a user subfile whose name is to be stored in the G_TAP_COBOL_OPT_PTB variable (declared NAME). This subfile is assumed to belong to the user SL library declared when entering the PTB_PREP command (SLLIB parameter)

EXAMPLE:

1. Define a source member MY_BATCH_COBOL_OPT that contains:

```
COBOL SOURCE = &1
       INLIB  = (&2 &3)
       CULIB  = (&4 &5)
       PRTLIB = (&6 &7)
       LEVEL  = L64
       LIST  CLIST  XREF  MAP;        (your options)
```

   and store it in your private source library.

2. Initiate your startup with:

```
GLOBAL  G_TAP_COBOL_OPT_PTB  NAME;
LET  G_TAP_COBOL_OPT_PTB  MY_BATCH_COBOL_OPT;
```

❑

9.2.7 PROJ_MANAGER Environment

Presentation of Commands

The project management environment is composed of a main environment named PROJ_MANAGER and a subenvironment named BG_PROJ_MGER.

The BACKGROUND_PROJ_MANAGER command is used to switch from PROJ_MANAGER to BG_PROJ_MGER. A return to the main environment can be done with the RETURN_PROJ_MANAGER command.

The set of accessible commands is the same in both environments. In other words, the end-user will use the BACKGROUND_PROJ_MANAGER command only if he wishes to be presented with menu screens that contain commands to run the utilities associated with project management, such as file or catalog management. The more experienced user will avoid switching continuously from one environment to the other.

The menu screens for both environments are presented below.

```
        PROJ_MANAGER                    BG_PROJ_MGER
------------------------------|------------------------
1/4
 1  CLEAR_LIBRARY             |  1  CREATE_LINK
 2  BUILD_LIBRARY             |  2  DELETE_LINK
 3  DELETE_LIBRARY            |  3  LIST_LINK
 4  MAINTAIN_LIBRARY          |
                              |  4  CREATE_GEN
 5  MAINTAIN_COMMAND          |  5  DELETE_GEN
 6  MAINTAIN_FORM             |  6  LIST_GEN
 7  CREATE_HELP_TEXT          |  7  MODIFY_GEN
                              |  8  SIFT_GEN
                              |  9  CREATE_MT_FILE
 8  BUILD_FILE                | 10  MAINTAIN_DATA_BASE
 9  CLEAR_FILE                |
10  COMPARE_FILE              | 11  RETURN_PROJ_MANAGER
11  CREATE_FILE               |
12  DELETE_FILE               |
13  LIST_FILE                 |
14  LOAD_FILE                 |
------------------------------|------------------------
2/4
 1  PRINT_FILE                |  1  MAINTAIN_FILE
 2  SORT_FILE                 |
 3  SORT_INDEX                |  2  COMPARE_FILESET
                              |  3  CREATE_FILESET
 4  CREATE_DIR                |  4  DELETE_FILESET
 5  DELETE_DIR                |  5  EXPAND_FILESET
 6  LIST_DIR                  |  6  LIST_FILESET
                              |  7  LOAD_FILESET
 7  BACKGROUND_PROJ_MANAGER   |  8  PRINT_FILESET
                              |
                              |  9  COPY_CATALOG
                              | 10  CREATE_CATALOG
                              | 11  DELETE_CATALOG
                              | 12  LIST_CATALOG
                              | 13  MODIFY_CATALOG
------------------------------|------------------------
```

```
3/4
 1  ENTER_JOB_REQUEST          |  1  ENTER_JOB_REQUEST
                               |
 2  CANCEL_JOB                 |  2  CANCEL_JOB
 3  HOLD_JOB                   |  3  HOLD_JOB
 4  RELEASE_JOB                |  4  RELEASE_JOB
                               |
 5  CANCEL_OUTPUT              |  5  CANCEL_OUTPUT
 6  HOLD_OUTPUT                |  6  HOLD_OUTPUT
 7  RELEASE_OUTPUT             |  7  RELEASE_OUTPUT
                               |
 8  BYE                        |  8  BYE
-------------------------------|------------------------

4/4
 1  ALTER_INPUT                |  1  ALTER_INPUT
                               |
 2  DISPLAY_CONFIGURATION      |  2 DISPLAY_CONFIGURATION
 3  DISPLAY_JOB                |  3  DISPLAY_JOB
 4  DISPLAY_LOAD               |  4  DISPLAY_LOAD
 5  DISPLAY_OUTPUT             |  5  DISPLAY_OUTPUT
                               |
 6  DISPLAY_PROFILE            |   6  DISPLAY_PROFILE
 7  MODIFY_PROFILE             |  7  MODIFY_PROFILE
                               |
 8  DPRINT                     |  8  DPRINT
 9  LET                        |  9  LET
10  MAIL                       | 10  MAIL
11  SEND                       | 11  SEND
                               |
12  TUTORIAL                   | 12  TUTORIAL
-------------------------------|------------------------
```

Hidden Commands

The commands that are not accessible in the PROJ_MANAGER environment are related to file and volume management, or operations which are strictly reserved to the System Administrator. The non-accessible commands are listed below.

```
CLEAR_VOLUME
CREATE_QUOTA_FILE
DELETE_QUOTA_FILE
INVALIDATE_QUOTA_FILE
MAINTAIN_CATALOG
MAINTAIN_MIGRATION
MAINTAIN_QUOTA
MAINTAIN_VOLUME
MODIFY_ACL
MODIFY_CATSPACE
```

```
MODIFY_DISK
MODIFY_FILE_SPACE
MODIFY_FILE_STATUS
MODIFY_IO_CACHE
MODIFY_QUOTA_FILE
PREPARE_DISK
PREPARE_DISKETTE
PREPARE_TAPE
PREPARE_VOLUME
RESTORE_CATALOG
SAVE_CATALOG
VALIDATE_QUOTA_FILE
```

9.2.7.1 The INIT_TDS_MGT_FILE Procedure

We describe here the INIT_TDS_MGT_FILE (ITMF) procedure and its associated items (source subfiles, load module, etc).

The use of this procedure is documented in the Help texts which are available at each level (environment, command and parameter).

Description of the TDS Session Log File

This file is named tds-name_MNGT and is located on a resident disk. The file must first be created using the following allocation parameters:

```
BUILD_FILE  FILE=tds-name_MNGT$RES
            FILESTAT=UNCAT
            UFAS=SEQ
            CISIZE=512
            SIZE=2
            UNIT=TRACK        (for VBO)
            UNIT=BLOCK        (for FBO)
            RECSIZE=72;
```

The System Administrator will find in the PROJ_MANAGER standard environment the elements required to initiate and update this file (the procedure INIT_TDS_MGT_FILE) for the information concerning him. This information defines the time interval during which tests are authorized and the user (who is the only one) who can start TDS sessions.

Tests are authorized within the limits of two dates in the form yy/mm/dd/hh/mm, giving an upper and lower limit. If the days are not specified, the hours define daily intervals. If no user name is given, all users may start TDS if their catalogs allow it.

The following are found in the part of the file not accessible to the project manager:

- a TDS status indicator:

  0 means TDS has not been started
  1 means TDS has been started
  2 means TDS is active.

- a switch status indicator:

  0 means no switch valid
  1 means switch valid

- a search path indicator:

  0 means order defined by JOBLIB (LIB1/LIB2)
  1 means inverse order (LIB2/LIB1)

- the name of the MASTER operator
- the name of the last user by whom the file was accessed in update mode
- the date and time of this access (this information is not currently used).

The record format of the tds-name_MNGT file is:

```
Character Positions            Description

      1 - 6        lower date limit for tests
      7 - 10       lower time limit for tests
     11 - 16       upper date limit for tests
     17 - 20       upper time limit for tests
     21 - 32       name of user authorized to start TDS
     33 - 35       not used
        36         TDS status indicator
        37         switch status indicator
        38         search path indicator
     39 - 50       name of MASTER operator
     51 - 62       name of last user to access file in
                    update mode
     63 - 68       date of last access (not used)
     69 - 72       time of last access (not used)
```

This file is accessed through a load module (H_TAP_TDSFILE_MGT) started by EXEC_PG. Dialog between these is performed using two global variables:

```
G_TAP_REQUEST          to prepare requests
G_TAP_ANSWER           to determine replies.
```

The different G_TAP_REQUEST codes used are as follows:

0 request for contents of the article in the variable G_TAP_UA

1 request to establish a test context

2 TDS is active (indicator set to 2)

3 a switch has ended (indicator set to 0)

4 end of TDS session (resetting of session log information to initial values)

5 request to update information relating to test conditions

6 user, when informed that a switch is required, does not want it to be executed (the indicator concerning the switch and the search order are inverted).

The different G_TAP_ANSWER codes used are the following:

1 TDS has not yet been started

2 TDS is active (a switch is required)

6 file not found

7 after 10 attempts, file could not be read (permanently locked by other users: attempts at an interval of a number of milliseconds contained in the global variable G_TAP_WAIT)

9 TDS is being started

10 switch being executed

11 end of session requested by a user other than the MASTER operator

12 file empty.

9.2.8 FILE_VOLUME Environment

Presentation of Commands

The file and volume environment is composed of a main environment named FILE_VOLUME and three subenvironments named BG_FILE, BG_CATALOG and BG_VOLUME.

The commands that enable you to switch from FILE_VOLUME to these three subenvironments are BACKGROUND_FILE, BACKGROUND_CATALOG and BACKGROUND_VOLUME respectively. A return to the main environment can be done with the RETURN_FILE_VOLUME command.

The set of accessible commands is the same in the main environment and the subenvironments.

The main environment FILE_VOLUME presents in the menu screens the basic commands that are available to the person in charge of files and volumes: creation, deletion, modification and the maintenance of files and libraries. The three subenvironments present the whole set of commands related to files (files, filesets, file space, libraries), catalogs (catalogs, acl, link, gen, dir, catspace), and volume (disks, tapes, diskettes).

The menu screens for the FILE_VOLUME environment and its subenvironments are presented below.

```
     FILE_VOLUME                |     BG_VOLUME
--------------------------------|---------------------
1/3
  1  BUILD_LIBRARY              |  1  MODIFY_DISK
  2  CLEAR_LIBRARY              |
  3  DELETE_LIBRARY             |
                                |  2  PREPARE_DISK
  4  BUILD_FILE                 |  3  PREPARE_DISKETTE
  5  CLEAR_FILE                 |  4  PREPARE_TAPE
  6  COMPARE_FILE               |  5  PREPARE_VOLUME
  7  COPY_FILE                  |
  8  CREATE_FILE                |  6  CLEAR_VOLUME
  9  DELETE_FILE                |  7  LIST_VOLUME
 10  LIST_FILE                  |  8  MAINTAIN_VOLUME
 11  LOAD_FILE                  |
 12  PRINT_FILE                 |  9  RETURN_FILE_VOLUME
 13  RESTORE_FILE               |
 14  SAVE_FILE                  |
                                |
 15  CREATE_DK_FILE             |
 16  CREATE_MT_FILE             |
--------------------------------|-------------------------
```

```
2/3
  1  BACKGROUND_CATALOG        |  1  ENTER_JOB_REQUEST
  2  BACKGROUND_FILE           |
  3  BACKGROUND_VOLUME         |  2  CANCEL_JOB
                               |  3  HOLD_JOB
  4  ENTER_JOB_REQUEST         |  4  RELEASE_JOB
                               |
  5  CANCEL_JOB                |  5  CANCEL_OUTPUT
  6  HOLD_JOB                  |  6  HOLD_OUTPUT
  7  RELEASE_JOB               |  7  RELEASE_OUTPUT
                               |
  8  CANCEL_OUTPUT             |  8  BYE
  9  HOLD_OUTPUT               |
 10  RELEASE_OUTPUT            |
                               |
 11  BYE                       |
-------------------------------|-----------------------
3/3
  1  ALTER_INPUT               |  1  ALTER_INPUT
                               |
  2  DISPLAY_CONFIGURATION     |  2  DISPLAY_CONFIGURATION
  3  DISPLAY_JOB               |  3  DISPLAY_JOB
  4  DISPLAY_LOAD              |  4  DISPLAY_LOAD
  5  DISPLAY_OUTPUT            |  5  DISPLAY_OUTPUT
                               |
  6  DISPLAY_PROFILE           |  6  DISPLAY_PROFILE
  7  MODIFY_PROFILE            |  7  MODIFY_PROFILE
                               |
  8  DPRINT                    |  8  DPRINT
  9  LET                       |  9  LET
 10  MAIL                      | 10  MAIL
 11  SEND                      | 11  SEND
                               |
 12  TUTORIAL                  | 12  TUTORIAL
 ------------------------------|------------------------
```

```
     BG_CATALOG                 |     BG_FILE
--------------------------------|-------------------------
1/4
  1  LIST_ACL                   |  1  BUILD_LIBRARY
  2  MODIFY_ACL                 |  2  CLEAR_LIBRARY
                                |  3  DELETE_LIBRARY
  3  CREATE_LINK                |
  4  DELETE_LINK                |  4  BUILD_FILE
  5  LIST_GEN                   |  5  CLEAR_FILE
                                |  6  COMPARE_FILE
  6  CREATE_GEN                 |  7  COPY_FILE
  7  DELETE_GEN                 |  8  CREATE_FILE
  8  LIST_GEN                   |  9  DELETE_FILE
  9  MODIFY_GEN                 | 10  LIST_FILE
 10  SHIFT_GEN                  | 11  LOAD_FILE
                                | 12  MERGE_FILE
 11  CREATE_DIR                 | 13  MODIFY_FILE
 12  DELETE_DIR                 | 14  PRINT_FILE
 13  LIST_DIR                   | 15  RESTORE_FILE
                                | 16  SAVE_FILE
                                | 17  SORT_FILE
--------------------------------|-------------------------
2/4
  1  ATTACH_CATALOG             |  1  MAINTAIN_FILE
  2  COPY_CATALOG               |  2  MODIFY_FILE_STATUS
  3  CREATE_CATALOG             |
  4  DELETE_CATALOG             |  3  COMPARE_FILESET
  5  LIST_CATALOG               |  4  COPY_FILESET
  6  MODIFY_CATALOG             |  5  CREATE_FILESET
  7  RESTORE_CATALOG            |  6  DELETE_FILESET
  8  SAVE_CATALOG               |  7  EXPAND_FILESET
                                |  8  LIST_FILESET
  9  LIST_CATSPACE              |  9  LOAD_FILESET
 10  MODIFY_CATSPACE            | 10  PRINT_FILESET
                                | 11  RESTORE_FILESET
 11  RETURN_FILE_VOLUME         | 12  SAVE_FILESET
                                |
                                | 13  LIST_FILE_SPACE
                                | 14  MODIFY_FILE_SPACE
                                |
                                | 15 RETURN_FILE_VOLUME
--------------------------------|-------------------------
```

```
     BG_CATALOG                |     BG_FILE
-------------------------------|------------------------
3/4
  1  ENTER_JOB_REQUEST         |  1  ENTER_JOB_REQUEST
                               |
  2  CANCEL_JOB                |  2  CANCEL_JOB
  3  HOLD_JOB                  |  3  HOLD_JOB
  4  RELEASE_JOB               |  4  RELEASE_JOB
                               |
  5  CANCEL_OUTPUT             |  5  CANCEL_OUTPUT
  6  HOLD_OUTPUT               |  6  HOLD_OUTPUT
  7  RELEASE_OUTPUT            |  7  RELEASE_OUTPUT
                               |
  8  BYE                       |  8  BYE
-------------------------------|------------------------
4/4
  1  ALTER_INPUT               |  1  ALTER_INPUT
                               |
  2  DISPLAY_CONFIGURATION     |  2  DISPLAY_CONFIGURATION
  3  DISPLAY_JOB               |  3  DISPLAY_JOB
  4  DISPLAY_LOAD              |  4  DISPLAY_LOAD
  5  DISPLAY_OUTPUT            |  5  DISPLAY_OUTPUT
                               |
  6  DISPLAY_PROFILE           |  6  DISPLAY_PROFILE
  7  MODIFY_PROFILE            |  7  MODIFY_PROFILE
                               |
  8  DPRINT                    |  8  DPRINT
  9  LET                       |  9  LET
 10  MAIL                      | 10  MAIL
 11  SEND                      | 11  SEND
                               |
 12  TUTORIAL                  | 12 TUTORIAL
-------------------------------|------------------------
```

Hidden Commands

The whole of the set of commands (with the exception of the commands reserved to the System Administrator, namely CREATE_SYSTEM_FILES and MAINTAIN_CATALOG) is accessible to the user working under the FILE_VOLUME environment, or one of its subenvironments.

9.2.9 SYSADMIN Environment

Presentation of Commands

The System Administration environment is composed of a main environment named SYSADMIN and three subenvironments, similar to those available in the FILE_VOLUME environment, named BG_FILE, BG_CATALOG, and BG_VOLUME.

The commands that enable you to switch from SYSADMIN to these three subenvironments are BACKGROUND_FILE, BACKGROUND_CATALOG, and BACKGROUND_VOLUME respectively. A return to the main environment can be done with the RETURN_SYSADMIN command.

The whole set of commands is available from both the SYSADMIN environment and its three subenvironments.

The main environment SYSADMIN presents in the menu screens the commands for which the System Administrator is in charge:

- defining users, projects, and billings in the Site Catalog
- installing the system and tailoring the system files
- creating commands
- producing documentation.

The menu screens for the SYSADMIN environment are presented below. The three subenvironments have the same visibility as those of the FILE_VOLUME environment.

```
        SYSADMIN
--------------------------------
1/4 1  BUILD_SYSTEM
    2  MAINTAIN_CATALOG
    3  MODIFY_DISK
    4  PREPARE_DISK
    5  PREPARE_TAPE
    6  CLEAR_VOLUME
    7  LIST_VOLUME
    8  MAINTAIN_VOLUME
    9  MAINTAIN_COMMAND
   10  CREATE_HELP_TEXT
   11  LIST_ACL
   12  MODIFY_ACL
--------------------------------
```

```
2/4 1  CREATE_FILE
    2  DELETE_FILE
    3  LIST_FILE
    4  RESTORE_FILE
    5  SAVE_FILE
    6  CREATE_DIR
    7  DELETE_DIR
    8  LIST_DIR
    9  BACKGROUND_CATALOG_
   10  BACKGROUND_FILE_
   11  BACKGROUND_VOLUME_
--------------------------------
3/4 1  ENTER_LIBTRANS_REQUEST
    2  ENTER_JOB_REQUEST
    3  ENTER_FILETRANS_REQ
    4  CANCEL_JOB
    5  HOLD_JOB
    6  RELEASE_JOB
    7  CANCEL_OUTPUT
    8  HOLD_OUTPUT
    9  RELEASE_OUTPUT
   10  CANCEL_USER_REQ
   11  BYE
--------------------------------
4/4 1  DISPLAY_SYSTEM_STATUS
    2  DISPLAY_CONFIGURATION
    3  DISPLAY_LOAD
    4  DISPLAY_TRACE
    5  DISPLAY_JOB
    6  DISPLAY_OUTPUT
    7  DISPLAY_USER_REQ
    8  MAIL
    9  SEND
--------------------------------
```

Hidden Commands

The whole set of commands is accessible from the SYSADMIN environment and its subenvironments.

9.2.10 MAINTAIN_LIBRARY Domains

The presentation of the MAINTAIN_LIBRARY domains is the same for all the environments. This presentation is given below. All the MAINTAIN_LIBRARY commands that do not appear on the menus are hidden (therefore accessible).

```
                                |     MAINTAIN_CU_LIBRARY
                                |     MAINTAIN_LM_LIBRARY
     MAINTAIN_SL_LIBRARY        |     MAINTAIN_BIN_LIBRARY
-------------------------------|-------------------------
1/4                             1/3
  1  COMPARE                   |  1  COPY
  2  COPY                      |  2  DELETE
  3  CREATE                     |  3  DISPLAY
  4  DISPLAY                   |  4  LIB
  5  DELETE                     |  5  LIST
  6  EDIT                      |
  7  FSE                       |  6  INFILE
  8  LIB                       |  7  INLIB1
  9  LIST                      |  8  INLIB2
 10  RENAME                     |  9  INLIB3
 11  RENUMBER                  | 10  OUTFILE
 12  PRINT                     |
                               | 11  SAVE
 13  QUIT                      | 12  RESTORE
                               |
                               | 13  QUIT
-------------------------------|-------------------------
2/4                             2/3
  1  INFILE                     |  1  ENTER_JOB_REQUEST
  2  INLIB1                    |
  3  INLIB2                     |  2  CANCEL_JOB
  4  INLIB3                     |  3  HOLD_JOB
  5  OUTFILE                   |  4  RELEASE_JOB
                               |
  6  RESTORE                   |  5  CANCEL_OUTPUT
  7  SAVE                      |  6  HOLD_OUTPUT
                               |  7  RELEASE_OUTPUT
                               |
                               |  8  QUIT
 -------------------------------|-------------------------
                               |  MAINTAIN_CU_LIBRARY
                               |  MAINTAIN_LM_LIBRARY
     MAINTAIN_SL_LIBRARY       |  MAINTAIN_BIN_LIBRARY
-------------------------------|-------------------------
```

```
3/4                              3/3
  1  ENTER_JOB_REQUEST          |  1  ALTER_INPUT
                                |
  2  CANCEL_JOB                 |  2  DISPLAY_CONFIGURATION
  3  HOLD_JOB                   |  3  DISPLAY_JOB
  4  RELEASE_JOB                |  4  DISPLAY_LOAD
                                |  5  DISPLAY_OUTPUT
  5  CANCEL_OUTPUT              |
  6  HOLD_OUTPUT                |  6  DISPLAY_PROFILE
  7  RELEASE_OUTPUT             |  7  MODIFY_PROFILE
                                |
  8  BYE                        |  8  DPRINT
                                |  9  LET
                                | 10  MAIL
                                | 11  SEND
                                |
                                | 12  TUTORIAL
--------------------------------|------------------------
4/4
  1  ALTER_INPUT                |
                                |
  2  DISPLAY_CONFIGURATION      |
  3  DISPLAY_JOB                |
  4  DISPLAY_LOAD               |
  5  DISPLAY_OUTPUT             |
                                |
  6  DISPLAY_PROFILE            |
  7  MODIFY_PROFILE             |
                                |
  8  DPRINT                     |
  9  LET                        |
 10  MAIL                       |
 11  SEND                       |
                                |
 12  TUTORIAL                   |
--------------------------------|
```

9.2.11 MAIN_FULL and MAIN_REDUCED Environments

All MAIN operator commands are available under the MAIN_FULL environment. Most are also available under the MAIN_REDUCED environment. The following commands are not in the MAIN_REDUCED environment:

```
CONNECT_DIMENSION
CONNECT_LOAD              DISCONNECT_DIMENSION
CREATE_DIMENSION          DISCONNECT_LOAD
DELETE_DIMENSION          START_LOAD
MODIFY_DIMENSION          MODIFY_LOAD
MODIFY_XL_CLASS           TERMINATE_LOAD
DISPLAY_XL_CLASS
```

The absence of these commands makes the MAIN_REDUCED environment more suited to ordinary operation, where the System Administrator prefers not have operator control over scheduling and regulation. However, if any of these commands needs to be used in the system startup, the OPERATOR project must not have the MAIN_REDUCED environment as its default environment.

9.3 Standard Environment Installation

IMPORTANT:
The standard environments installations below are meant to be used as general guides and not for a particular case.

The System Administrator is in charge of the standard environments. The installation is performed by running a batch job after the standard procedures have been generated in SYS.HSLLIB, and after each new delivery of SYS.HBINLIB. This job generates all the standard environments at once.

The job, named STANDARD_ENVT, is stored in the SYS.HSLLIB library.

The STANDARD_ENVT job contains commands that generate the following items:

The environments	in SITE.CATALOG
The access rights and priorities of the GCL procedures	in SYS.HBINLIB
The non-standard procedures	in SYS.HBINLIB
The associated Help texts	in SITE.HELP
The source subfiles	in SYS.HSLLIB

All these items are created using the specific commands that manage these object types.

9.3.1 MAINTAIN_COMMAND Step

The MAINTAIN_COMMAND step of the STANDARD_ENVT job does the following:

- creates (using the CREATE command) the GCL procedures in the IOF domain of SYS.HBINLIB which are specific to the standard environments, such as:

```
PREPARE_PROGRAM_INT         (PROGRAM_PREP environment)
BACKGROUND_FILE             (FILE_VOLUME environment)
PREPARE_TPR_BATCH           (TDSAPPL_PREP environment)
```

- sets (using the MODIFY_ACCESS command) the access rights and priorities of all the commands of SYS.HBINLIB
- creates (using the ENVT command) the standard environments and their subenvironments in the SITE.CATALOG.

9.3.2 MAINTAIN_LIBRARY Step

The MAINTAIN_LIBRARY step creates in SYS.HSLLIB all the subfiles that are used in relation with the non-standard procedures that are generated in the preceding step.

These subfiles are used as JCL submitted by the EJR directive, or command files used as COMFILEs or invoked by an ALTER_INPUT directive.

9.3.3 MAINTAIN_LIBRARY Step and HELPGEN Step

The MAINTAIN_LIBRARY step followed by the HELPGEN step create the Help texts that are associated with the standard environments and all the commands and keywords that belong to them.

9.3.4 Running the STANDARD_ENVT Job

As a new delivery of the SYS.HBINLIB library resets the access rights and priorities of the standard commands, and then deletes the non-standard procedures used by the standard environments, the System Administrator must submit the adapted STANDARD_ENVT generation job each time a new technical status is delivered.

Specific operating practices that concern the JCL for the installation of the standard environments are given later in this section.

9.4 Creating New Environments

The standard environments do not cover all the situations of particular users of all sites. They are delivered as prototype environments to be used as models for building environments that actually fit the site users classes.

For example, one such user class might not be allowed to compile programs interactively. Another group of users might only use the GPL programming language. To fit these requirements, new environments may be created from the PROGRAM_PREP environment.

As another example, a System Administrator may think the FILE_VOLUME environment is not suitable for the file and volume management on his site, and that several classes of privileged users exist that are allowed to manage the files and the volumes. This may lead to the definition of three environments rather than one, with three levels of rights on the commands.

All these operations are performed by the System Administrator. Some specific problems are described in Appendix B.

9.4.1 Creating Environments from Standard Environments

This is the lowest level for creating new environments. It consists of updating a part of an environment without modifying its design.

The changes may consist of the following:

- CHANGING THE PRESENTATION: For example, changing the priority of a command which leads to a new menu screen presentation, changing a command name or a Help text, or making a visible command hidden.
- MODIFYING THE ACCESSIBLE SET OF COMMANDS: For example, forbidding access to the interactive compilers or some other commands or, conversely making the set of accessible commands larger.
- SUBSTITUTING ONE FUNCTION FOR ANOTHER: For example, changing the call to a COBOL compiler to a call to a FORTRAN compiler, or modifying the default parameters of the COBOL statement.
- MODIFYING THE OPERATION OF NON-STANDARD COMMANDS: That is, re-programming the GCL procedures that are delivered with the standard environments.

These modifications lead to one of two situations:

- One standard environment is changed to another. For example, PROGRAM-PREP is modified in order not to allow access to the GPL command. This is the simplest case.
- One standard environment is split into several environments (for example PROGRAM_PREP) each of which is adapted to the particular needs of a class of users.

  For example:

  – PROGRAM_PREP without interactive compilation

  – PROGRAM_PREP using COBOL

  – PROGRAM_PREP with some commands for project management

  – etc.

All the combinations that match the needs of some particular classes of user may exist.

The means for performing these modifications are described under "Standard Families and Users' Families".

9.4.2 Creating Environments from Standard Families

Another approach is to build new environments using directly the command groups that have been defined as standard families. These families have been defined in relation to the standard environments but they may be freely collated in order to constitute a new environment.

This is the easiest way of defining new environments. Perhaps the standard families will not be sufficient and you will have to define some non-standard families.

The standard families and their contents are described in Appendix A of this manual.

9.4.3 Creating Environments from Nothing

If nothing in certain of the standard environments is adapted to your needs, you are free not to use them.You need not generate the standard environments, nor set the relationships between them and the projects.

You may then fully redefine the environments that may be adapted to your use. The standard families themselves are also delivered as prototypes, and nothing prevents you from redefining them.

Note that dividing the command set into families does not partition the command set. A command may belong to one or more families, and the environments give access to one or more families.

As there are 256 families available, it is better to define new families (non-standard families) rather than redefine the standard families. At present, the families numbered 1 to 52 are used as the standard families. It is advisable to keep the numbers 1 to 99 to refer to standard families.

9.5 Operating Practices

9.5.1 JCL Management

Attention must be paid to the management of the standard environment generation job because:

- The JCL may be modified in order to take into account the particular needs of the site.
- New versions of the domains of SYS.HBINLIB are delivered periodically.
- New versions of the STANDARD_ENVT job may be made available.

Several situations can arise:

- If the standard environments (as delivered with the system) are used without any change , no problems arise. (This is not recommended in general because your system should be configured for your needs and these needs will change over time.) The System Administrator need only run the job each time a new version of SYS.HBINLIB is delivered, and to set the relationships between the projects that exist on the site and the environments. The PROJ commands which manage the relationships may be recorded in another job which is submitted when the need arises.
- If the standard environments are modified, the System Administrator has to copy the generation JCL and modify the copy. The modifications may either be inserted in the standard JCL copy or appended to the same member or another member which is invoked after the standard one. When possible the latter solution is the easiest to use, the modification of the standard JCL and the System Administrator-created JCL being clearly separated. This System Administrator-created JCL also contains the PROJ commands that establish the connections between the existing projects and the environments.

NOTE:
A new delivery of the STANDARD_ENVT job contains the description of the modifications as comment lines.

A project that is granted access to an environment must also be granted access to all its subenvironments. If it is not, users connected to the project that does not have access to the subenvironments will receive error messages when trying to access the background environments.

EXAMPLES OF PROJ COMMANDS ARE GIVEN BELOW:

```
PROJ  X   (PROGRAM_PREP, BG_PROG_PREP);

PROJ  Y   (PROGRAM_PREP, BG_PROG_PREP,
           TDSAPPL_PREP, BG_TDS_PREP,
           PROJ_MANAGER, BG_PROJ_MGER);

PROJ  Z   (FILE_VOLUME, BG_FILE, BG_CATALOG,
           BG_VOLUME, SYSADMIN, BG_FILE_,
           BG_CATALOG_, BG_VOLUME_);
```

❑

9.5.2 Advice on Designing Environments

9.5.2.1 Naming Conventions

You will use items such as local and global variables, source subfiles and Help texts when you define GCL commands.You must avoid naming these items with names that may conflict with names defined by the user of these GCL commands.

The conventions used for the standard environments are shown below (each environment is given a three character indicative):

```
PGF     for PROGRAM_PREP
TAP     for TDSAPPL_PREP
PJM     for PROJ_MANAGER
FVM     for FILE_VOLUME
SYS     for SYSADMIN
```

Global variables, local variables and source subfiles are named G_xxx_name, L_xxx_name and C_xxx_name respectively, where xxx is replaced by the environment indicative.

Help texts are named as follows:

```
environment_name    for environment level texts
H_xxx_cmd           for command level texts
H_xxx_cmdkwd        for keyword level texts
```

where cmd and cmdkwd represent the command and keyword names respectively.

9.5.2.2 Environments and Subenvironments

Several levels of environments may be defined, using the MWENVT command inside a procedure (refer to Appendix B). This provides ease-of-use for a novice user, but frequent switching between the levels of environments should be avoided as it is expensive in time.

A good solution is to make the subenvironments available for novice users and make the commands specific to the subenvironments accessible from the main environment for the more experienced users.

9.5.2.3 Effect of a Break

The non-standard procedures associated with the standard environments are generated with the LOCK parameter (LOCK=1) so that the break is taken into account by the processors started from these procedures.

The effect of a break during the execution of a non-locked GCL procedure is that the break would be treated by GCL, which would display the procedure name and the current line number.

9.5.3 Standard Families and Users' Families

The items that the System Administrator will use to build an environment are the command families.

As mentioned above, 52 standard families are provided in order to define the standard environments and help the user define new environments.

The standard families offer subsets of the complete set of commands. It is often insufficient simply to define new environments given that your problem is, for example, to make certain commands hidden or inaccessible. If a family does not already exist that conforms to your needs, the solution is to create a new family.

9.5.3.1 Advice on Defining Command Families

Command families do not constitute partitions of the complete command set. The command families may overlap and can be thought of as non-disjoint sets.

A single family may contain commands that come from different domains. For example, a family may be defined as containing the commands MAINTAIN_COMMAND, DPRINT and PRINT that belong to the domains IOF, H_NOCTX and MAINTAIN_SL_LIBRARY respectively.

It is important to ensure the internal coherence of each family. That is:

- At domain level, a family which grants access to MAINTAIN_LIBRARY, for example, must also grant access to at least one command of the invoked domain (e.g., the QUIT command of MAINTAIN_SL_LIBRARY).
- At the 'call' level, a family which contains a command that calls (CALL, VCALL, SCALL statement) another command must contain this called command. For example, the family which contains the PREPARE_PROGRAM_INT command must also contain the COBOL, LINK_PG and EXEC_PG.

There are enough families available to avoid having to give too strict a definition. In particular, some families may be only very slightly different from other families, or some families may not be very useful for the definition of a specific environment, but make the definition of a future environment easier.

9.5.3.2 Example

The following simple example shows that no general algorithm can be given and that it is important that the internal coherence of each family be treated independently of any other families.

Suppose an environment is to be created which gives access to the commands A, D, E and F and for which A is to be visible and D, E and F hidden. Suppose also that there are two standard families, 1 and 2, which contain:

```
family 1 :    A, B, C  as visible commands
              D, E, F  as hidden commands

family 2 :    no visible commands
              D, E, F  as hidden commands
```

Obviously, family 1 cannot be part of the environment to be created, since commands B and C are not to be accessible.

Thus a new family, say 100, has to be defined in which only command A is visible.

Family 100 may be defined either as containing only A as a visible command and no hidden commands, or as containing only A as a visible command and D, E, and F as hidden commands.

If you choose the first definition, the new environment must give access to families 2 and 100. If you choose the second definition, the new environment must give access to family 100 only.

It is important that family 100 be internally coherent. Therefore, for your purposes, the best solution is to define family 100 as containing A as a visible command. The new environment must then give access to families 2 and 100.

9.5.4 Standard Domains

In order to extend the set of standard commands which are available to the terminal users, the System Administrator has the facility to define new commands that belong to the standard domains. Most of them will belong either to the IOF domain (commands accessible at system level) or to the H_NOCTX domain (directives available at all levels).

Users may also define their own domains in relation to private programs (for further details refer to the *GCL Programmer's Manual*).

To help you in defining new commands, the standard domain names are listed below with the commands that permit you to enter them:

Table 9-3. Standard Domains (1/2)

COMMANDS	STANDARD DOMAINS
System level	IOF
Directives	H_NOCTX
Main Operator Commands	MAIN
CREATE_FILE CREATE_FILESET	CREATE_BFAS_NONE CREATE_BFAS_SEQ CREATE_LIBRARY CREATE_LIBRARY_FBO CREATE_UFAS_INDEXED CREATE_UFAS_INDEXED_FBO CREATE_UFAS_RANDOM CREATE_UFAS_SEQ_REL CREATE_UFAS_SEQ_REL_FBO
CRPMM (under MAINTAIN_SYSTEM processor)	CREATE_BPMM_FUNCTION
CREATE_COMPLEX_GENERATION (Iin MAIN domain)	CXGEN
CREATE_network_GENERATION (Iin MAIN domain)	NG
CREATE_SYSTEM_FILE (under GIUF processor)	TL_DAT TL_DSA TL_FW TL_GCOS TL_GSF TL_OLTD TL_SESSION
Fileset utilities (in driven mode)	DMU_FILESET
DEBUG	PCF
DPAN	DPAN DPANCRTR DPANPRTR
EDIT	EDIT
ENTER_GIUF	GIUG
ENTER_RMOS	RMOS
FSE (system level)	F_S_E
FSE (under the LIBMAINT_SL processor)	FSE
Expression "$*BRK" (break)	H_BREAK
CREATE (under CMDMGT processor)	H_GCL
IQS	IQS
MAINTAIN_AUDIT7	AUDIT7
MAINTAIN_CATALOG	CATMAINT
MAINTAIN_COMMAND	CMDMGT

Table 9-4. Standard Domains (2/2)

COMMANDS	STANDARD DOMAINS
MAINTAIN_DATA_DESCRIPTION	MAINTAIN_DATA_DESCRIPTION
MAINTAIN_FILE	MAINTAIN_FILE MAINTAIN_FILE_FBO
MAINTAIN_FORM	FORMGEN
MAINTAIN_JAS	MAINTAIN_JAS
MAINTAIN_LIBRARY BIN	LIBMAINT_BIN
MAINTAIN_LIBRARY CU	LIBMAINT_CU
MAINTAIN_LIBRARY LM	LIBMAINT_LM
MAINTAIN_LIBRARY SL	LIBMAINT_SL
MAINTAIN_LIBRARY SM	LIBMAINT_SM
MAINTAIN_MIGRATION	LIBMAINT_MIGRATION
MAINTAIN_STORAGE_MANAGER (in full VOLSET FACILITY/QUOTAS context alias MAINTAIN_QUOTA	MNSTM_HPS
MAINTAIN_STORAGE MANAGER (in basic VOLSET context)	MNSTM_AP
MAINTAIN_SYSTEM	MAINTAIN_SYSTEM
MNPMM (under MAINTAIN_SYSTEM processor)	MAINTAIN_PMM
MNSYSLM (under MAINTAIN_SYSTEM processor)	MAINTAIN_SYSTEM_LM
MNSYSSM (under MAINTAIN_SYSTEM processor)	MAINTAIN_SYSTEM_SM
MNTZS (under MAINTAIN_SYSTEM processor)	MAINTAIN_TYPE)_SET
MAINTAIN_EXTENDED _BACKUP	MNXBUP
MAINTAIN_VOLUME	MAINTAIN_VOLUME MAINTAIN_VOLUME_FBO
MODIFY_PMM (under MAINTAIN_PMM processor)	MODIFY_PMM
Maintain_system update commands	MODIFY_SYSTEM_UNIT
PREPARE_TAPESET	PREPARE_TAPESET
MODIFY (under MAINTAIN_LIBRARY processor)	UPDATE
SCANNER	SCANNER
SCAM_VCAM_TRACE	SCAN_VCAM_TRACE

CAUTION:
Adding private commands or changing access (with the MDA command of MAINTAIN_COMMAND processor) on the commands of the H_GCL domain will cause **serious problems** to the system.

9.6 User Assistance

An important function of the System Administrator is to help the users define their own commands. For that purpose, he or she has to inform them about the environments, the families and the command priorities in order to permit them to make accessible and visible their own commands in the environments and in the position they wish.

It is therefore necessary that the System Administrator keep the following information up to date:

- environment definitions in terms of families
- family definitions (lists of visible and hidden commands for each domain)
- list of priorities assigned to each command.

All this information is contained in this manual. However, as already mentioned, the System Administrator may modify all the preceding items (environments, families) or create new environments and new commands.

Another function of the System Administrator is to help end-users and project managers to define their own parameters for using PROGRAM_PREP or TDSAPPL_PREP environments. Examples are COBOL options or EJR options; see the descriptions of these two environments and Appendix B in this manual.

10. Operator Management Procedures

This chapter explains the role of the System Administrator in defining operator types, startup sequences, and system restart procedures.

Topics include:

- Operator access rights
- Operator responsibilities
- Creating system, site, and project startups

10.1 Operator Types

For a typical DPS 7000 installation there are three distinct types of operator:

- the MAIN operator
- the STATION operator
- the STANDARD operator

In terms of access to commands:

- the MAIN operator is the most privileged user. He or she has the right to override the class limits and priorities of jobs submitted by IOF users, and has reserved access to a very wide range of system control commands.
- the STATION operator is a privileged IOF user who has reserved access to a range of device management commands, as well as access to the GCL commands of the ordinary IOF user.
- the STANDARD operator is an ordinary IOF user with access to one or more GCL environments, depending on the project.

It is the System Administrator's responsibility to define the role and access rights of each operator type. These roles will vary greatly according to the size and nature of the installation, the personnel available, and the extent to which automation (e.g. DOF 7-RM, DOF 7-PO, etc.) has been introduced.

In the following paragraphs, however, we present a fairly typical list of tasks that might be performed by each category of operator, along with the type of command sets that each would require.

10.1.1 The MAIN Operator

A MAIN operator is any user who is registered in the Site Catalog as belonging to a project for which the MAIN parameter has been specified. Consequently, there may be several MAIN operators logged on at the same time on a given system, either locally or through the communications network. The operating tasks can be apportioned among them by means of the command environments and by using filters for system messages.

The MAIN operator category is usually the most privileged category with the widest access to command sets.

The responsibilities of a MAIN operator include the following:

1. Powering up the hardware of the DPS 7000 system.
2. Performing hardware initialization by loading the firmware from disk.
3. Performing the Initial Storage Load (ISL), by loading the GCOS operating system from disk.
4. During normal system operation, the operator is responsible for the initiation of locally-submitted jobs, and for monitoring all jobs (whether locally or remotely submitted) from submission to termination. This includes:
   - introducing jobs, in the correct order as defined within the installation,
   - ensuring that the machine maintains a good mix of work to be done, controlling printed job output, which is "spooled" until the operator starts the Output Writer, controlling printing activity and, more generally, the activity of local writers and writers working for an unattended station.
5. Controlling and administering a communications network using the following functions:
   - (system and) network generation
   - maintenance services
   - gathering statistics
   - surveying and managing network components; i.e.:
     - physical units such as lines, modems and terminals
     - logical entities such as programs and connections
     - DSA primary network objects.
   - software generation for front-end processor loading and dumping
   - trace operations on:
     - TNS
     - VCAM
     - FEPS
     - OCS

6. Performing the necessary peripheral operations. This includes mounting cartridges, magnetic tape reels, printer stationary, and print belts as and when requested to do so by the system. The operator may anticipate these system requests by premounting media which he or she knows is about to be requested.
7. Controlling (under Operations Management supervision) certain system files, both through use of the relevant operator commands and by calling utilities to deal with these files; for example, the utility PRLOG for listing the contents of the system error log file SYS.LOG, and the utility PRLOGC for listing the console log file SYS.LOGC.
8. Performing the recovery procedure appropriate to any hardware or software incidents which may arise during system operation. In such an event, the operator must carefully note the incident's circumstances, especially any error indications on the Operator console.
9. At the appropriate time, closing the system down and powering it off.
10. Preventive Maintenance.
11. General installation-defined responsibilities such as the organization of disks, tapes, cartridges, paper, disk and file save, etc.

For further details of the above operations, refer to the *System Operator's Guide* and the *Network Operations Reference Manual*, which give details of the commands available to the MAIN operator, associated with the functions listed above.

MAIN Operator Command Set

MAIN operators have the exclusive use of GCL commands allowing modification of the general conditions of the operation. They are responsible for the management of hardware devices. Commands reserved for the main operator may be submitted in batch, if the JOB statement contains a project with the MAIN attribute.

MAIN operators control the peripherals of the MAIN Station; that is, those which are not allocated to a remote station. These operators may submit jobs from the console for all projects.

The MAIN operator:

- can use all the GCL commands to control the scheduling and the execution of all jobs, including those submitted from a remote station.

  is the only operator allowed to cancel another MAIN operator.
- is the only operator allowed to issue HJ and RJ commands on jobs already executing.
- can modify the attributes of a job or of an output (class, priority, etc.) with the same limitations as other projects.
- modify the default attributes of classes (if any modifiable attributes have been declared in CONFIG).
- can use the CPU management commands DXLC and MDXLC, and dimension management commands DDIM and MDDIM. If the system is equipped with Full ARM, there are the additional commands CRDIM, DLDIM, CNDIM and DISDIM available.
- can use the CPU resources manegement commands CNFUNC, DISFUNC and DFUNC on DPS 7000/MTxx.
- can use the output management commands for any output in the system, but the use of output queue management commands (MDO ALL, SOW, TOW, FO, MDC OC) is generally restricted to the output queues of the MAIN Station and of unattended stations. A MAIN operator may, however, start any spooling towards any host registered in the site list of stations (i.e., those in STTNLIST - see Chapter 8).
- can use file transfer commands such as EFTR, MDST for any station.
- can also serve as a network operator and so can use the network control commands (TTSVR, DNET, MDNET, MDTN, STSVR, etc.).
- can use all the display commands with the whole installation as the default scope.
- control of the ARS service job and migration facilities (e.g. DARS, SARS...).

10.1.2 The STATION Operator

A STATION operator is a special IOF user who is registered in the Site Catalog as belonging to a project for which the STATION attribute has been specified. He or she is responsible for the first station in the operator project's station list. If the project has more than one user-id associated with it in the Site Catalog, the first user to log on becomes the STATION operator and initiates startup for the station. Any other users subsequently log onto that station, become ordinary IOF operators and initiate only who their projects' startups.

For a given station name (for example, PUB1) the conditions necessary to become the STATION operator of that station are the following:

- The project used for log-on must have the STATION attribute.
- The station name PUB1 must be the first station of the list attached to this project.
- The station declared at log-on must be PUB1, otherwise no station must be declared so that PUB1 is the default station.
- The station PUB1 must not be currently operated by another STATION operator already logged on.
- The operator must log on under his default station.

If any of the above conditions is not satisfied, the log-on is accepted as that of a common IOF user.

The STATION operators manage output devices (printers, etc.). They may submit batch jobs from their terminal, as may any other interactive user, and from the hardware devices allocated to the station. In both cases, these jobs must be submitted for projects on which they have access rights.

The STATION operator is identified by the station name and not by the user name. SEND commands must include the station name; they will be received by the station operator whatever his user name may be.

A station with an operator is an "attended station", without an operator an "unattended station".

When a station is attended, MAIN operator rights on the station are limited to automatic rescue rights, and so do not include SOW, FO, and MDO commands.

When a station is unattended, MAIN operator rights do include SOW, TOW, MDO, FO, etc., provided the station name is specified (ST=...).

If a MAIN operator uses the command MP STATION=..., he can operate an unattended station as if he were a STATION operator. The station is still seen as "unattended", and it can be operated by several MAIN operators.

The STATION Operator Command Set

In addition to the GCL commands available to ordinary IOF users, the STATION operator has access to a specially reserved set of commands:

- DISPLAY_CONFIGURATION (DC) allows the operator to access information held by the system about the station.
- FORCE_OUTPUT (FO) enables the operator to release an output file from the READY, HOLD, or WAIT state, for delivery as soon as an appropriate output device becomes available.
- FORCE_USER_REQ (FUR) allows the operator to release a user request from the READY, HOLD, or WAIT state.
- MODIFY_CONFIGURATION (MDC OC) enables him to modify the default priority and media associated with a given output class.
- MODIFY_STATION (MDST) allows the operator to modify the station description. That is:
  - name input devices or files
  - name output destinations
  - redirect output to another station
  - alter the printing of output banners
  - request another station to accept output otherwise destined for his own station (e.g., in the event of a local output device fault).
- START_INPUT_READER (SIR) and TERMINATE_INPUT_READER (TIR) enable the operator to initiate/terminate reader jobs submitted by the EJR command.
- START_OUTPUT_WRITER (SOW) and TERMINATE_OUTPUT_WRITER (TOW) enable the operator to initiate/terminate a remote output writer.
- The commands CJ, HJ, RJ, and MDJ with the same restrictions as for the interactive user, but for all the jobs submitted from the station, either by themselves or by interactive users attached to his station. When a job is already in execution, they cannot use the RJ and HJ commands.
- The commands CO, FO, HO, RO, and MDO on the outputs of the jobs submitted from the station, and on outputs destined for the station. They are also the only ones allowed to manage the output queues of the station (using commands SOW, TOW, MDO ALL, MDST, MDC OC).

  Note that when using the MODIFY_STATION (MDST) command, the rescue station specified must belong to the station list attached to the project.
- File transfer commands for their station only.

10.1.3 The STANDARD Operator

This is an interactive or batch user under a project with the STD attribute.

When access rights are used, such a user may not submit jobs with a different user name or project name.

Standard users may use the commands CANCEL_JOB (CJ) and MODIFY_JOB (MDJ) on jobs they have submitted, and have access to the commands HOLD_JOB (HJ) and RELEASE_JOB (RJ) for jobs that have been submitted but have not yet started executing. They are given access to the commands CANCEL_OUTPUT (CO), HOLD_OUTPUT (HO), RELEASE_OUTPUT (RO), and MODIFY_OUTPUT (MDO) to manage the outputs of the jobs they have submitted; however, the use of modification commands (MDJ, MDO) is subject to existing job class limitations (maximum priorities, classes allowed for the project, stations allowed for the project, etc.)

Standard users have access to all the display commands (DISPLAY_LOAD etc.). The default scope of the display commands is the set of jobs they have submitted, but upon explicit request they may obtain any display information available in the system.

10.2 Startup Sequences

It is possible to have a predefined sequence of JCL or GCL commands executed automatically, before all user code, whenever an operator or IOF user logs on at a terminal (MAIN or otherwise), or when a job is started. Such startup sequences may be mandatory or optional, as defined by the following parameters of the project cataloging commands (see Chapter 8):

MSTUPI	mandatory (IOF)
MSTUPB	mandatory (BATCH)
OSTUPI	optional (IOF)
OSTUPB	optional (BATCH)

For the SYSADMIN project, special rules apply: any NSTARTUP (or STARTUP=0) parameters given in the logon sequence or $JOB statement or EJR command, inhibit mandatory and optional startup of this project whereas for other projects only the optional startup can be inhibited.

For startup sequences using GCL, refer also to the *IOF Terminal User's Reference Manual*.

Advantages

Startup sequences can be used to perform the following functions:

- Specification of a common environment at the installation level (site startup), the project level (project-name startup), or the user level (user-name startup). For example, the JCL statement VALUES or SET_VALUES in GCL can be used to specify parameter values and the JCL statement LIB or MWLIB in GCL can be used to specify libraries.
- Users of a given project can be installed immediately in a given processor (e.g., MAINTAIN_LIBRARY or IQS) without explicitly calling the processor. In such a case, the GCL or JCL necessary to call and execute the processor concerned is stored in the project-name or user-name startup.
- Certain special steps can be executed automatically (for accounting or initialization purposes, for example) at the installation level (site startup) and/or the project/user level (project-name or user-name startup).

Creation and Maintenance

The startup sequences are members of a source library named SITE.STARTUP.

The SITE.STARTUP library must be on a resident volume. The members of the library are created and maintained using the standard library maintenance facilities of MAINTAIN_LIBRARY, as described in the *Library Maintenance Reference Manual*. In installations which have access rights implemented, all users must have at least the READ access right to the SITE.STARTUP library, and the main operator project must have the READ access right. In such installations, it is desirable to restrict the WRITE access on this library to the SYSADMIN project.

Each startup sequence corresponds to a library member. There are five different types of startup sequence, one type for each of the following:

- the site
- the system
- Stations
- Projects
- Users

Testing

Startup sequences can be tested using the JCL statement $SWINPUT or the GCL command ALTER_INPUT. For example, to test the startup of a particular project, the following statement would be used:

```
S: AI project-name_I SITE.STARTUP;
```

This statement will execute the startup called project-name_I. It is also possible to execute the procedure from a work file and then move it to the SITE.STARTUP library when it is suitable for use.

10.2.1 Site Startups

There are two startup sequences for the site; they are held in the members SITE_B and SITE_I of the SITE.STARTUP library. These sequences may be inserted in all batch or IOF jobs (depending on MSTUPB/I), and if they are, they are the first to be executed.

Because SITE members may be executed for all users, they should be carefully designed to handle only common needs and procedures.

10.2.2 System Startups

The system startup sequence is held in the member SYSTEM of the SITE.STARTUP library, and is executed whenever the system restarts. The contents of switches 0, 1, 2, and 3 can be tested using GCL commands to determine which RESTART has been done:

SW0 ON : WARM RESTART
SW1 ON : COLD RESTART
SW2 ON : CLEAN RESTART
SW3 ON : DUMP taken.

10.2.3 Station Startups

There may be one station-name startup sequence for each of the other stations (not the main station). The sequences are named after the stations to which they apply. When the first operator with STATION operator status logs on at a station he or she becomes the STATION operator for that station, and the startup sequence for that station (located by station-name in the SITE.STARTUP library) is executed.

10.2.4 Project Startups

There may be a batch and/or an IOF project-name startup sequence for each of the projects on the site. The sequences are named after the project to which they apply, and are suffixed by _I or _B for interactive or batch sequences, respectively.

FOR EXAMPLE:

- if MSTUPI/B = PROJECT, the project startup will be executed. The project startup is mandatory in this case.
- if MSTUPI/B = SITE and OSTUPI/B = PROJECT, the site startup will be executed, then, if it exists, the project startup (unless NSTARTUP is specified at log-on).

❑

10.2.5 User Startups

Startup sequences can also be associated with particular users. In this case the SITE.STARTUP library member(s) will bear both the user-name and the project-name (i.e., project_user_I or project_user_B). This allows different sequences to be assigned to different users working under the same project.

Here too the sequences are controlled by the MAINTAIN_CATALOG parameters MSTUPI/B, OSTUPI/B, and NSTARTUP, as described above.

11. File and Volume Management Procedures

File and volume management from the System Administrator's standpoint is mainly a question of establishing rules and conventions for the site that minimize confusion, and maximize data security and data integrity.

Administrative responsibilities tend to fall into the following categories:

- implementing file, catalog, and volume naming conventions which encourage the use of unique names throughout the site
- protecting user files against hardware and software failures
- recovering user files
- protecting files against unauthorized access
- managing disk space quotas
- migrating files from VBO volumes to FBO volumes (or vice versa)

11.1 File Protection and Recovery Procedures

11.1.1 Installing the Before Journal

The SYS.JRNAL is a non-standard BFAS disk file created (and cataloged) at system configuration using the TAILOR command JBEFORE. Its default location is on the system volume, though any disk volume, resident or non-resident, FBO or VBO, can be specified. Full details are given in the *System Installation, Configuration, and Updating Guide*.

Extensions to the SYS.JRNAL are dynamically located on media specified by the JCL commands ASSIGN H_BJRNL and ASSIGN H_BJRNL1, if these have been included at job submission. Full details are given in the *File Recovery Facilities User's Guide*.

Size considerations

The default size for SYS.JRNAL is:

- 27 cylinders for an MS/B10 volume (VBO)
- 21 cylinders for an MS/D500 volume (VBO)
- 16,368 file blocks for an FBO volume

It is recommended that BJ be installed on an FSA disk. (Loss of space will then result on CKD-FBO.)

The default size is normally sufficient in a site where up to 90 jobs can journalize simultaneously, that is, in a site where both the MULTLEV and BJSIMU (see below) configuration parameters are set to their default values. If the number of journalizing jobs is likely to be higher than 90, the System Administrator should choose a proportionately higher size for the SYS.JRNAL file.

As file extensions are very time-consuming, it is generally better to over-estimate the initial size than to under-estimate it. If on the other hand memory, backing store and/or disk space resources are low, some optimization can be achieved through the BJSIMU parameter.

Configuration parameters MULTLEV and BJSIMU

MULTLEV places a limit on how many jobs can execute concurrently, and BJSIMU specifies how many of these are allowed to use SYS.JRNAL simultaneously. If BJSIMU is not specified, it defaults to a value slightly less than MULTLEV (see the *System Installation, Configuration and Updating Guide for details*).

There are considerable advantages though, to choosing a BJSIMU value which reflects the site's normal simultaneity level, particularly if this is consistently and significantly lower than MULTLEV. An appropriate value for BJSIMU can save:

- approximately 0.4 Kbytes of backing store for each job using the Journal
- up to 1.5 Kbytes of main memory while the Before Journal mechanism is active
- 1 track of disk space for each job using the SYS.JRNAL file

11.1.2 Modifying the Size of the Before Journal

If SYS.JRNAL is being extended too often, the System Administrator should allocate it a new size. The procedure is as follows:

- Wait until there are no steps journalizing, and then de-allocate SYS.JRNAL.
- Preallocate SYS.JRNAL with a new size.
- Shut down the system.
- Perform a RESTART COLD.

A useful indication of the required size can be found in the Job Occurrence Report (JOR) which, if the job terminates normally, gives the maximum number of tracks or blocks used for journalization.

11.1.3 Installing the After Journal

The After Journal is created using the MAINTAIN_JAS command CREATE. This command:

- Allocates disk space to the After Journal directory, and catalogs it with the name SYS.JADIR. This is a linked queued file with a default size of 12,000 Kbytes and a file extension size of 500 Kbytes. Its default location is on a resident disk, though any disk, resident or non-resident, FBO or VBO, can be specified.
- Allocates and catalogs the primary journal files. These are named "SYS.JA.J<n><vol>", where <n> is a number between 1 and 8 differentiating between journals on the same volume (always 1 for a tape volume), and <vol> is the name of the volume on which the journal is allocated.
- Designates a media list and device class for the primary and secondary journals. Primary journals must be disk or tape files (though disk is recommended), and secondary journals must be tape or cartridge files.
- Defines whether the journals are to be recycled automatically (default), or on request by the System Administrator.

Generally speaking, the System Administrator should select the default values when installing the After Journal. These can afterwards be modified using the MODIFY_PARAMETERS command if necessary.

Full details are given in the *File Recovery Facilities User's Guide*.

11.1.4 Choosing the After Journal Blocksize

An optimal choice of blocksize for the journal files can significantly improve the use of disk space. The default blocksize is 8192 bytes (of which 8162 are usable for data), but the System Administrator can modify this using the MODIFY_PARAMETERS command.

An optimum blocksize is one which achieves a fill rate of between 50% and 80%. Generally, it is better to begin with a large blocksize and gradually reduce it to obtain the best fill rate. The LIST command of the MAINTAIN_JAS utility provides statistics on the filling of blocks.

If the journals are on FBO disks, the maximum blocksize is 32,000 bytes. If the journals are on VBO disks, the blocksize should be less than half the track size.

11.1.5 Maintaining the After Journal

From time to time the System Administrator may have to call the MAINTAIN_JAS utility, and:

- Add or remove primary and/or secondary media from the After Journal (MODIFY_MEDIA command).
- Optimize the blocksize and the maximum file size of the journals (see above).
- Recycle journals from "active" to "available", to liberate space occupied by obsolete After Images (RECYCLE_JOURNAL_FILE command).
- Transfer primary journals to tape or cartridge (TRANSFER_PRIMARY command).
- Delete a user file from the SYS.JADIR directory (FORGET_USER_FILE command).
- Delete a TDS journal from the SYS.JADIR directory (FORGET_USER_JOURNAL command).

Full details are given in the *File Recovery Facilities User's Guide*.

11.1.6 Dumping the After Journal (Using DUMPJRNL)

TDS users are able to journalize their own private data in the After Journal while TDS is running. The DUMPJRNL utility is used to extract this data from the After Journal and append it to a sequential output file, called the "User Journal".

The User Journal can be a disk or tape file, and must have a variable record size at least as large as the largest record to be dumped. A fixed record length may lead to a meaningless padding of the dumped records. An insufficient record length leads to a DUMPJRNAL abort with a LINERR return code.

NOTE:
If the After Journal uses automatic recycling, there is no guarantee that the same TDS records can be dumped more than once.

Full details are given in the *File Recovery Facilities User's Guide*.

11.1.7 Recovering User Files (Using ROLLFWD)

Before recovering user files using the ROLLFWD utility, you must first:

1. Restore the user file(s) up to their last save date.

   For tape files, you can use the RESTORE_FILE or RESTORE_DISK commands. For disk files, use preferably the RESTORE_FILE, RESTORE_FILESET, or RESTORE_DISK utilities. The LOAD_FILE or LOAD_FILESET commands cannot be used to provide restored versions for ROLLFWD.

   For a multivolume file, you must provide a complete restored version resulting from a coherent set of extents.

   Full details on file restoration are given in the *Data Management Utilities User's Guide*.

2. For UFAS and IDS/II files, verify that the maximum control interval (MCI) is <= 4 Kbytes. If it is, you need not specify SIZE or POOLSIZE values, as the default values will suffice. If, however, the MCI is greater than 4 Kbytes, you must increase SIZE and POOLSIZE as follows:

   ```
   SIZE = 120 + (6 * (MCI-4)) Kbytes
   POOLSIZE = 60 + (6 * (MCI-4)) Kbytes
   ```

   In order to optimize UFAS and IDS/II file management, SIZE and POOLSIZE should if possible be increased by the same value.

11.1.8 Recovering the After Journal (JRU) and Recovering the SYS.JADIR Using REBUILD

The SYS.JADIR directory is considered damaged if it does not contain correct information concerning aborted FRUs, or if it is not available.

If SYS.JADIR contains incorrect information (message JP44), the Journal Recovery Utility (JRU) may recover the directory if journalization is on disk. Full details are given in the *GCOS 7 Guide to Security*

If SYS.JADIR is not available (message JP41 variant 1 or JP43), the System Administrator must:

- stop all journalizing steps
- save all the journalized files
- create a new After Journal

If user files are damaged at the same time as SYS.JADIR, the System Administrator must:

- restore SYS.JADIR from its last save date
- restore the user files from their last save dates
- execute ROLLFWD to recover the files up to SYS.JADIR's save date

NOTE:
This recovery procedure is possible only if no recycling has been performed (automatically or manually) since the last SYS.JADIR save date.

11.1.9 Recovering UFAS Files (Using the Salvager)

An additional aid to file recovery is the UFAS File Salvager. This is called automatically when a UFAS file is reopened in I/O mode, and produces JOR messages prefixed with the code DUF. These messages are described in the *Messages and Return Codes Directory*.

For UFAS indexed sequential files, only primary indexes are salvaged. If any secondary indexes have to be salvaged, the return code SECIDXNAV is returned when the file is reopened. The user must run the SORTIDX utility for salvaging secondary indexes.

Salvaging of libraries is performed automatically the first time the library is opened following a system crash.

11.2 File Security Procedures

11.2.1 Setting Access Rights on the System

The System Administrator, via the project SYSADMIN, is responsible for setting up all the access rights on the system.

First, he must initialize the access rights mechanism by declaring the SYSADMIN project OWNER of the Site Catalog.

Second, he must grant all projects access rights on the SITE directory; starting with WRITE access on the SITE.CATALOG file, and then at least READ access on all other SITE files. (Remember that WRITE access includes READ access, just as OWNER access includes both WRITE and READ access.)

Third, he can, if necessary, set access rights on the SYS directory.

Fourth, if other master directories have been created, he should grant individual projects OWNER access on these directories in the SITE.CATALOG.

Fifth, he should grant projects OWNER access on private catalogs root.

Sixth, he should grant projects OWNER access on individual volumes.

11.2.1.1 Initializing Access Rights on the System

Before the access rights can be applied on the system, they must first be set up. The requirements for this are:

1. The access rights facility must be available on the site.
2. The user, project, (and optionally billing) information checks in the Site Catalog must be validated by the command VAL (see Chapter 9).

If these requirements have been met, the System Administrator can initialize the access rights mechanism with the following command:

```
MODIFY_ACL NAME=*      PROJECT=SYSADMIN/OWNER
           TYPE=DIR    CATNAME=SITE;
```

This makes the SYSADMIN project the owner of the root of the Site Catalog. Now only SYSADMIN users can create master directories and other objects (though preferably not files) directly under the root of the Site Catalog.

NOTE:
It is strongly recommended that CATNAME be specified in all uses of the MDACL command , when no catalog has been explicitly attached.

11.2.1.2 Setting Access Rights on the Site Catalog

After the access rights mechanism has been initialized (see above), the System Administrator can set access rights on the master directories and files in the Site Catalog. The procedure is as follows:

1. Set OWNER access on the SITE directory

SYSADMIN project must be OWNER of the SITE directory and all of the files under it (Such as SITE.STARTUP and SITE.CATALOG). The command is as follows:

```
MDACL SITE TYPE=DIR PROJECT=SYSADMIN/OWNER
      CATNAME=SITE
```

2. Set WRITE access on the SITE.CATALOG file

All projects must be given WRITE access to the file SITE.CATALOG. This allows project users to catalog objects under the Site Catalog. The command is as follows:

```
MDACL NAME=SITE.CATALOG PROJECT=*/WRITE TYPE=FILE CATNAME=SITE;
```

3. Set READ/WRITE access on other SITE files

All projects should have READ access to all files in the SITE directory. Selected projects should also have WRITE access to certain files, such as SITE.STARTUP. The following command is an example of what you can do:

```
MDACL NAME=SITE.STARTUP
      PROJECT=(*/READ PROJ1/WRITE PROJ2/WRITE)
      TYPE=FILE
      CATNAME=SITE;
```

11.2.1.3 Set Access Rights on the SYS Files (If Necessary)

It is recommended that only the default set of access rights be set on the system files. This is done using the GIUF facility:

```
PROTECT_GCOS ACCESS_RIGHT=SET
```

This facility is fully described in the *System Installation, Configuration and Updating Guide*.

11.2.1.4 Setting OWNER Access on Master Directories

After setting READ/WRITE access rights on the SITE directory, the System Administrator can now grant OWNER access on the other master directories in the Site Catalog.

If the directory does not already exist, it must first be created using the command:

```
CREATE_DIR  NAME=master-directory;
```

Once the directory exists, the OWNER access rights can be set, using:

```
MDACL  NAME=master-directory
       TYPE=DIR
       PROJECT=project-name/OWNER
       CATNAME=SITE;
```

Note that OWNER access on a master directory can also be granted at project creation time, using the command:

```
CRP project-name,...., MASTDIR;
```

This automatically makes the specified project the owner of the master directory which is created with the same name as the project.

If the project already exists, and the name of the master directory is identical to the project name (as is recommended), then use:

```
MDP project-name,...., MASTDIR;
```

The CRP and MDP commands are fully described in Chapter 9.

11.2.1.5 Setting Access Rights on a Private Catalog

Ensure first that access rights have already been set on the master directory in the SITE.CATALOG (as explained above).

A private catalog must be attached before the OWNER access right can be set on it. If the catalog is auto-attachable (as is recommended) then there is no need to attach it before setting the access right. If it is not auto-attachable, and is not currently attached, then it must first be attached as follows:

```
ATTACH_CATALOG CAT=catalog-name;
```

Once the catalog is attached, the OWNER access right can be set, as follows:

```
MDACL  NAME=*
       PROJECT=project-name/OWNER
       TYPE=DIR
       [CATNAME=catalog-name];
```

The parameter CATNAME is mandatory if the catalog concerned is auto-attachable.

Note that users of projects which have no private catalogs can create them using the command CREATE_CATALOG. In this case, the OWNER access right is automatically set.

11.2.1.6 Setting Access Rights on a Volume

The OWNER access right on a volume is set by the System Administrator using the commands:

```
PREPARE_DISK/PREPARE_TAPE   for volumes containing no files
MODIFY_DISK                  for volumes containing files
```

These set or modify the volume security flag (CTLACC parameter) and the volume owner information (OWNER parameter).

Full details are given the *GCOS 7 Guide to Security*.

11.2.2 Returning to an Unprotected System

In some cases, the System Administrator may wish to cancel the access rights that were previously set on the system, either to return to the situation of an unprotected system, or to set new access rights from scratch. The procedure for cancelling the access rights on a system is as follows:

1. Cancel the OWNER access right on each private catalog

If a private catalog is not auto-attachable, and is not currently attached, it must first be attached before the OWNER access right is cancelled. The command to attach a catalog is:

```
ATTACH_CATALOG  CAT=catalog-name;
```

Once a private catalog is attached, the OWNER access right is cancelled with the command:

```
MDACL  NAME=*
       DELETE=owner-project
       TYPE=DIR
       [CATNAME=private-catalog];
```

The parameter CATNAME is mandatory if the catalog concerned is auto-attachable.

2. Cancel the OWNER access right on each protected volume

The command to do this is:

```
MODIFY_DISK  VOL=vol-name:device-class  CTLACC=NO;
```

Then delete the list of volumes that are assigned to projects, using the MAINTAIN_CATALOG command MDP. Enter this command with MSVOL='; or MTVOL=';.

3. Cancel the access rights on the system files

If the default access rights were set on the system files using the GIUF function PROTECT_GCOS, these can be cancelled using:

```
PROTECT_GCOS  ACCESS_RIGHT=RESET;
```

The SYSADMIN project remains the OWNER of these files.

4. Cancel the access rights on SYS.CATALOG

If SYS.CATALOG is protected, the access rights on this catalog must be deleted before the access rights are deleted on SITE.CATALOG. The command to do this is:

```
MDACL  NAME=*       DELETE=SYSADMIN
       TYPE=DIR     CATNAME=SYS;
```

5. Cancel the OWNER access right on SITE.CATALOG

```
MDACL  NAME=*
       DELETE=SYSADMIN
       TYPE=DIR
       CATNAME=SITE;
```

If at system initialization the Site Catalog is unprotected, and the System Catalog is still protected, the following error message will be issued:

```
CG09. SYS.CATALOG IS CONSIDERED AS NOT VALID
```

If this happens, the System Administrator must:

- Reset the access rights on the Site Catalog.
- Delete the access rights on the System Catalog.
- Reset the access rights on the Site Catalog again.

Similar operations must be performed if the Site Catalog is unprotected and a private catalog is still protected.

11.2.3 Changing the Owner of a Private Catalog

Suppose that the owner of a private catalog is to be changed from the project P1 to the project P2. First grant P2 OWNER access right on the master directory in the Site Catalog, then grant it OWNER access right on the private catalog.

These modifications must be performed by the System Administrator via the project SYSADMIN,

1. Change access right on the Site Catalog

```
MDACL  NAME=master-directory
       PROJECT=P2/OWNER
       TYPE=DIR
       CATNAME=SITE;
```

2. Change access right on the private catalog

If the private catalog is not auto-attachable, and is not currently attached, it must first be attached before the OWNER access right is changed. The command to attach a catalog is:

```
ATTACH_CATALOG CAT=catalog-name;
```

Once the catalog is attached, the OWNER access right on it can be changed as follows:

```
MDACL  NAME=*
       PROJECT=P2/OWNER
       TYPE=DIR
       [CATNAME=catalog-name];
```

The parameter CATNAME is mandatory if the catalog concerned is auto-attachable.

Note that changing OWNER is not equivalent to deleting the OWNER access right and then resetting it, because in this latter case, all the Access Control Lists are deleted.

11.2.4 Deleting a Project which is the Owner of a Private Catalog

Suppose that a project, OWNER of a Private Catalog, is to be deleted. First, delete its access rights on its master directory in the Site Catalog; then delete its access right on its private catalog.

1. Delete access right on the Site Catalog

```
MDACL  NAME=master-directory
       TYPE=DIR
       DELETE=project-name
       CATNAME=SITE;
```

2. Delete access right on the private catalog

If the private catalog is not auto-attachable, and is not currently attached, it must first be attached before the OWNER access right is deleted. The command to attach a catalog is:

```
ATTACH_CATALOG CAT=catalog-name;
```

Once the catalog is attached, the OWNER access right on it can be deleted as follows:

```
MDACL  NAME=*
       TYPE=DIR
       DELETE=project-name
       [CATNAME=catalog-name];
```

The parameter CATNAME is mandatory if the catalog concerned is auto-attachable.

11.3 VBO<->FBO File Migration

The VB0<->FBO file migration facility, is an administrative tool for moving user files and applications from one set of fixed disk volumes to another, in particular from VBO volumes to FBO volumes.

The files concerned may be mono or multi-volume, cataloged or uncataloged. System files and temporary files cannot migrate.

Files can be passed from source volumes to target volumes via a sequential tape or cartridge; a useful feature when migrating data to a distant site, for example.

If the Quota mechanism is active on the site, it must be de-activated if the source or target volume(s) are quota-controlled volumes.

There are four phases to the migration procedure:

- the preparation phase for defining the migration context
- the analysis phase for examining the impact of the current migration context
- the implementation phase in which the necessary JCL is generated to execute the migration
- the updating phase for tidying up the JCL and GCL user programs (and possibly catalogs) containing references to the migrated files.

A full description of this procedure and the associated MAINTAIN_MIGRATION commands is given in the *File Migration Tool User's Guide*.

11.4 The Storage Manager

The Storage Manager is a set of disk space administration facilities available on GCOS 7 systems. It can consist of:

- the ***BASIC VOLSET FACILITY*** for systems operating with or without access rights.
- the ***FULL VOLSET FACILITY*** for systems operating with access rights.
- the ***QUOTA FACILITY*** for systems operating with access rights.

The VOLSET and QUOTA facilities operate independently of each other. To know which of these (if any) are installed on your system, execute the IOF directive:

```
DISPLAY_STORAGE_FACILITIES (DSF)
```

From the System Administrator's standpoint, the VOLSET and QUOTA facilities are very useful tools for controlling, distributing, and optimizing the use of the system's disk space. The QUOTA facility controls the amount of disk space that can be allocated, and the VOLSET facility controls where and how this space can be allocated.

Figure 1-1 below gives a general picture of the Storage Manager and its user and system interfaces.

Compatibility

A SITE.QUOTA file allocated on a pre-TS7254 release does not support the Storage Manager. However this file can be migrated to the new release (and its format changed) without loss to your Quota configuration. The name "SITE.QUOTA" is retained for compatibility.

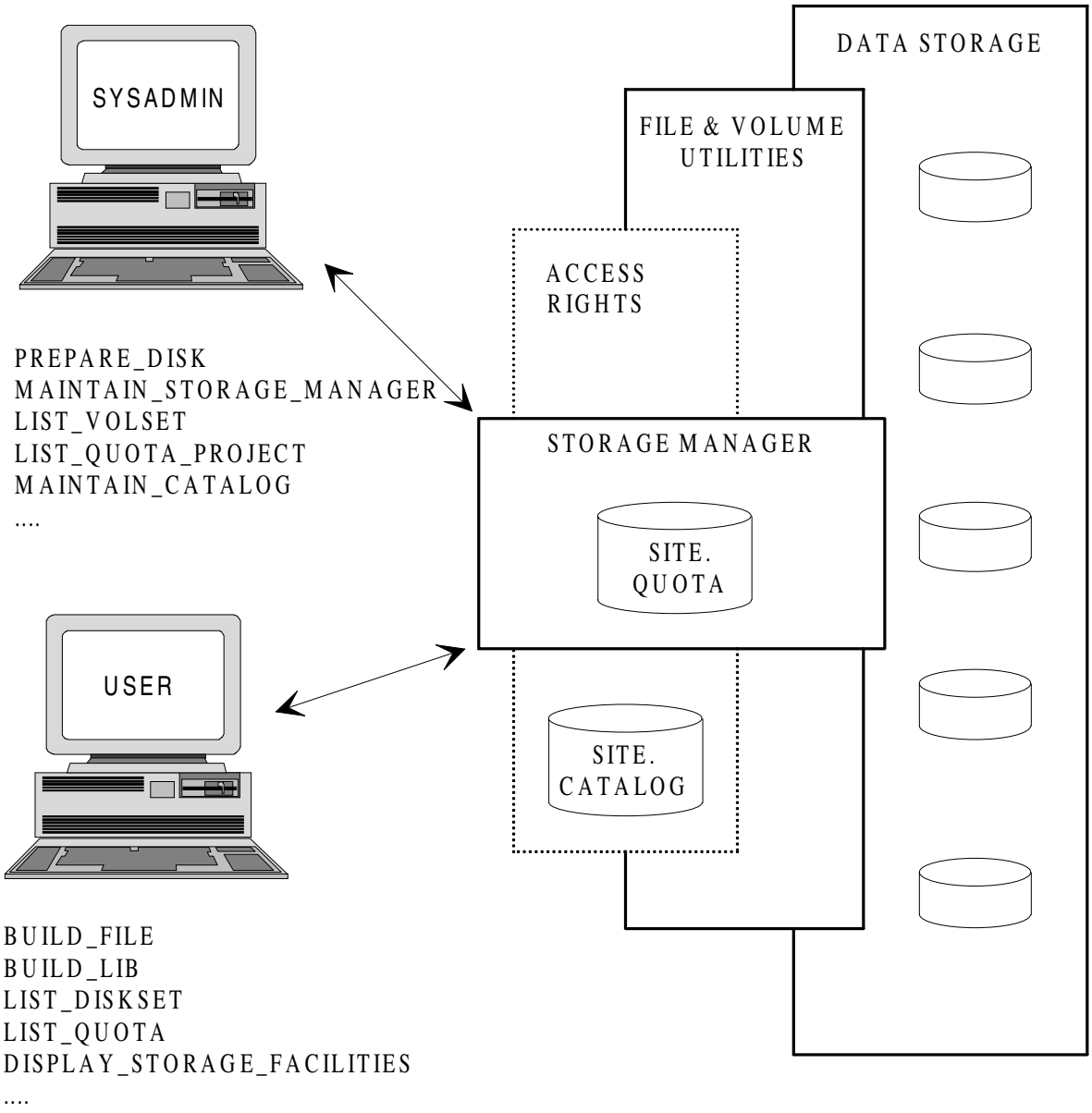

Figure 11-1. The Storage Manager

11.4.1 The VOLSET Facility

The VOLSET facility provides GCOS 7 users a simple and transparent method for allocating and extending cataloged files and temporary files.

The traditional method requires users to know and to specify the volume's media name and device class. They have to check the volume's access rights and that it has sufficient free space, and then select another if it does not.

CAUTION:

With the VOLSET facility, users can request a "default allocation", that is, they can omit all location syntax when allocating a file. The file is automatically allocated on the Site Volset or, if it exists, the project's default Volset.

A "Volset" is a logical and mono-organized set of disk volumes, created by the System Administrator and registered under a unique name in the Storage Administration file (SITE.QUOTA).

When a file is created or extended on a Volset, the VOLSET facility decides which volumes are appropriate and then selects from these the volumes with the most free space. Other factors, such as quota consumption and volume protection level, are also taken into account.

The VOLSET facility has two levels of operation, the basic facility and the full facility. They are both unbundled. The full facility is available only when access rights are set on the system.

To know which level (if any) is available on your site, execute the following IOF directive:

```
DISPLAY_STORAGE_FACILITIES (DSF)
```

Basic VOLSET facility

With the basic facility, only one Volset is allowed. This can be used by all projects and is called the "Site Volset".

All cataloged and temporary files are allocated on the Site Volset unless a media and device class or resident volume is requested.

Given that the SiteVolset is mono-organized, then only FBO files or only VBO files, but not both, can be allocated through the VOLSET facility.

Full VOLSET facility

With the full VOLSET facility, several Volsets are allowed. One of these is the Site Volset, and the others are called "Project Volsets".

A Project Volset is any Volset linked by the System Administrator to a specific project. Projects not linked to this Volset cannot use it. As Figure 11-2 shows, a Volset can be linked to more than one project and a project can be linked to more than one Volset. Typically a project will have a default Volset and one or more alternative, non-default Volsets.

All cataloged and temporary files are allocated on a project's default Volset unless a non-default Volset, a media and device class, or a resident volume is requested.

If a project does not have a default Volset, then its cataloged and temporary files are allocated on the Site Volset (unless a non-default Volset, a media and device class or a resident volume is requested).

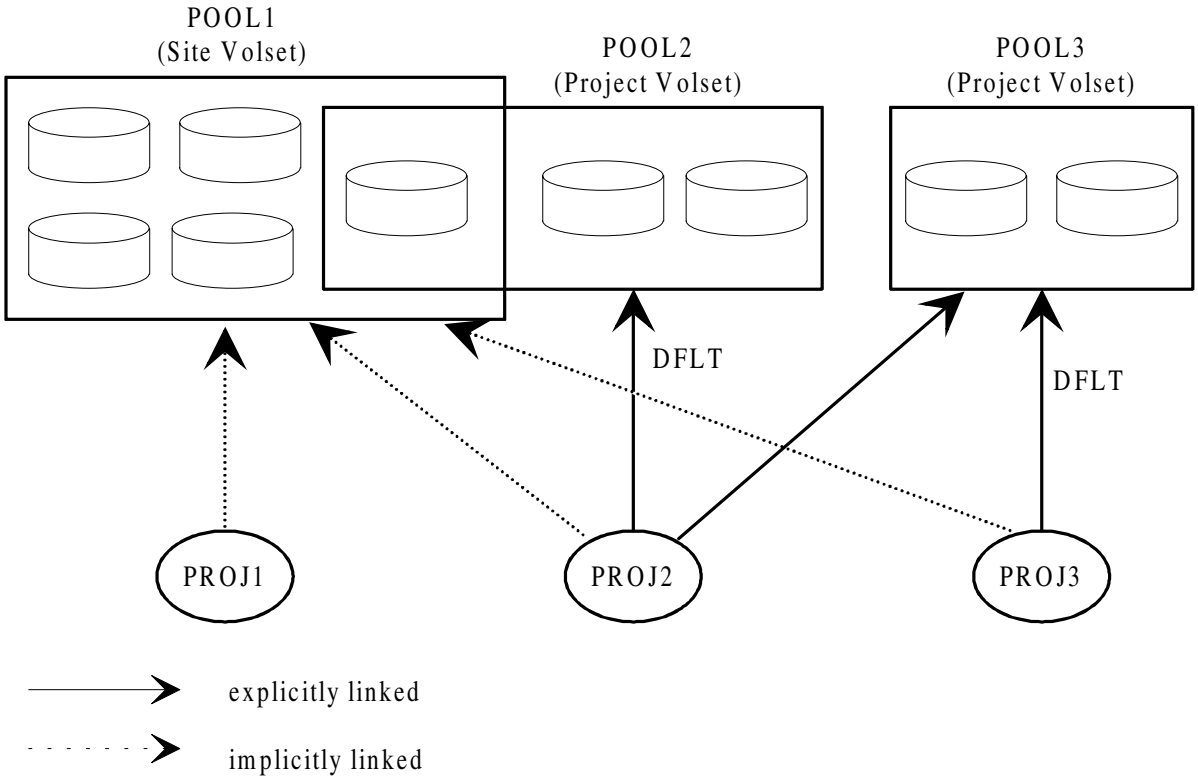

Figure 11-2. Example Volset Scenario (Full Facility)

In the above example:

All projects can use the Site Volset, POOL1. They do not have to be explicitly linked (and in fact they cannot be explicitly linked) to the SiteVolset.

PROJ1 can use only the Site Volset. It has not been linked to any Volset (and is consequently not registered in the SITE.QUOTA file).

PROJ2 uses POOL2 as its default Volset and POOL3 as an alternative, non-default Volset.

PROJ3 uses POOL3 as its default Volset. It has not been linked to POOL2 and so cannot use it.

11.4.2 Quota Facility

The QUOTA facility is an administrative and accounting tool for organizing, limiting, and monitoring the use of the system's disk space.

It controls the amount of cataloged and temporary space that can be allocated by each project.

For large DPS 7000 installations in particular, the QUOTA facility brings a number of substantial benefits, including:

- better project budgeting and accountability,
- better control over the allocation and distribution of user files,
- and, indirectly, better optimization of disk I/O workloads.

As an aid to project accountability, the QUOTA facility can write space consumption statistics to a private file where they can be used by a customer application for project budgeting purposes.

Access rights

The QUOTA facility cannot function unless access rights are present on the system.

For it to function effectively, all permanent files must be correctly cataloged, that is to say, cataloged under a master directory belonging to an owner project. This ensures that all permanent files have an identifiable and therefore accountable owner under all circumstances.

Full details are given in the publication, *Administering the Storage Manager*.

11.5 ASM 7 Solution

This is a package of file management products that cooperate with each other to provide a comprehensive and automated storage management solution for GCOS 7. It includes:

- the ASM 7 File Migrator (ARS)
- the ASM 7 Archive Manager (FDS7, EpochBackup7, or proprietary application)
- the ASM 7 Disk Space Manager (VOLSET facility)

For an overview of ASM 7 concepts and components, see the publication *Introduction to ASM 7.*

For full details of each ASM 7 component, refer to:

- *ASM 7 File Migration Administrator's Guide*
- *Administering the Storage Manager (for the VOLSET facility)*
- *EpochBackup7 Installation and User's Guide*
- *FDS7 partner documentation*

11.6 Mirror Disks Management

A mirror disk consists of a pair of FBO-formatted disks containing identical data. A single write request on a mirror disk generates two asynchronous writes, one to each of its constituent disks. If either the primary or secondary write fails, the affected disk is invalidated and any further writes are directed to the surviving disk. Thus, despite the I/O incident, data continues to be available and jobs can continue processing.

Mirror disks therefore provide a much greater guarantee against I/O-related interruption than do non-mirror disks. For this reason, they should be used for critical files such as:

- The JAS directory
- The After Journals (TDS User Journal included)
- The Before Journal (but not on a CKD-FBO disk)
- TDS Swap files (including the .RECOV file)
- TDS system files (including the CTLM and CTLN files)
- Database files
- Critical business application files
- Private catalogs
- Application logfiles
- CMSC files for TDS-HA systems

It is the System Administrator's responsibility to create the mirror disks (START_MIRROR) and to control their use by setting project access rights and quota rights.

In the case of mirror disk failure, the System Administrator will need to recreate the mirror disk (REFILL_MIRROR).

Full details are given in the *Mirror Disks User's Guide*.

12. Job Management Procedures

This chapter describes the following procedures and facilities:

- reducing job input overheads
- job management and data security
- job checkpoint/restart mechanism
- job accounting procedures
- the Main console log file (SYS.LOGC)

12.1 Reducing Job Input Overheads

A job is introduced into the system by the command ENTER_JOB_REQ (EJR). This specifies the name and location of the job source file, plus details such as job class, priority, associated project, and so on.

When an EJR command is entered, two things happen:

- First, the consistency of the EJR details is checked. If these are correct the reading of the source file begins. This is the function of the READER service job.
- Next, the JCL is translated. This function is performed by the JTRA service job.

Both the READER and the JTRA service jobs are designed to continue working as long as requests exist in their work queues, but otherwise they must be initiated and terminated like any other jobs or job steps. They therefore entail overhead costs for the system.

There are three ways of reducing this overhead:

- by controlling the number of READER jobs (see below)
- by using the NTERM option of the INPUTCL statement (see below)
- by batching the job inputs whenever possible.

12.1.1 One or More READER Jobs ?

Normally, there should be at least two READER jobs started. This is the default number specified by the MAXTFSIT parameter of the DJPCTL statement (CONFIG utility).

If MAXTFSIT is reduced to 1, all EJR commands are handled successively by a single READER. This causes delays each time a job source file is on an unmounted volume. The READER is stopped when this happens, and no other jobs can be read in the meantime. With 2 READERS, however, the second READER can continue working.

Too many READERS, on the other hand, leads to jobs being executed in a very different order to that in which they are introduced. A small job, for example, may start execution before an important job, even if this latter job has an execution priority of 1.

12.1.2 The NTERM Option of INPUTCTL

The NTERM option of INPUTCTL can be used to prevent the READER job from terminating and the JTRA job from going into its idle state, as they otherwise would do when no pending requests remain to be processed. Rather, with this option, they are maintained in a waiting state as active processes.

When a READER job is held in the Wait Process state as a result of NTERM, it releases its working set. On most sites, this option will be useful if READER jobs are started every 2 minutes or less.

When JTRA is held in the Wait Process state as a result of NTERM, it does not release its working set. This option will be useful where JTRA steps are activated every 2 minutes or less. Leaving JTRA idle costs approximately 48 Kbytes.

When NTERM is used with either READER or JTRA, the service job to which it applies is started at the first work request after System Restart, and remains active until the TSYS command, which terminates it automatically. The START_LOAD command with the TRACE option should be used to observe the rate of READER and JTRA job steps (steps H_RUN and H_EXECUTE, as described in the *System Operator's Guide*).

The NTERM affect can also be achieved dynamically with the command SIR PERM, and cancelled using the command TIR PERM.

12.2 Job Management and Data Security

12.2.1 The System is Unprotected

In an unprotected system any user can submit jobs for the account of any other user.

There is no check on the jobs introduced, *unless* the Site Catalog is validated (VAL command of MAINTAIN_CATALOG) in which case the user/project information of the EJR command is checked against that in the Site Catalog. If the BILLCHECK option is used then user/project/billing information is checked.

In order to submit jobs from an RBF station the project must be an authorized project for that station.

For further details, refer to Chapter 8.

12.2.2 The System is Protected

In a protected system (one in which access rights have been applied) the Site Catalog must be validated. The same checks are made on user/project as for an unprotected system and the following additional controls exist:

- Normal users (i.e., whose projects have not got the attributes MAIN/STATION) can submit jobs only in their own names; it is not possible for them to submit jobs for the account of another user.
- The MAIN and STATION operators can submit jobs for the account of other users.

The reason for having these controls is to make sure that a user does not claim the identity of another user in order to access files which he would not be able to access when using his own identity.

When access rights are implemented on a site it is recommended that a policy be established for the site as regards the stored JCL or GCL. This may involve grouping stored JCL or GCL into several libraries. Some possibilities are:

- No $JOB/$ENDJOB in the stored JCL or GCL. This is the general case. A user of a project with at least the EXECUTE right on files containing the JCL or GCL can use ENTER_JOB_REQUEST on any stored JCL or GCL which has no $JOB/$ENDJOB; the system automatically creates the $JOB/$ENDJOB with the identity (user/project/billing) of the submitter of the job. An advantage is that the stored JCL can also be used by INVOKE, EXECUTE, and $SWINPUT or the stored GCL by EXECUTE_GCL and $SWINPUT.

- $JOB/$ENDJOB present on the stored JCL, but no specification of the user or project. This fixes parameters by default for the running of the job; making it unnecessary to give the parameter when the job is run. A user of a project with at least the EXECUTE right can use ENTER_JOB_REQUEST on any such stored JCL. This JCL cannot be used by INVOKE, EXECUTE or $SWINPUT.
- $JOB present, specifying the project but not the user. The system will take the name of the submitter as the user-name. However, the submitter must be a member of this project to be able to submit this job. In this case the stored JCL could be grouped together by project. (Care must be taken if the operator is to run such jobs using ENTER_JOB_REQUEST as it is unlikely that the operator will be a member of all the projects).
- $JOB present, specifying the user and the project. In this case the user named will be able to run the job, and the MAIN or STATION operator will be able to run the job if he has at least the EXECUTE right (assuming that the user is the submitter, or that the submitter's project has the MAIN attribute).

12.3 JOB Checkpoint/Restart Mechanism

The function of the checkpoint/restart mechanism is to save an image of a job step and to update the current recovery point according to this saved image.The point of activation for checkpoint, within the user program, is specified by the user.

During the restart phase the operator is asked if he wants to restart the step. If restart is requested, a new step initiation phase is entered to re-execute the step from the last recovery point.

The following conditions must be satisfied for an aborted step to be restarted:

- The job is known to the system.
- The step completion severity code is at least 3, and the abort did not result from a CANCEL_JOB STRONG operator command.
- All the resources allocated to the step are still allocated. (They can be partially released if the operator puts the job in the suspended state by entering a HOLD_JOB command before answering the question REPEAT?)
- The STEP statement of the aborted step specifies the REPEAT option.

The consequences of restart processing are:

- The files held by the restarted step are closed and rolled back if necessary.
- A restarted step will keep the same dynamic step number until it has terminated fully.
- The step completion code and the job switches will be temporarily restored to the values they had during the first step initiation, thereby allowing the JCL to be reprocessed.

REPEAT in the $JOB statement will be taken into account only if REPEAT is not specified in the STEP statement; that is, REPEAT at step level overrides REPEAT at $JOB level.

12.3.1 Programming for Checkpoint/Restart

The system provides two routines through which the programmer may call the checkpoint facility. Both routines have identical parameters and return codes.

H_CK_UCHKPT	causes a snapshot to be taken and provides, via parameters, the result of the action.
H_CK_UMODE	reports on the current checkpoint status of the step. It does not cause a snapshot to be taken.

12.3.2 Checkpoint Errors

Certain errors within checkpoints are reported as warnings to the operator and the JOR. The job step accessing the checkpoint will be informed of the error through the CKINF parameter of H_CK_UCHKPT or H_CK_UMODE.

12.3.3 Checkpoint at System Shutdown

When the operator issues a TERMINATE_SYSTEM command for system shutdown, any job steps using the checkpoint mechanism are automatically suspended when the next successful checkpoint is made. These job steps will then be automatically restarted if the next system session begins with a Warm Restart.

12.3.4 Checkpoint/Restart Limitations

Checkpoint/Restart does not support the following:

- deferred restart
- relocation of files before the restart
- immediate SYSOUT deliveries (they are deferred until the end of the step)
- several coexisting recovery points for the same step (multiple checkpoints)
- multi-task step recovery.

12.4 Job Accounting

User access to the system accounting functions described here is the responsibility of the System Administrator, who should establish appropriate control procedures for his site.

The system accounting facilities record all relevant accounting information for each job and each job step. The information is gathered by the system within a sequential file handled by the standard editing utility, or a program written by the user to meet his own requirements.

Job accounting can be either active or inactive depending on a system configuration parameter. Different types of record can be stored in the accounting file; these must also be specified at system configuration time.

12.4.1 The System Accounting Files, ACT1 and ACT2

All accounting information is first gathered in two system files, SYS.ACT1 and SYS.ACT2, located in backing store and used alternately. These Virtual Memory Files (VMFs) are defined as system permanent files, with only one of the two files active at any point in time. The files are erased when a clean restart is performed.

Size of SYS.ACT1 and SYS.ACT2

The size of the files SYS.ACT1 and SYS.ACT2 can be adjusted at CONFIG time using the parameter ACTSIZE which specifies the number of blocks for each accounting file. The default size for each file is 200 blocks. Refer to the individual record descriptions in Appendix D for the record sizes.

If the size is adjusted, the contents of the file(s) are lost even if there is no clean restart. Thus a clean restart is not strictly necessary, but it may be useful because these files may be too big or too small depending on the number of records that are selected through the ACCOUNT parameter. If it is too big, valuable space will be lost in backing store. If it is too small the DUMPACT utility (see below) will need to be run too frequently due to the files' filling up too rapidly.

12.4.2 Specifying the Accounting Options (Using CONFIG)

Accounting information is collected at each of the different stages in the life of a job, and at critical states of the system; i.e., shutdown, crash, and restart. This information consists of various types of accounting records, each one being identified by a record type. These records are stored sequentially within the user accounting file by the DUMPACT utility program. See Appendix D of this manual for the format of each accounting record.

The choice of which accounting records are to be stored is specified by the CONFIG statement ACCOUNT. See the *System Installation and Configuration Guide*.

12.4.3 Dumping the Accounts (Using DUMPACT)

The DUMPACT utility is used to dump accounting information from whichever system accounting file is full, or from the currently active file if neither file is full.

NOTE:
The H_DUMPACT step may only be started by a project which has the DUMPACT application assigned. If this is not the case then use the MDP command with ADDAPPL=DUMPACT, to modify the Site Catalog (see Chapter 8).

When the active file becomes full, the second file becomes active, and the operator is informed by the following message:

```
AC01*GCOS:ACCOUNTING FULL.RUN DUMPACT.
```

The message given above will be repeated until DUMPACT is started in order to transfer the full file to the user accounting file. If DUMPACT is run when there is no full file, the active file is automatically closed and dumped to the user accounting file.

If both files become full, the older (inactive) file is erased and set to the active state.

You can specify at configuration time that the command EJR DUMPACT be submitted automatically to the system each time the active file becomes full. This command starts the DUMPACT utility. If this option is used, the operator is informed that DUMPACT has been started by the message:

```
AC02*GCOS: ACCOUNTING FULL, DUMPACT COMMAND SUBMITTED
```

The operator should verify that the DUMPACT utility has terminated normally.

The DUMPACT utility provides various facilities. It enables the user accounting file to be opened in OUTPUT or in an APPEND mode; this file could also be used as a dummy file. Furthermore it ensures the compatibility of record formats between releases.

A debugging option is provided to enable the user to debug billing programs when adapting them to a new release format, without destroying the previous format necessary for current processing. The principle of this option is to maintain the system input file as it is, available for another run of the DUMPACT utility.

A message to the operator signifies whether the DUMPACT operation has been successful or unsuccessful. If it has been unsuccessful, details are given in the JOR.

For further information, see the *System Operator's Guide*.

12.4.4 Editing the Accounts (Using EDITACT)

The EDITACT utility is used to print in clear form the information stored in a user accounting file. See the System Operator's Guide for further details.

12.4.5 The User Accounting File

The user accounting file is a UFAS Sequential tape, cartridge or disk file with the following characteristics:

```
CISIZE  = 2048
RECSIZE = 1400
RECFORM = V
```

These characteristics are specified at file allocation time using the GCL command BUILD_FILE or the JCL statement PREALLOC.

The size of this file must be chosen on the basis of how many accounting records are to be contained. This figure is dependent on how many jobs and job steps are run each day and how many days the same file is to be used to accommodate the output from DUMPACT.

12.5 The Main Console Log (SYS.LOGC)

The SYS.LOGC file is a circular file intended to record the dialog with the main console(s). The size of the file is two cylinders (one on a MS/D500), which is large enough to hold the main console dialog for a 24-hour session. The file can be printed once a day or searched for a specific item.

12.5.1 Writing to SYS.LOGC

Each input and output line at the main operator's console is automatically logged in the SYS.LOGC file, if it exists.

The command TERMINATE_CONSOLE_LOG (TCLOG) stops the logging process. This should not be done unless there is an important reason to do so (for example if there is an I/O problem with the file).

The command START_CONSOLE_LOG (SCLOG) resumes the logging process. If SCLOG CLEAR is specified, the contents of SYS.LOGC are overwritten. If not, the new information is appended to the old.

If the following message appears:

```
LOGC FILE UNAVAILABLE
```

then either the SYS.LOGC file does not exist, or a TCLOG command has been issued.

12.5.2 Printing SYS.LOGC

The file is edited and printed by means of the H_PRLOGC utility. The printout is as close as possible to the image of the hard copy. A pointer, LASTPRINT, is maintained to mark the last line of dialog printed.

The H_PRLOGC utility can be run using ENTER_JOB_REQUEST (for simple cases), or by the JCL statement:

```
STEP H_PRLOGC SYS.HLMLIB [,OPTIONS = 'options'];
```

where 'options' is:

```
{ REPORT = { ALL | hex8 }                                          }
{                                                                  }
{                  [      {LASTP}] [           ] [                ] }
{ REPORT= ACTIVE [,FROM={BEG  }] [,LINES=nnn] [,SUMMARY=/string/] }
{                  [      {hh.mm}] [           ] [                ] }
```

where:

ALL	The whole file is printed.
hex8	Dialog concerning a given CPU is printed. The CPU concerned is identified by a string of 8 hexadecimal characters. The characters are found in the footer of the SYSOUT banner.
ACTIVE	Only those dialog lines concerning the CPU where H_PRLOGC is running are printed.

The following parameters may be used with the ACTIVE option:

FROM = LASTP	Printing is resumed from the LASTPRINT pointer. If this level has been overwritten (because nothing has been printed for a long time), printing starts from the oldest available dialog line.
FROM = BEG	Printing starts from the oldest dialog logged.

If any of the following three parameters are used, the LASTPRINT pointer is not modified and therefore several partial outputs can be made.

FROM = hh.mm	Printing starts from the dialog logged at the given time, hh.mm, within the last 24 hours. If no dialog is found for that time, printing starts from the oldest dialog logged.
LINES = nnn	Specifies the number of lines to be output if output to the current end of file is not required.
SUMMARY=/string/	Only those lines containing the string enclosed within the slashes are printed (for example: a job name, run number or device identification). The string must not exceed 12 characters or contain a slash character.

12.5.3 Archiving the SYS.LOGC File

The following set of JCL statements copies the SYS.LOGC file into the permanent SYSOUT file PR_OUT (H_PRLOGC). Then LIBMAINT copies PR_OUT into the archive MYLOGLIB and changes the name to OUTLOGddmm-hh, where ddmm-hh represents the day, month, and hour of the run.

```
$JOB PRLOGLIB,USER=JM,HOLDOUT,JOR=ABORT;
 VALUES DDMM-HH,ALL,MD=K172,DVC=MS/D500;
$COMMENT 'DDMM-HH TO BE REPLACED BY DAY MONTH-HOUR';
$COMMENT 'ALL MAY BE REPLACED BY ACTIVE,FROM=HH.MM';
 STEP H_PRLOGC,SYS.HLMLIB,OPTIONS='REPORT='&2';';
 ASSIGN H_PR,PR_OUT,RESIDENT,FILESTAT=TEMPRY,END=PASS;
 ALLOCATE H_PR,SIZE=1,INCRSIZE=1,UNIT=CYL,CHECK;
 DEFINE H_PR,CISIZE=1280,RECSIZE=128,RECFORM=F,
        FILEFORM=UFAS;
 ENDSTEP;
 LIBMAINT SL,INFILE=(PR_OUT,RESIDENT,FILESTAT=TEMPRY,
                     END=DEASSIGN)
 ,LIB=(MYLOGLIB DVC=&DVC MD=&MD)
 ,COMMAND='MOVE INFILE:H_PRLOGC'&1';
          LIST OUTLOG*;';
$ENDJOB;
```

Assuming that this set of JCL statements is the member OUTJCL of MYSLLIB, it could be run using the following command:

```
EJR MEMBERS=OUTJCL LIB=MYSSLIB:K172:MS/D500,
    VALUES=(0110-12);
```

The Output Listing

The heading page displays the time and date of the first extracted message.

The heading line of each page displays the CPU name.

The image of the hard copy is left-justified.

Each system restart, or logging restart, is denoted by a few skipped lines and an explicit warning in the righthand margin.

A mini status is given at the end of the printout. This mini status contains the following information:

- Number of lines delivered
- Whether the current file is full
- Whether the LASTPRINT pointer has been updated.

Figure 12-1 is a visual representation of this job.

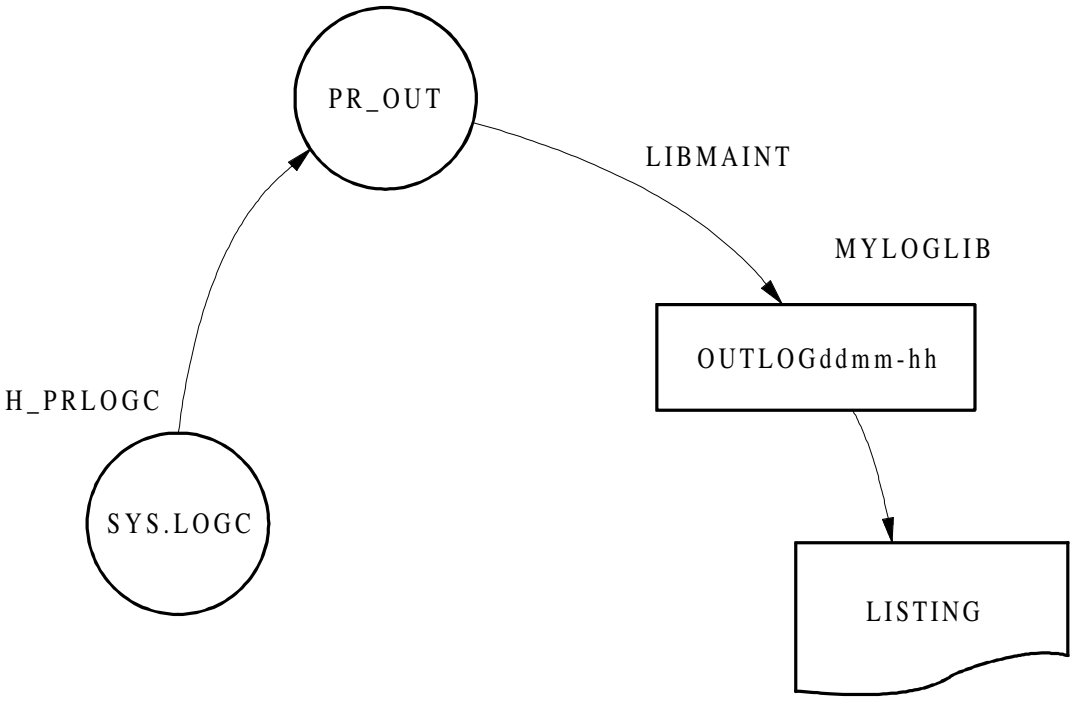

Figure 12-1. Archiving SYS.LOGC

13. System Regulation Procedures

Correcting a performance problem usually involves three levels of approach:

1. ***Obtaining an equilibrium of all the system resources (CPU, MEMORY, DISK I/Os, and MPLs).*** The objective is to maximize throughput while maintaining the maximum load on the different resources.
2. ***Tuning the different environments in relation to one another.*** This objective is dependent upon the performance criteria that have been defined by the System Administrator. The main way of achieving this objective is either to give access priorities to the system resources, or to voluntarily create artificial bottlenecks to slow down certain activities such as those associated with the CPU, the disks, and the job scheduling classes.
3. ***Tuning the processors individually.*** It is important to analyze the requirements of each request in the different processors. These requirements must be quantified and optimized with regard to the service provided. Because IOF is an open-ended and dynamic environment, it is not easy to avoid over-use of system resources. Each processor has different operating modes and reducing resource demand is the most efficient way of tuning them. Optimization is not the same for all systems. Each site has its own performance criteria and behavior patterns. Furthermore, it is easier to analyze and improve upon a system that is naturally unbalanced (by an absence of tuning) rather than a system that is badly tuned by error. Therefore, it is indispensable to start with a minimum-tuned system and not try to re-tune a system upon which attempts at tuning have already been made.

13.1 Disk I/O

Balancing the load on the different spindles by optimizing the file distribution

The I/O throughput can only be an initial approximation of the load on a disk. Refer to the SBR User Guide for details on how to obtain a better approximation.

To obtain reasonable response times, the disk occupation rate must not exceed 40-60%. This range corresponds to about 25-30 I/Os per second for the LSS disks.

SBR provides the figures for the throughput, the length of the queue, and the service time on each spindle (not taking into account the current request). This information indicates the irregularity of the traffic of data and is the best estimation of the system load with regard to I/Os. An average value greater than 0.4 for the queue length often means there are contentions.

It is possible to tune a system that has saturated resources. In this case better results can be obtained by saturating a disk rather than a controller.

Balancing the load on the different controllers by distributing the files on different volumes

Throughput is only an approximation of the actual load. The MSC instrumentation option of the SBR utility can be used to measure controller load.

The load on the controllers influences the global service time of the disks, and therefore indirectly the response time. When there are contentions on a controller, the service time, as seen by SBR, becomes abnormally long for all the disks connected to that controller.

To obtain reasonable response times, a controller must never be loaded to more than 30-50% of its capacity, depending on the model. Experimentally, this value is attained on average with about four disks per controller when these disks are working at maximum throughput.

It is recommended that the disks be shared between two controllers: two controllers for 5-8 spindles. By doing this, the effects of channel saturation are reduced.

Reducing the disk load

For each disk it is recommended to optimize the head movement by allocating contiguously the most frequently accessed files.

The use of disks with high transfer rates in Mbytes/second (LSS) allows the disk load to be reduced as well as that of the controllers.

Determining the load on shared disks

To determine the load on shared disks it is necessary to analyze the load at a given instant on both machines. SBR measures the total load of the controllers, even if the controllers are shared. However, it measures the load of the disks only induced by the system analyzed.

13.2 System Files

Distributing the main system files

The main system files are:

- SYS.BKST[1-15]: contain the pages that have been unloaded from main memory by VMM.
- SYS.LIB[1-15]: contain the PLMs, SMs, and checkpoints.
- SYS.PVMF[1-15]: contain the permanent virtual memory files.
- SYS.TVMF[1-15]: contain the temporary virtual memory files.

It is recommended that each of the above types of file be distributed over several volumes; however, a single volume can contain several file types. That is, the SYS.BKST files, for instance, should be spread over several volumes, but a given volume may contain SYS.BKST, SYS.PVMF, SYS.LIB, and SYS.TVMF files.

SYS.KNODET	description of assigned files.
SYS.SPOOL0-9	working files for GCL.
SYS.OUT	output file (can be multi-volume, <= 10 volumes.)
SITE.CATALOG	user/project/billing information etc.

Providing immediate access to each file

It is recommended that the number of resident disks be minimized. This will reduce the number and the duration of the I/Os during the search for a file at the time it is opened.

The shared and backing store volumes can be non-resident.

Determining the size and allocating SYS.SPOOL0-9

On heavy IOF sites, these files should be allocated on disks that are only lightly loaded. This enables certain stages to be speeded up (logon, entry into processors, etc.).

Similarly, it is preferable not to install SYS.HBINLIB on the same spindle as a SYS.SPOOLi file, because during certain stages there are many I/O transfers between the two files.

There is a volatile storage mechanism at the GCL domain management level of a job. When a domain is initialized for the first time, a subfile (domain subfile) is created in a SYS.SPOOLi file. Following the closure of the domain, this subfile is closed, but not deleted. GCL permits a certain number of dormant subfiles to exist depending on the space available in the SYS.SPOOLi files. If the size of the SYS.SPOOLi files is too small, the dormant subfiles are deleted dynamically and have to be reallocated if necessary. For recommendations on the size of SYS.SPOOLi refer to the *System Installation, Configuration, and Updating Guide*.

Shared disks

In the case of shared disks, a disk lock can cause a blockage. For further details refer to the *Coupled Systems User's Guide*.

Allocating temporary files

Temporary files are allocated on resident disks. If the site requires temporary files, the operator command MDV NTEMP should be used on the disks upon which the temporary files are not to be allocated.

13.3 Memory

Four indications of special importance in tuning VMM are:

- the page fault rate
- the asynchronous page out rate
- the percentage of processes in VMM
- the least recently used (LRU) size

Unusually high values for the first three or an unusually low value for the fourth often mean there is a memory problem. Increasing the size of the swappable memory can produce a great improvement. The % of processes in VMM must not exceed 5% during long intervals.

Two other indicators of the amount of VMM activity are the ratio of the number of missing pages to the number of I/Os (which takes into account the initial renewal of the working sets) and the ratio of the number of I/Os (specific to the missing pages) to the missing pages. The values of these two indicators must be analyzed for all the memory, the dimensions, the IOF, TDS, and Batch activities, the load modules, etc. The usual values under IOF must not exceed 10-20% and 1-1.2 respectively.

To complete the description of VMM activity, it is necessary to separate the shared working sets from the private working sets. To do this, the ratio of the number of missing system segments (types 0, 1, and 10) to the total number of missing segments must be calculated. The value should not exceed 15-25% but this depends on the type of activity (system processors, load modules, etc).

13.4 Telecommunications

Telecommunications aspects are at least as important as those already mentioned, with regard to response times. However, they cannot be treated in detail here, and a simple introduction is given below. For further details, refer to *Networks: Overview and Generation*.

Secondary line speed

The speed of the secondary lines is of major importance when considering response time, particularly when terminals are connected in a cluster.

Polling frequency

The polling frequency (frequency at which the telecommunications system "listens" to the terminals) can influence the times which the system takes to analyze the requests.

NOTE:

SBR does not enable the response times of the telecommunications system to be analyzed.

A. The Standard Families

CAUTION:
The standard environments below are meant to be used as general guides and not for a particular case.

The list of the standard families and their associated commands is given below. For each family, the left column lists the visible commands and the right column lists the hidden commands. The commands that are not mentioned are not accessible for the family.

Families numbered 1 to 52 are defined as standard families. It is advisable to reserve as standard families the ones numbered 1 to 99 (for future extensions).

Family 1: program_prep, proj_manager, file_volume Directives

ENTER_JOB_REQ
CANCEL_JOB
HOLD_JOB
RELEASE_JOB
CANCEL_OUTPUT
HOLD_OUTPUT
RELEASE_OUTPUT

BYE
ALTER_INPUT
DISPLAY_CONFIGURATION

DISPLAY_JOB
DISPLAY_LOAD
DISPLAY_OUTPUT
DISPLAY_PROFILE
DISPLAY_SUBMITTOR
MODIFY_PROFILE
DPRINT
LET
MAIL
SEND
TUTORIAL

CALL
CANCEL_USER_REQ
COMM
DELETE_GLOBAL

DISPLAY_DJP
DISPLAY_IO_CACHE
DISPLAY_DEVIATION_TIME
DISPLAY_SECURITY_OPTIONS
DISPLAY_IOF
DISPLAY_TIME
DISPLAY_REQUEST
DISPLAY_OUPUT_PARAMETERS
DISPLAY_USER_REQ
EJECT
ENTER_FILETRANS_REQ
ENTER_LIBTRANS_REQ
GLOBAL
HOLD_USER_REQ
LIST_GLOBAL
LOG

MODIFY_JOB
MODIFY_OUTPUT
MODIFY_USER_REQ

CANCEL_DUMP
DISPLAY_DUMP
LIST_DUMP

RELEASE_USER_REQ
REPLY

Family 2: Operator Directives

MODIFY_STATION
FORCE_OUTPUT
FORCE_USER_REQ

MODIFY_CONFIGURATION
START_INPUT_READER
START_OUTPUT_WRITER
TERMINATE_INPUT_READER
TERMINATE_OUTPUT_WRITER

START_IO_CACHE
TERMINATE_IO_CACHE
START_IO_SERVER
TERMINATE_IO_SERVER

Family 3: program_prep, prog_manager, file_volume Common Hidden Commands

```
PREPARE_PROGRAM_INT
PREPARE_PROGRAM_BATCH
PREPARE_TPR_INT
PREPARE_TPR_BATCH
CBL
CLANG
COBOL
CONVERT_FORM
DUMPJRNL
FOR77
MAC
PASCAL
GPL

BIND_CU
LINK_PG
EXEC_PG
FSE
EDIT
MAINTAIN_LIBRARY
ROLLFWD
SCANNER
DEBUG
PRINT

MWLIB
MWINLIB
MPRTLIB
MWENVT

MAINTAIN_COMMAND
MAINTAIN_FORM
MAINTAIN_JAS
CREATE_HELP_TEXT

MODIFY_OUTPUT_PARAMETERS
MODIFY_OPERATING_MODE
MAINTAIN_DATA_BASE
REORG_DATA_BASE
IQS
ATTACH_CATALOG
CONNECT_APPLICATION

ALTER_INPUT
ENTER_JOB_REQ
DPRINT(IOF)
```

Family 4: PROGRAM_PREP Tools

```
MAINTAIN_COMMAND          FSE
MAINTAIN_FORM             EDIT
                          PRINT
                          MAINTAIN_LIBRARY
                          MAINTAIN_JAS
```

Family 5: PROJ_MANAGER Tools

```
CREATE_HELP_TEXT          FSE
MAINTAIN_COMMAND          EDIT
MAINTAIN_LIBRARY          PRINT
MAINTAIN_FORM             MAINTAIN_JAS
INIT_TDS_MGT_FILE
```

Family 6: SYSADMIN Tools

```
MAINTAIN_COMMAND          EDIT
CREATE_HELP_TEXT          MAINTAIN_LIBRARY
PRINT                     MAINTAIN_FORM
                          MAINTAIN_JAS
                          FSE
```

Family 7: Hidden Contexts

```
                          MWLIB
                          MWINLIB
                          MPRTLIB
                          MWENVT
                          MODIFY_OUTPUT_PARAMETERS
                          MODIFY_OPERATING_MODE
```

Family 8: Interactive LINKER compilation

```
CBL                       CLANG
COBOL
LINK_PG                   FOR77
PREPARE_PROGRAM_INT        MAC
MAINTAIN_LIBRARY          PASCAL
FSE                       GPL
BIND_CU                   EDIT
                          SCANNER
                          DPRINT (IOF)
                          EXEC_PG
                          DEBUG
                          PRINT
```

Family 9: DEBUG Execution

```
EXEC_PG                    EDIT
DEBUG                      DPRINT (IOF)
PREPARE_PROGRAM_BATCH      ALTER_INPUT
MAINTAIN_LIBRARY           ENTER_JOB_REQ
FSE                        COBOL
                           LINK_PG
                           SCANNER
                           PRINT
                           CBL
```

Family 10: SCANNER

```
SCANNER
```

Family 11: FILE1 (PROGRAM_PREP, PROJ_MANAGER, FILE_VOLUME)

```
PRINT_FILE                 BUILD_FILE
LIST_FILE                  CLEAR_FILE
                           COMPARE_FILE
                           COPY_FILE
                           CREATE_FILE
                           LOAD_FILE
                           MAINTAIN_FILE
                           MODIFY_FILE
                           MERGE_FILE
                           SORT_FILE
                           SAVE_FILE
                           RESTORE_FILE

                           SORT_INDEX

                           BUILD_LIBRARY
                           CLEAR_LIBRARY
                           MAINTAIN_LIBRARY
                           COMPARE_FILESET
                           COPY_FILESET
                           CREATE_FILESET
                           LIST_FILESET
                           LOAD_FILESET
                           PRINT_FILESET
                           RESTORE_FILESET
                           SAVE_FILESET
                           EXPAND_FILESET
```

```
                                LIST_ACL
                                LIST_VOLUME
                                LIST_GEN
                                LIST_FILE_SPACE
                                LIST_DIR
                                LIST_CATALOG
                                LIST_LINK
                                LIST_CATSPACE
                                LIST_QUOTA

                                SHIFT_GEN
```

Family 12: FILE2 (PROGRAM_PREP, PROJ_MANAGER, FILE_VOLUME)

```
BUILD_FILE                      CLEAR_FILE
CREATE_FILE                     COMPARE_FILE
LOAD_FILE                       COPY_FILE
                                MAINTAIN_FILE
                                MODIFY_FILE
                                MERGE_FILE
                                SAVE_FILE
                                RESTORE_FILE
                                LIST_FILE
                                PRINT_FILE
                                SORT_FILE

                                SORT_INDEX

                                CLEAR_LIBRARY
                                BUILD_LIBRARY
                                MAINTAIN_LIBRARY

                                COMPARE_FILESET
                                COPY_FILESET
                                CREATE_FILESET
                                LIST_FILESET
                                LOAD_FILESET
                                PRINT_FILESET
                                RESTORE_FILESET
                                SAVE_FILESET
                                EXPAND_FILESET

                                LIST_ACL
                                LIST_VOLUME
                                LIST_GEN
                                LIST_FILE_SPACE
                                LIST_DIR
                                LIST_CATALOG
                                LIST_LINK
                                LIST_CATSPACE
                                LIST_QUOTA

                                SHIFT_GEN
```

Family 13: FILE3 (PROGRAM_PREP, PROJ_MANAGER)

SORT_INDEX
SORT_FILE

BUILD_FILE
CLEAR_FILE
LOAD_FILE
CREATE_FILE
COMPARE_FILE
COPY_FILE
MAINTAIN_FILE
MODIFY_FILE
MERGE_FILE
SAVE_FILE
RESTORE_FILE
LIST_FILE
PRINT_FILE
BUILD_LIBRARY
CLEAR_LIBRARY
MAINTAIN_LIBRARY

COMPARE_FILESET
COPY_FILESET
CREATE_FILESET
LIST_FILESET
LOAD_FILESET
PRINT_FILESET
RESTORE_FILESET
SAVE_FILESET
EXPAND_FILESET

LIST_ACL
LIST_VOLUME
LIST_GEN
LIST_FILE_SPACE
LIST_DIR
LIST_CATALOG
LIST_LINK
LIST_CATSPACE
LIST_QUOTA

SHIFT_GEN

Family 14: FILE4 (PROJ_MANAGER, FILE_VOLUME)

DELETE_FILE
CLEAR_FILE
COMPARE_FILE

BUILD_FILE
COPY_FILE
CREATE_FILE
COPY_FILE
LOAD_FILE
MAINTAIN_FILE
MODIFY_FILE
MERGE_FILE
SORT_FILE
SAVE_FILE
RESTORE_FILE
LIST_FILE
PRINT_FILE

SORT_INDEX

CLEAR_LIBRARY
BUILD_LIBRARY
DELETE_LIBRARY
MAINTAIN_LIBRARY

COMPARE_FILESET
COPY_FILESET
CREATE_FILESET
LIST_FILESET
LOAD_FILESET
DELETE_FILESET
PRINT_FILESET
SAVE_FILESET
EXPAND_FILESET
RESTORE_FILESET
LIST_ACL
LIST_VOLUME
LIST_FILE_SPACE
LIST_CATSPACE
LIST_QUOTA

CREATE_LINK
DELETE_LINK
LIST_LINK

CREATE_GEN
DELETE_GEN
LIST_GEN
MODIFY_GEN
SHIFT_GEN

COPY_CATALOG
CREATE_CATALOG
LIST_CATALOG
DELETE_CATALOG
MODIFY_CATALOG
CREATE_MT_FILE
MAINTAIN_DATA_BASE
REORG_DATA_BASE

Family 15: FILE5 (BG_PROJ_MANAGER)

CREATE_MT_FILE
MAINTAIN_DATA_BASE
MAINTAIN_FILE
REORG_DATA_BASE

BUILD_FILE
COPY_FILE
CREATE_FILE
LOAD_FILE
MODIFY_FILE
DELETE_FILE
CLEAR_FILE
COMPARE_FILE
MERGE_FILE
SORT_FILE
SAVE_FILE
RESTORE_FILE
LIST_FILE
PRINT_FILE

SORT_INDEX

CLEAR_LIBRARY
BUILD_LIBRARY
MAINTAIN_LIBRARY
DELETE_LIBRARY

COPY_FILESET
SAVE_FILESET
RESTORE_FILESET

CREATE_DIR
DELETE_DIR
LIST_DIR

LIST_ACL
LIST_VOLUME
LIST_FILE_SPACE
LIST_CATALOG
LIST_CATSPACE
LIST_QUOTA

Family 16: FILE6 (FILE_VOLUME)

COPY_FILE
SAVE_FILE
RESTORE_FILE

Family 17: FILE7 (FILE_VOLUME)

```
CREATE_DK_FILE
CREATE_MT_FILE
```

Family 18: File Space (FILE_VOLUME)

```
LIST_FILE_SPACE
LIST_QUOTA
MODIFY_FILE_SPACE

SORT_FILE
MERGE_FILE
MODIFY_FILE
```

Family 19: Libraries

```
BUILD_LIBRARY
CLEAR_LIBRARY
DELETE_LIBRARY
PRINT
```

Family 20: Directories

```
CREATE_DIR
DELETE_DIR
LIST_DIR
```

Family 21: CATALOG1

```
COPY_CATALOG
CREATE_CATALOG
MODIFY_CATALOG
DELETE_CATALOG
LIST_CATALOG
```

Family 22: CATALOG2

```
ATTACH_CATALOG
SAVE_CATALOG
RESTORE_CATALOG
```

Family 23: LINK

```
CREATE_LINK
DELETE_LINK
LIST_LINK
```

Family 24: GEN1

```
CREATE_GEN
DELETE_GEN
MODIFY_GEN
LIST_GEN
SHIFT_GEN
```

Family 25: GEN2

```
SHIFT_GEN
```

Family 26: Access Rights

```
LIST_ACL
MODIFY_ACL
```

Family 27: FILESET1

```
COMPARE_FILESET
CREATE_FILESET
DELETE_FILESET
EXPAND_FILESET
LIST_FILESET
LOAD_FILESET
PRINT_FILESET
```

Family 28: FILESET2

```
COPY_FILESET
SAVE_FILESET
RESTORE_FILESET
```

Family 29: VOLUME

```
LIST_VOLUME
MAINTAIN_VOLUME
CLEAR_VOLUME
```

Family 30: Prepare

```
PREPARE_VOLUME
PREPARE_DISK
PREPARE_DISKETTE
PREPARE_TAPE
MODIFY_DISK
RESTORE_DISK
SAVE_DISK
```

Family 31: SYSADMIN commands

```
MAINTAIN_CATALOG
MAINTAIN_MIGRATION
BUILD_SYSTEM
CREATE_QUOTA_FILE
DELETE_QUOTA_FILE
INVALIDATE_QUOTA_FILE
MAINTAIN_QUOTA
MODIFY_QUOTA_FILE
VALIDATE_QUOTA_FILE
LIST_QUOTA
```

Family 32: Catalog Space

```
LIST_CATSPACE
MODIFY_CATSPACE
```

Family 33: SYSADMIN file

```
MODIFY_DISK
PREPARE_DISK
PREPARE_TAPE
CREATE_FILE
DELETE_FILE
SAVE_FILE
RESTORE_FILE
RESTORE_DISK
SAVE_DISK
LIST_FILE
FILL_MIRROR
START_MIRROR
```

Family 34: Hidden FILE_VOLUME commands

```
ATTACH_CATALOG
COPY_CATALOG
CREATE_CATALOG
DELETE_CATALOG
LIST_CATALOG
MODIFY_CATALOG
SAVE_CATALOG
RESTORE_CATALOG
RESTORE_DISK
SAVE_DISK

PREPARE_DISK
PREPARE_DISKETTE
PREPARE_TAPE
PREPARE_VOLUME

CLEAR_VOLUME
LIST_VOLUME
MAINTAIN_VOLUME
MAINTAIN_DATA_BASE

REORG_DATA_BASE
MODIFY_FILE_STATUS
IQS
MODIFY_DISK
SORT_INDEX

BUILD_FILE
COPY_FILE
CREATE_FILE
LOAD_FILE
MAINTAIN_FILE
MODIFY_FILE
MERGE_FILE
SORT_FILE
DELETE_FILE
CLEAR_FILE
COMPARE_FILE
LIST_FILE
PRINT_FILE
SAVE_FILE
RESTORE_FILE
CREATE_DK_FILE
CREATE_MT_FILE

CLEAR_LIBRARY
BUILD_LIBRARY
DELETE_LIBRARY
MAINTAIN_LIBRARY
```

```
CREATE_FILESET
COMPARE_FILESET
COPY_FILESET
LIST_FILESET
LOAD_FILESET
DELETE_FILESET
PRINT_FILESET
SAVE_FILESET
RESTORE_FILESET
EXPAND_FILESET

LIST_FILE_SPACE
LIST_QUOTA
MODIFY_FILE_SPACE

CREATE_LINK
DELETE_LINK
LIST_LINK

CREATE_GEN
DELETE_GEN
MODIFY_GEN
LIST_GEN
SHIFT_GEN

CREATE_DIR
DELETE_DIR
LIST_DIR

LIST_CATSPACE
MODIFY_CATSPACE

LIST_ACL
MODIFY_ACL

FILL_MIRROR
START_MIRROR
```

Family 35:

```
BACKGROUND_PROGRAM_PREP
```

Family 36:

```
RETURN_PROGRAM_PREP
```

Family 37:

BACKGROUND_PROJ_MANAGER

Family 38:

RETURN_PROJ_MANAGER

Family 39:

BACKGROUND_FILE
BACKGROUND_CATALOG
BACKGROUND_VOLUME

Family 40:

RETURN_FILE_VOLUME

Family 41:

RETURN_FILE_VOLUME_

Family 42:

BACKGROUND_FILE_
BACKGROUND_CATALOG_
BACKGROUND_VOLUME_

Family 43:

RETURN_SYSADMIN

Family 44:

RETURN_SYSADMIN_

Family 45:

Reserved for future use

Family 46:

Reserved for future use

Family 47: FILE10 (FILE_VOLUME)

```
MAINTAIN_FILE
MODIFY_FILE_STATUS
MODIFY_FILE
```

Family 48: SYSADMIN directives

```
ENTER_JOB_REQ
ENTER_FILETRANS_REQ
ENTER_LIBTRANS_REQ
CANCEL_JOB
HOLD_JOB
RELEASE_JOB
CANCEL_OUTPUT
HOLD_OUTPUT
RELEASE_OUTPUT
CANCEL_USER_REQ
BYE
DISPLAY_CONFIGURATION
DISPLAY_LOAD
DISPLAY_IOF
DISPLAY_JOB
DISPLAY_OUTPUT
DISPLAY_SUBMITTOR
DISPLAY_USER_REQ
MAIL
SEND
DPRINT
HOLD_TERMINAL_OUTPUT
RELEASE_TERMINAL_OUTPUT
CANCEL_TERMINAL_OUTPUT
DISPLAY_TERMINAL_OUTPUT
 ALTER_INPUT
DISPLAY_PROFILE
 MODIFY_PROFILE
 DPRINT (IOF)
LET
 TUTORIAL
CALL
 COMM
 DELETE_GLOBAL
 DISPLAY_DJP
  DISPLAY_IO_CACHE
  DISPLAY_REQUEST
  DISPLAY_TIME
  DISPLAY_OUPUT_PARAMETERS
 EJECT
 GLOBAL
HOLD_USER_REQ
LIST_GLOBAL
 LOG
 MODIFY_IO_CACHE
 MODIFY_JOB
  MODIFY_OUTPUT
  MODIFY_USER_REQ
  RELEASE_USER_REQ
 REPLY
```

Family 49: Interactive TPR preparation

```
EXEC_PG                    CBL
FSE                        COBOL
PREPARE_TPR_INT            DEBUG
LINK_PG                    DPRINT
MAINTAIN_FORM              EDIT
MAINTAIN_LIBRARY           MAC
PREPARE_TPR_BATCH          FOR77
PRINT                      GPL
                           PASCAL
                           SCANNER
                           ALTER_INPUT
                           ENTER_JOB_REQ
                           BIND_CU
                           CLANG
```

Family 50: All hidden SYSADMIN commands

```
IOF
H_NOCTX
MAINTAIN_LIBRARY
```

Family 51:

```
BACKGROUND_TDSAPPL_PREP
```

Family 52:

```
RETURN_TDSAPPL_PREP
```

B. Environment Management Examples

B.1 Modifying a Standard Environment

The standard environments are environment prototypes that may be modified to fit the needs of the user. The simplest modification consists of adding a command to, or removing it from an existing environment (or making it visible or hidden). More complex modifications may be made which are combinations of these simple changes.

The following examples illustrate this kind of operation.

EXAMPLE 1: Making a Non-accessible Command Accessible

Problem

The PROJ_MANAGER standard environment suits your needs but you want to make accessible (but keep hidden) the MODIFY_FILE_SPACE command which is reserved for users working in the FILE_VOLUME or SYSADMIN environments.

Solution

The MODIFY_FILE_SPACE command belongs to the standard families:

- 18 (for which it is visible)
- 34 and 50 (for which it is hidden)

If you refer to the standard families list (see Appendix A of this manual) you will notice that giving the PROJ_MANAGER environment access to one of these families would not help. This is because:

- access to family 18 would make the command visible (you want it hidden)
- access to family 34 or 50 would allow the user to access commands that must remain inaccessible (for example PREPARE_VOLUME).

A possible solution would be to modify one of these standard families. This is not recommended, however, as it disturbs the definitions of the other environments (FILE_VOLUME, SYSADMIN, and their subenvironments).

It is not advisable (although it is permitted) to modify the standard families if a simpler solution can be found.

In this example, it is better to define a new family (say 100) which contains only the MODIFY_FILE_SPACE command as a hidden command.

You therefore have to modify the MDA instructions that apply to the MODIFY_FILE_SPACE command in the standard environment generation GCL. This allows it to be accessed from family 100.

The lines to be modified are listed below:

Standard GCL

```
MDA MODIFY_FILE_SPACE  PRTY=115
                       HIDE=(34 50)
                       NO_ACCESS=(1-17 19-33 35-49 51-256);
```

New GCL

```
MDA MODIFY_FILE_SPACE  PRTY=115
                       HIDE=(34 50 100)
                       NO_ACCESS=(1-17 19-33 35-49 51-99 101-256);
```

The ENVT commands that create the environments must be modified in the standard environment installation job to take into account the new family (as the effect of the ENVT commands is not cumulative).

```
ENVT PROJ_MANAGER (1 2 3 5 7 11 12 13 14 19 20 37 100);
ENVT BG_PROJ_MGER (1 2 3 7 15 21 23 24 25 27 38 100);
```

❑

EXAMPLE 2: Hiding a Command

Problem

The PROGRAM_PREP standard environment suits your needs but you want to make the PRINT_FILE command hidden (but accessible) in the main environment.

Solution

In both PROGRAM_PREP and BG_PROG_PREP environments, the PRINT_FILE command is visible because access is given to standard family 11.

In this example, it is easier to modify the standard families than keep them and define a new family, because standard family 11 contains many hidden commands that are not contained as a whole in another standard family (see Appendix A for a description of family 11).

To define a new family that contains all the commands of the family 11, except PRINT_FILE, would mean updating as many MDA commands as there are hidden commands, in the environment generation standard GCL.

However, an easier way is to:

- hide the PRINT_FILE command in standard family 11
- make it visible in the standard family 12.

The standard environment definitions are then unchanged if families 11 and 12 are used together in all the environments.

The only exception is the TDSAPPL_PREP environment for which either the same option is taken (i.e., hide the PRINT_FILE command in the main environment) or the PRINT_FILE command is made available in standard family 49 (or a new family) for which it is visible.

In the latter case, perform the following operations:

- change the MDA command that applies to PRINT_FILE in the standard GCL:

```
MDA PRINT_FILE  PRTY=80
                HIDE=(11-15 34 50)
                NO_ACCESS=(1-10 16-33 35-48 51-256);
```

- leave the environment definitions unchanged
- update the standard family list given in Appendix A

❑

EXAMPLE 3: Modifying a Non-Standard Procedure

Problem

The program preparation procedures (PREPARE_PROGRAM_INT and PREPARE_PROGRAM_BATCH) suit your needs but you want to use EDIT instead of Full Screen Editor (FSE) to modify your programs.

Solution

If you want to edit a program in both PPI and PPB, the MODIFY command of FSE is executed allowing you to modify your program.

The corresponding sequence of code in the PPI and PPB procedures is:

```
LET L_PGP_ANSWER
  #QUERY(#CAT('DO YOU WANT TO EDIT ',%G_PGP_PROG,'?'));
IF %L_PGP_ANSWER;
  AI C_PGP_FSE_MODIFY   LIB=SYS.HSLLIB;
  CALL FSE LIB=%SLLIB;
ENDIF;
```

C_PGP_FSE_MODIFY is a subfile stored in the SYS.HSLLIB library. Its format is:

```
MODIFY %G_PGP_PROG;
```

You must replace this by a sequence of code which calls EDIT, loads the program in a workspace, and allows you to modify it. To do this you have to:

- replace the standard subfile C_PGP_FSE_MODIFY by a user subfile C_PGP_EDIT_READ, whose format is R[v(%G_PGP_PROG) and store it in a user SL library.
- replace the sequence of code above by:

```
LET L_PGP_ANSWER
    #QUERY(#CAT('DO YOU WANT TO EDIT ',%G_PGP_PROG,'?'));
IF %L_PGP_ANSWER;
    AI C_PGP_EDIT_READ LIB=MYLIB;
    CALL EDIT LIB=%SLLIB;
ENDIF;
```

MYLIB is the user SL library where the subfile C_PGP_EDIT_READ is stored.

NOTE:
This modification must be performed in both PPI and PPB procedures.

❑

B.2 Creating a New Environment

This subsection describes how to define a new environment. The simplest cases are similar to the previous examples, the only difference being that a standard environment is duplicated. In this case, you may use the same mechanisms and just enter a different ENVT command to create a new environment.

More complex examples are given below.

EXAMPLE 1: Combining Two Existing Environments

Problem

You wish to define a single environment that combines the functions contained in both PROGRAM_PREP and PROJ_MANAGER.

Solution

You want to define a main environment, named EXT_PG_PREP that shares and gives access to the same commands as PROGRAM_PREP and PROJ_MANAGER. You also want two subenvironments for file management and catalog management (as for the standard FILE_VOLUME environment).

The PROGRAM_PREP environment gives access to families:

1 2 3 7 8 9 10 11 and 35

The PROJ_MANAGER environment gives access to families:

1 2 3 5 7 11 12 13 14 19 20 and 37

Families 35 and 37 contain the dedicated commands:

```
BACKGROUND_PROGRAM_PREP
BACKGROUND_PROJ_MANAGER
```

which you want to replace by two new commands, for example:

```
BACKGROUND_FILE__
BACKGROUND_CATALOG__
```

(The suffix "__" is needed to prevent confusion with commands used in the FILE_VOLUME and SYSADMIN environments).

The main environment EXT_PG_PREP will be given access to the families:

1 2 3 5 7 8 9 10 11 12 13 14 19 20 and 101

Family 101 is assumed to contain the two background commands, as visible commands.

The visible commands are listed below together with their assigned priorities:

SCREEN 1

```
PREPARE_PROGRAM_BATCH       PRIORITY=50
PREPARE_PROGRAM_INT

COBOL                        PRIORITY=55
EXEC_PG
FSE
LINK_PG
BUILD_LIBRARY               PRIORITY=56
CLEAR_LIBRARY
DELETE_LIBRARY
MAINTAIN_LIBRARY

CREATE_HELP_TEXT            PRIORITY=70
MAINTAIN_COMMAND
MAINTAIN_FORM
```

SCREEN 2

```
BUILD_FILE                   PRIORITY=80
CLEAR_FILE
COMPARE_FILE
CREATE_FILE
DELETE_FILE
LIST_FILE
LOAD_FILE
PRINT_FILE
SORT_FILE
SORT_INDEX

CREATE_DIR                   PRIORITY=90
DELETE_DIR
LIST_DIR

INIT_TDS_MGT_FILE           PRIORITY=96
BACKGROUND_CATALOG__        PRIORITY=97
BACKGROUND_FILE__
```

SCREEN 3

```
ENTER_JOB_REQ                PRIORITY=160

CANCEL_JOB                   PRIORITY=165
        .
        .
        .
```

(Identical to the last screens of PROGRAM_PREP and PROJ_MANAGER). This command list shows that a priority of 97 may be associated with the background commands so that they will appear at the bottom of the second screen.

All the visible commands are shown on a menu of four screens.

No modification has to be made in the standard families. But two new families have to be defined (for example, 101 and 102) that contain the background commands and the RETURN_EXT_PG_PREP command (which is defined for the background environment).

Note that all the commands needed for generating these new environments may be stored in a separate GCL member.

To resume:

1. Define the BACKGROUND_FILE__ and BACKGROUND_CATALOG__ commands in the IOF domain.
   - associated priority : 97
   - associated family : 101
   - same contents as BACKGROUND_FILE procedure

```
PROC(BACKGROUND_FILE__,BKF__)
    PROMPT='switch to environment BG_FILE__'
    ACCESS=101
    PRTY=97;
MWENVT BG_FILE__;
ENDPROC;

PROC (BACKGROUND_CATALOG__, BKC__)
     PROMPT='switch to environment BG_CATALOG__'
     ACCESS=101
     PRTY=97;
MWENVT BG_CATALOG__;
ENDPROC;
```

2. Define the RETURN_EXT_PG_PREP command in the IOF domain
   - associated priority: 145
   - associated family: 102
   - same contents as RETURN_PROGRAM_PREP

```
PROC (RETURN_EXT_PG_PREP,RTXPP)
     PROMPT='switch to environment EXT_PG_PREP'
     ACCESS=102
     PRTY=145;
MWENVT EXT_PG_PREP;
ENDPROC;
```

3. Define the EXT_PG_PREP environment and its subenvironments:

```
ENVT EXT_PG_PREP (1 2 3 5 7 8 9 10 11 12 13 14 19 20 101);
ENVT BG_FILE__ (1 2 3 7 11 12 14 16 18 19 27 28 34 47 102);
ENVT BG_CATALOG__ (1 2 3 7 20 21 22 23 24 26 32 34 102);
```

The last two environment definitions are identical to those of BG_FILE and BG_CATALOG standard environments, except that family 102 (RETURN_EXT_PG_PREP) is substituted for 40 (RETURN_FILE_VOLUME).

❑

EXAMPLE 2: Substituting One Command for Another in an Existing Environment

Problem

To define a specific environment for FOR77 programmers.

Solution

This kind of problem has two aspects:

1. Defining a new visibility for the menu screens and making inaccessible those commands that are not needed
2. Tailoring the dialoging procedures of the PROGRAM_PREP environment (PPI, PPB) for the FOR77 programmer.

There are too many modifications for them to be given in detail but some guidelines are given below.

Defining a New Visibility

The new environment will be built using the PROGRAM_PREP environment as a model. Families 8 and 9 will be replaced by two new families (for example 103 and 104) which are identical to 8 and 9 except that COBOL has been replaced by FOR77 and other compilers are not accessible.

This means that 11 MDA commands will be modified in the environment generation standard GCL.

However, all the hidden commands are also contained in the standard family 3. Thus, the minimal family specification only concerns the commands shown below:

```
FOR77                 }
LINK_PG               }
                      }  in family 103
MAINTAIN_LIBRARY      }
FSE                   }


EXEC_PG               }
DEBUG                 }
                      }  in family 104
MAINTAIN_LIBRARY      }
FSE                   }
```

Only 6 MDA commands are therefore affected in the standard GCL.

Note that the commands that may replace PPI and PPB for the FORTRAN programmer may be inserted in families 103 and 104 respectively.

Take the example of entering the FSE command in families 103 and 104.The same operations are performed on the MDA statements that apply to the other commands that are to be entered in those families.

Standard GCL

```
MDA  FSE  PRTY=55
          HIDE=(3-6 50)
          NO_ACCESS=(1-2 7 10-48 51-256);
```

New GCL

```
MDA  FSE  PRTY=55
          HIDE=(3-6 50)
          NO_ACCESS=(1-2 7 10-48 51-102 105-256);
```

Tailoring the Dialoging Procedures for the FORTRAN Programmer

Note that several components are affected during this operation.

- the GCL procedures (PPI, PPB and procedures called from them)
- the source members used as comfiles or GCL executed in batch mode
- Help texts

1. Tailoring the PPI Command and its Items for the FORTRAN Programmer

PPI enables you to edit, compile, link and execute COBOL programs. The modifications necessary to adapt it to the FORTRAN programmer are concerned with the compilation step. PPI calls the COBOL_OPT_PPI_ procedure which:

- declares the parameters for the COBOL compilation
- calls the COBOL_OPT_PPI_NAME procedure which assigns the parameters or the user procedure whose name is given by the global variable G_PGP_COBOL_OPT_PPI as explained in Chapter 9.
- calls COBOL
- uses subfiles:

  ```
  C_PGP_INIT_CR_SEV2
  C_PGP_INIT_CR_SEV3
  C_PGP_INIT_CR_SEV4
  ```

  to write the results of compilation, and which uses subfiles:

  ```
  C_PGP_EDIT_CR_SEV2
  C_PGP_EDIT_CR_SEV3
  C_PGP_EDIT_CR_SEV4
  ```

  for the outputs of COBOL compilations.

You therefore have to:

- create a procedure PPI_FORTRAN, like PPI, which:
  - calls FORTRAN_OPT_PPI_ instead of COBOL_OPT_PPI_
  - uses subfiles:

    ```
    C_PGP_INIT_CR_SEV2_FOR
    C_PGP_INIT_CR_SEV3_FOR
    C_PGP_INIT_CR_SEV4_FOR
    ```

    instead of:

    ```
    C_PGP_INIT_CR_SEV2
    C_PGP_INIT_CR_SEV3
    C_PGP_INIT_CT_SEV4
    ```

- create a hidden procedure FORTRAN_OPT_PPI_, which:
  - declares the FORTRAN parameters
  - calls the procedure FORTRAN_OPT_PPI_NAME, or, if it exists, a user procedure whose name is given by a global variable G_PGP_FORTRAN_OPT_PPI
  - calls FORTRAN.
- create a hidden procedure FORTRAN_OPT_PPI_NAME which assigns parameters of FORTRAN.

  Note that all the procedures called by PPI, except COBOL_OPT_PPI_, are called by PPI_FORTRAN. So they must be accessible to the environment of the FORTRAN programmer.
- create subfiles:

```
C_PGP_INIT_CR_SEV2_FOR
C_PGP_EDIT_CR_SEV2_FOR
C_PGP_INIT_CR_SEV3_FOR
C_PGP_EDIT_CR_SEV3_FOR
C_PGP_INIT_CR_SEV4_FOR
C_PGP_EDIT_CR_SEV4_FOR
```

- to use the outputs of FORTRAN compilations
- create the PPI_FORTRAN Help texts like those for the PPI command

2. Tailoring the PPB Command and its Items for the FORTRAN Programmer

PPB calls the procedure EJR_COB_OPT_PPB which:

- declares the parameters of ENTER_JOB_REQ
- calls the procedure EJR_COB_OPT_NAME, which assigns these parameters or the user procedure whose name is given by the global variable G_PGP_EJR_COBOL_OPT, as explained in Chapter 9.
- executes in batch the C_PGP_GCL_COBOL member.

The C_PGP_GCL_COBOL subfile uses other subfiles:
COBOL_OPT_PPB_NAME or, if it exists, a user subfile whose name is given by the global variable G_PGP_COBOL_OPT_PPB and which contains the COBOL step and:

```
C_PGP_EDIT_CR_NOLISTING
C_PGP_EDIT_CR_PPB_SEV2
C_PGP_EDIT_CR_PPB_SEV3
C_PGP_EDIT_CR_PPB_SEV4
```

which use the outputs of COBOL compilations. You therefore have to:

- create a procedure PPB_FORTRAN, like PPB which calls EJR_FORTRAN_OPT_PPB instead of EJR_COB_OPT_PPB
- create a procedure EJR_FORTRAN_OPT_PPB which declares the parameters of ENTER_JOB_REQ and calls the procedure EJR_FOR_OPT_NAME. This assigns these parameters or, if it exists, the user procedure whose name is given by the global variable G_PGP_EJR_FOR_OPT, as explained in Chapter 9. It also executes in batch the C_PGP_GCL_FORTRAN member.
- create a procedure EJR_FOR_OPT_NAME. Note that all the procedures called by PPB, except EJR_COB_OPT_PPB, are called by PPB_FORTRAN, so they must be accessible to the FORTRAN programmer.
- create a subfile C_PGP_GCL_FORTRAN which uses FORTRAN_OPT_PPB_NAME, or a user subfile, whose name is given by G_PGP_FOR_OPT_PPB, and which contains the FORTRAN step, and the subfiles:

  ```
  C_PGP_EDIT_CR_NOLISTING_FOR
  C_PGP_EDIT_CR_PPB_SEV2_FOR
  C_PGP_EDIT_CR_PPB_SEV3_FOR
  C_PGP_EDIT_CR_PPB_SEV4_FOR
  ```

  which use the outputs of FORTRAN compilations.
- create the subfiles:

  ```
  FORTRAN_OPT_PPB_NAME
  C_PGP_EDIT_CR_NOLISTING_FOR
  C_PGP_EDIT_CR_PPB_SEV2_FOR
  C_PGP_EDIT_CR_PPB_SEV3_FOR
  C_PGP_EDIT_CR_PPB_SEV4_FOR
  ```

- create the PPB_FORTRAN Help texts, like those for the PPB command.

❑

EXAMPLE 3: Making a Command Inaccessible in an Existing Environment

Problem

You wish to define an environment similar to the standard PROGRAM_PREP environment but where the users are not allowed to access any interactive compilers.

Solution

This example is similar to Example 2. The new environment will use the PROGRAM_PREP environment as a model. The standard families 3, 8, and 9 must be removed from the family list to which the new environment gives access.

New families must be defined in order to give access to those commands that belong to families 3, 8, and 9 and do not belong to the other standard families.

These are the following:

```
Visible                    Hidden

PREPARE_PROGRAM_BATCH      EDIT
DEBUG                       MAINTAIN_COMMAND
EXEC_PG                     MAINTAIN_FORM
FSE                         CREATE_HELP_TEXT
LINK_PG                     ATTACH_CATALOG
MAINTAIN_LIBRARY           MAINTAIN_DATA_BASE
                            CONNECT_APPLICATION
                            DISPLAY_SYSTEM_STATUS
```

You then have to create new families that contain the above commands.

Suppose you want to create both this environment (Example 3) and the one described in Example 2 (for a FORTRAN programmer). Ideally, you need to create families that can be used for both environments. For example, the FOR77 command might be removed from family 103 and defined alone in another family (e.g., family 105).

Similarly, families 103 and 104 may contain hidden commands that are not strictly necessary for the definition of the FOR77 programmer environment (Example 2) because they also belong to family 3. However, they may be reused for the environment that forbids the use of interactive compilers.

In this case, the following families may be defined:

Family	Visible Commands	Hidden Commands
103	LINK_PG	EDIT
	MAINTAIN_LIBRARY	SCANNER
	FSE	DPRINT
		EXEC_PG
		DEBUG
104	EXEC_PG	EDIT
	DEBUG	DPRINT
	MAINTAIN_LIBRARY	LINK_PG
	FSE	SCANNER
105	FOR77	EDIT
		FSE
106	PREPARE_PROGRAM_BATCH	MAINTAIN_COMMAND
		MAINTAIN_FORM
		CREATE_HELP_TEXT
		MAINTAIN_DATA_BASE
		ATTACH_CATALOG
		CONNECT_APPLICATION
		DISPLAY_SYSTEM_STATUS

To resume, the operations to be performed are:

- modify the MDA statements that apply to the specified commands in the standard environment generation GCL
- define the environments:

```
ENVT BATCH_PGPREP (1 2 7 10 11 103 104 106)
ENVT FOR_PGPREP (1 2 3 7 10 11 103 104 105)
```

❑

EXAMPLE 4: Defining a Dedicated "Station Operator" Environment

In Chapter 9 we saw how operator rights are considered by GCL as a feature different from the environments (whose effects may be cumulative). A station operator may therefore be allowed to work at a terminal as a normal end user if he has IOF and GCL operator rights.

However, you may wish to define a dedicated environment that gives a particular user only specific station operator commands with some IOF directives.

Suppose you want to give this operator access to the directives contained in families 1 and 48 and the context command of family 7. By writing:

```
ENVT STATION_OP (1 2 7 48);
```

you will get an environment that displays a menu of one screen containing the 10 station operator commands, and a second and third screen that mix the operator commands shown on the menus of users working in the PROGRAM_PREP and SYSADMIN environments.

All other non-visible directives are accessible but hidden, together with a few context commands. Note that such a specific environment must be made accessible only to users that are actually given the station operator right in the catalog.

❑

EXAMPLE 5: Creating New Families - Directives Subenvironment

Using a single command called DIRECTIVES, you can make accessible a whole menu of directives from the PROGRAM_PREP environment.

To do this, you have to define an environment DIRECTIVES in which all directives are given and create a DIRECTIVES procedure in the IOF domain that permits switching to this environment.

The Standard Families Description in Appendix A shows that:

- Family 1 contains directives that are common to the PROGRAM_PREP and PROJ_MANAGER environments
- Family 2 contains the station operator directives
- Family 48 contains the SYSADMIN directives

Giving access to families 1, 2, and 48 will give access to the whole set of directives. However, some of them will be hidden. You then have to define a new family of visible commands (for example 107) that only contains those directives that are not visible in either family 1 or family 48.

If we take the CALL and COMM commands as hidden commands, the new family (107) will contain the following set of visible commands:

```
GLOBAL
DELETE_GLOBAL
LIST_GLOBAL
EJECT
DISPLAY_DJP
HOLD_USER_REQ
LOG
MODIFY_JOB
MODIFY_OUTPUT
MODIFY_USER_REQ
RELEASE_USER_REQ
REPLY
```

You have to define appropriate priorities for these commands so that they appear in the right order in the menu.

You can reuse the RETURN_PROGRAM_PREP command by making it accessible in the DIRECTIVES subenvironment. To do this, put family 36 in the family list that defines the DIRECTIVES environment.

The following steps are necessary to generate the different features of this environment:

1. Define family 107 as described above by modifying the MDA commands of the standard environment generation GCL.
2. Create the DIRECTIVES environment with:

   ```
   ENVT DIRECTIVES (1 2 36 48 107);
   ```

3. Create a DIRECTIVES procedure in the IOF domain, using the BACKGROUND_PROGRAM_PREP procedure as a model. This procedure may be attached to family 7 as a visible command and assigned a priority of 95 so that it will appear at the bottom of the first menu screen of PROGRAM_PREP environment.

❑

EXAMPLE 6: Personalized Environments

This section does not explain the whole operation necessary to define a personalized environment, as this depends on what you want to be presented and accessible.

Suppose you want to assign a specific set of commands (for example, dealing with finance management) to a group of users.

You create an environment FINANCE that has only 3 commands, for example FORECASTS, STATISTICS, and PASCAL. Other commands may be accessible though not presented. You have to decide which ones.

PASCAL is a standard command. FORECASTS and STATISTICS are your own commands that either switch into a subenvironment (giving access to a subset of commands) or invoke your own interactive load module that may also have a GCL menu-driven dialog with the end user.

This environment will be defined to give access to one or more user families (100 to 255) and possibly to some standard families (though usually a very restricted set of functions will be made available).

This kind of environment, which has no relation to the standard environment, needs the following:

- specific programs that may invoke the GCL facility
- GCL procedures that invoke these programs
- environments with associated commands for switching from one to another
- priorities and families associated with each user procedure

❑

B.3 Customizing a Standard Environment - Examples

EXAMPLE 1: Interactive COBOL Options: Parameterization

A user procedure (COBOL_OPTIONS) is stored in the user library. It is substituted for the standard COBOL options.

Startup Sequence

```
GLOBAL G_PGP_COBOL_OPT_PPI  NAME;
LET G_PGP_COBOL_OPT_PPI  COBOL_OPTIONS;
MWINLIB BIN .BINLIB;
```

User Procedure (Stored in .BINLIB)

```
PROC COBOL_OPTIONS ACCESS=(3, 8)
                    HIDE=(3, 8);
LET EXPLIST 1;
LET MAP 1;
LET DIAG AFT;
LET SILENT 1;
LET DDLIST 1;
LET OBJ 0;
LET DUMP DATA;
ENDPROC;
```

❑

EXAMPLE 2: EJR Options: Parameterization

A user procedure (EJR_OPTIONS) is stored in the user library. It is substituted for the standard EJR options at both program compilation and execution.

Startup Sequence

```
GLOBAL G_PGP_EJR_COBOL_OPT NAME;
GLOBAL G_PGP_EJR_EXEC_OPT NAME;
LET G_PGP_EJR_COBOL_OPT EJR_OPTIONS;
LET G_PGP_EJR_EXEC_OPT EJR_OPTIONS;
MWINLIB BIN .BINLIB;
```

User Procedure (Stored in .BINLIB)

```
PROC EJR_OPTIONS ACCESS=(3, 9)
                 HIDE=(3, 9);
LET CLASS B;
LET PRIORITY 3;
LET HOLDOUT 1;
ENDPROC;
```

❑

EXAMPLE 3: User Startup for TDSAPPL_PREP Environment

Global variables are initialized which prevent the environment user from entering the KEY_IN_PARA procedure.

```
GLOBAL G_TAP_TDS          CHAR 4;
GLOBAL G_TAP_TDS_CLASS    CHAR 1;
GLOBAL G_TAP_WORK_CLASS   CHAR 1;
GLOBAL G_TAP_GCLLIB      LIB 78;
GLOBAL G_TAP_GCLMB        CHAR 12;
GLOBAL G_TAP_SM           CHAR  4;
GLOBAL G_TAP_SM1          LIB  78;
GLOBAL G_TAP_SM2          LIB  78;
GLOBAL G_TAP_USR_TST      CHAR 12;
GLOBAL G_TAP_PSW_TST      NAME 12;
LET G_TAP_TDS             CTDS;
LET G_TAP_TDS_CLASS K;
LET G_TAP_WORK_CLASS B;
LET G_TAP_GCLLIB          CTDS.GCL;
LET G_TAP_GCLMB           EXECTDS;
LET G_TAP_SM              TPR;
LET G_TAP_SM1             CTDS.SMLIB1;
LET G_TAP_SM2             CTDS.SMLIB2;
LET G_TAP_USR_TST         CTDS1;
LET G_TAP_PSW_TST         T1;
SEND #CAT('LIB SM1 : ',%G_TAP_SM1);
SEND #CAT('LIB SM2 : ',%G_TAP_SM2);
MWINLIB SL CTDS.COBOL;
MPRTLIB .SLLIB;
MWLIB CU .CULIB;
```

❑

C. Environment Generation Using GCL

The STANDARD_ENVT job is delivered with the system and is available only to the SYSADMIN project as a model. It allows the System Administrator to create the following standard environments and their associated facilities:

```
PROGRAM_PREP
BG_PROG_PREP
TDSAPPL_PREP
BG_TDS_PREP
PROJ_MANAGER
BG_PROJ_MGER
FILE_VOLUME
BG_FILE
BG_CATALOG
BG_VOLUME
SYSADMIN
BG_FILE_
BG_CATALOG_
BG_VOLUME_
```

The System Administrator should also make sure that the following system elements are installed as noted:

Environments	in SITE.CATALOG
Access rights, GCL procedure priorities	in SYS.HBINLIB
Non-system procedures	in SYS.HBINLIB
Associated Help texts	in SITE.HELP
Source command subfiles	in SYS.HSLLIB

See also the *System Installation, Configuration, and Updating Guide*.

D. System Accounting Records

This appendix gives a description of the system accounting records, discussed in Chapter 12. Each record description consists of its COBOL and GPL declaration followed by a description of the fields appearing in the declarations.

D.1 General Format

Each accounting record corresponds to an event in the life of the system. The accounting records are identified by a 2-digit record-type.

Record type range 00 to 19 is reserved for GCOS system records.

Record type range 20 to 49 is reserved for GCOS application records.

Record type range 50 to 99 is reserved for user-defined records.

The records which are generated depend on the CONFIG statement ACCOUNT, which allows users to select the records to be stored on the accounting file.

All records start with the same identification fields (described below in COBOL and GPL syntax), allowing easy processing and sorting:

Standard Header (COBOL Syntax):

```
01      HEADER-RECORD.
   02   RECORD-TYPE             PICTURE X(2).
   02   USER-NAME               PICTURE X(12).
   02   PROJECT                 PICTURE X(12).
   02   BILLING                 PICTURE X(12).
   02   JOBID                   PICTURE X(8).
   02   RON                      PICTURE X(4).
   02   REPEATED-JOB            PICTURE X(1).
   02   DSN                      PICTURE X(3).
   02   DATE-C                  PICTURE X(6).
   02   TIME-C                  PICTURE X(6).
   02   Variable information...
```

Standard Header (GPL Syntax):

```
DCL   1      RECORD,
   2   RECORD_TYPE            CHAR(2),
   2   USER_NAME              CHAR(12),
   2   PROJECT                CHAR(12),
   2   BILLING                CHAR(12),
   2   JOBID                  CHAR(8),
   2   RON                    CHAR(4),
   2   REPEATED_JOB           CHAR(1),
   2   DSN                    CHAR(3),
   2   DATE                    CHAR(6),
   2   TIME                    CHAR(6),
   2   Variable information...
```

Field Description

USER-NAME, PROJECT, BILLING

Identification given either in the $JOB statement or at log-on time. These fields are 12 characters long.

JOBID	$JOB job identification field.
RON	Run Occurrence Number. The RON is reset to zero only at Clean Restart: A RON value is unique between two clean restarts unless a job is repeated after warm restart (it is repeated with the same RON), or else the RON values have reached the maximum value (9999) and a new slice has been started (0001-9999). The RON contains leading zeros, consistent with the value returned by the $H_JOBID primitive (GPL).
REPEATED-JOB	Flag is equal to space for the first occurrence of a RON, and set to "R" if the job is repeated after warm restart. The identification Ron+Repeated-Job is unique between two clean Restarts per (0001-9999) slice. Example: "0042R"
DSN	Dynamic Step Number (0 at job level). The DSN contains leading zeros.
DATE and TIME	Date and time of the recorded information, or date and time of the beginning of the related information (job, step, etc). The DATE format is yymmdd (year, month, day). The TIME format is hhmmss (hour, minutes, seconds), and is the same for all accounting records.

The above identification fields are automatically inserted in user records as an option (see below).

A list of all accounting records and their functions is given in Table D-1.

A system crash may mean th-at some accounting information is lost, so a flag is set in each record if the information is incomplete due to a crash. The user billing program must then take this problem into account.

Table D-1. Accounting Records

RECORD TYPE	CONFIG KEY	RECORDED INFORMATION
00 Job Output	JOBOUT	Job submission and spooling
01 Job Execution	JOBEX	Job execution
02 Step Execution	STEP	Step execution
03 Multi-process Step	MPROC	Detailed process activities
05 Crash/Shutdown	END	Status at crash or shutdown, and restarts
06 System Access Violation	SVIOL	Description of attempt to violate the system protection
07 File Access Violation	FVIOL	Description of attempt to violate file protection
20 TDS Session	TDS	Data on global TDS activity
21 TDS User Activity	TDS	Accounting of each TDS user activity
23 Transaction Accounting Record	TDS	Accounting of each TDS transaction
24 TPR Accounting Record	TDS	Accounting of each TDS TPR
35 File Updating	FILEUPD	Accounting of each file updating
50-99 User Records	USER	User defined accounting records

NOTE:

The CONFIG key JOB has been retained for compatibility with release 1D. JOB is equivalent to JOBOUT and JOBEX.

The CONFIG key MPROC may only be used with STEP.

D.2 User Record Insertion in Accounting

A user program can register accounting records of its own by using the $H_PUTACT primitive in GPL or the external call to the system procedure "H_ACT_UPACNT" in COBOL or FORTRAN.

PRIMITIVE $H_PUTACT (GPL)

Syntax

```
$H_PUTACT record, length [,HEADER] ;
```

Parameters

record

i_charn, record to be written to the accounting file. It must be of the following format :

```
DCL 1 record,
    2 type CHAR (2), /*RECORD TYPE*/
    2 info CHAR (n); /*ACCOUNTING INFORMATION*/
```

The record type is from 50-99

length

i_fb (15), length n of the accounting information (info field) in bytes. The length of this information cannot exceed 1024 characters.

HEADER

specifies that the standard header option is requested. The default is no header.

Return Codes

DONE	Operation completed successfully.
RECSZERR	INFO area length exceeds 1024 characters.
RECFERR	record type is not within the range 50 to 99.

This primitive is further described in Volume 1 the *GPL System Primitives Reference Manual*.

COBOL Call:

DATA DESCRIPTION Statements

```
01 USER-RECORD.
   02   RECORD-TYPE PICTURE X(2).
   02   INFO PICTURE X(n).

77 RECORD-LENGTH      USAGE IS COMP-1.
77 HEADER-OPTION      PICTURE X.
```

CALL Statement

```
CALL "H_ACT_UPACNT" USING
HEADER-OPTION,USER-RECORD,RECORD-LENGTH.
```

Parameters

USER-RECORD	Input area containing the record type and the user accounting information. The record type is specified by the user program and must be in the range 50 to 99. The INFO area is to be filled by the user. Its length is given in the RECORD-LENGTH parameter.
RECORD-LENGTH	Length of INFO in bytes. It cannot exceed 1024 characters.
HEADER-OPTION	Specifies whether the standard header option is required. When this option is requested (HEADER-OPTION = 1) a standard header is inserted in the user record (its length must not be included in the length given as a parameter, which applies only to the user-supplied information).

User records inserted into the accounting file as described above appear as described below when read from this file by the user billing programs.

COBOL Declaration of a User Record With No Standard Header

```
01 USER-RECORD.
   02  RECORD-TYPE    PICTURE X(2).
   02  USER-INFO      PICTURE X(n).
```

GPL Declaration of a User Record with no Standard Header

```
DCL 1  USER_RECORD,
       2  RECORD_TYPE  CHAR(2), /*TYPE GIVEN BY THE USER*/
       2  USER_INFO    CHAR(n) ;
```

The record type is specified by the user program and must be within the range 50 to 99.

The INFO area is defined by the user (length and contents).

The length of INFO cannot exceed 1024 characters.

COBOL Declaration of a User Record With Standard Header

```
01   USER-RECORD.
   02   RECORD-TYPE          PICTURE X(2).
   02   USER-NAME            PICTURE X(12).
   02   PROJECT              PICTURE X(12).
   02   BILLING              PICTURE X(12).
   02   JOBID                PICTURE X(8).
   02   RON                   PICTURE X(4).
   02   REPEATED-JOB         PICTURE X.
   02   DSN                   PICTURE X(3).
   02   DATE                 PICTURE X(6).
   02   TIME                 PICTURE X(6).
   02   HEADER-FLAG           PICTURE X(2).
   02   USER-INFO            PICTURE X(n).
```

GPL Declaration of a User Record With Standard Header

```
DCL  1  USER_RECORD,
        2  RECORD_TYPE   CHAR(2),  /*TYPE GIVEN BY THE USER*/
        2  USER_NAME     CHAR(12),  /*IDENTIFICATION INSERTED*/
        2  PROJECT       CHAR(12),  /*BY THE SYSTEM*/
        2  BILLING       CHAR(12),
        2  JOBID         CHAR(8),
        2  RON           CHAR(4),
        2  REPEATED_JOB  CHAR(1),
        2  DSN           CHAR(3),
        2  DATE          CHAR(6),
        2  TIME          CHAR(6),
        2  HEADER_FLAG   CHAR(2),
        2  USER_INFO     CHAR(n);  /*USER SUPPLIED INFO*/
```

Length of the record: (68 + USER_INFO length) bytes

NOTE:

DATE and TIME are those of the record insertion in the accounting file.

HEADER-FLAG is a two-character field which delimits the end of the standard header and is used by the EDITACT utility and other accounting procedures to detect whether a standard header has been inserted.

The value of HEADER-FLAG is "!!" (hexadecimal "5A5A").

D.3 System Records Description

When all the outputs of a job are completed, a record is added to the accounting file.

D.3.1 JOBOUT Record

COBOL Declaration

```
01      JOBOUT-RECORD.
   02   RECORD-TYPE           PICTURE X(2).
   02   USER-NAME             PICTURE X(12).
   02   PROJECT               PICTURE X(12).
   02   BILLING               PICTURE X(12).
   02   JOBID                 PICTURE X(8).
   02   RON                    PICTURE X(4).
   02   REPEATED-JOB          PICTURE X(1).
   02   DSN                    PICTURE X(3).
   02   DATE-IN               PICTURE X(6).
   02   TIME-IN               PICTURE X(6).
   02   DATE-OUT              PICTURE X(6).
   02   TIME-OUT              PICTURE X(6).
   02   CLASS                 PICTURE X(2).
   02   FILLER                PICTURE X(2).
   02   SUBMITTOR-CLASS       PICTURE X(2).
   02   SUBMITTOR-NAME        PICTURE X(12).
   02   SUBMITTOR-PROJECT     PICTURE X(12).
   02   SUBMITTOR-BILLING     PICTURE X(12).
   02   SUBMITTOR-RON         PICTURE X(4).
   02   SUBMITTOR-DSN         PICTURE X(3).
   02   STATION-ID            PICTURE X(8).
   02   LINES                 USAGE IS COMP-2.
   02   PAGES                 USAGE IS COMP-2.
   02   CARDP                 USAGE IS COMP-2.
   02   CARDR                 USAGE IS COMP-2.
   02   CPU-WRITER            USAGE IS COMP-2.
   02   CPU-READER            USAGE IS COMP-2.
   02   MAG-RD                USAGE IS COMP-2.
   02   REM-RD                USAGE IS COMP-2.
   02   MAG-WR                USAGE IS COMP-2.
   02   REM-LINES             USAGE IS COMP-2.
   02   REM-PAGES             USAGE IS COMP-2.
   02   REM-CARDP             USAGE IS COMP-2.
   02   FILLER                PICTURE X(4).
```

Total record length: 187 bytes.

GPL Declaration

```
DCL 1      JOBOUT_RECORD
      2    RECORD_TYPE          CHAR(2),  /*"00"*/
      2    USER_NAME            CHAR(12),
      2    PROJECT              CHAR(12),
      2    BILLING              CHAR(12),
      2    JOBID                CHAR(8),
      2    RON                  CHAR(4),
      2    REPEATED_JOB         CHAR(1),
      2    DSN                  CHAR(3),
      2    DATE_IN              CHAR(6),
      2    TIME_IN              CHAR(6),
      2    DATE_OUT             CHAR(6),
      2    TIME_OUT             CHAR(6),
      2    CLASS                CHAR(2),
      2    FILLER               CHAR(2),
      2    SUBMITTOR_CLASS      CHAR(2),
      2    SUBMITTOR_NAME       CHAR(12),
      2    SUBMITTOR_PROJECT    CHAR(12),
      2    SUBMITTOR_BILLING    CHAR(12),
      2    SUBMITTOR_RON        CHAR(4),
      2    SUBMITTOR_DSN        CHAR(3),
      2    STATION_ID           CHAR(8),
      2    LINES                FIXED BIN (31),
      2    PAGES                FIXED BIN (31),
      2    CARDP                FIXED BIN (31),
      2    CARDR                FIXED BIN (31),
      2    CPU_WRITER           FIXED BIN (31),
      2    CPU_READER           FIXED BIN (31),
      2    MAG-RD               FIXED BIN (31),
      2    REM_RD               FIXED BIN (31),
      2    MAG_WR               FIXED BIN (31),
      2    REM_LINES            FIXED BIN (31),
      2    REM_PAGES            FIXED BIN (31),
      2    REM_CARDP            FIXED BIN (31),
      2    RFU                  CHAR (4);
```

Total record length: 187 bytes.

Description of the Fields in the JOBOUT Record

RECORD-TYPE	Type of the record ("00")
USER-NAME	User name declared for the job
PROJECT	Project identification
BILLING	Billing identification
JOBID	Job identification
RON	Run occurrence number - leading zeros
REPEATED-JOB	Flag equal to space except when the job is repeated after Warm Restart: "R"
DSN	"000"
DATE-IN	Date of input of the job into the system. DATE format is YYMMDD (year, month, day)
TIME-IN	Time of input of the job into the system. TIME format is HHMMSS (hour, minutes, seconds)
DATE-OUT	Date of deletion of the job from the system after completion of the last output
TIME-OUT	Time of deletion of the job
CLASS	Class of the job
SUBMITTOR-CLASS	Class of the job which spawned the current job
SUBMITTOR-NAME	User name of the job or of the terminal user which spawned the current job
SUBMITTOR-PROJECT	Project identification of the submitter
SUBMITTOR-BILLING	Billing identification of the submitter
SUBMITTOR-RON	Run occurrence number of the submitter
SUBMITTOR-DSN	Dynamic step number of the submitter
STATION-ID	Station identification of the job
LINES	Number of lines printed by the job (directly or through the output writer)
PAGES	Number of pages printed by the job (directly or through the output writer)

CARDP	Number of cards punched by the job (directly or through the output writer)
CARDR	Number of cards read by the job (directly or through the input reader)
CPU-WRITER	CPU time used by output writers for this job (unit = 1/1000 minute)
CPU-READER	CPU time used by input readers for this job (unit = 1/1000 minute)
MAG-RD	Number of records read by the input reader at job input time, from sequential input files specified in SIR, EJR, RUN or $SWINPUT statements, or from diskette files (on local diskette devices)
REM-RD	Number of records read by the input reader at job input time, from remote files (SIR command at an RBF station) or any console through ALTER_INPUT
MAG-WR	Number of lines saved on a magnetic file for this output (through the magnetic writer).
REM-LINES	Number of lines sent for remote printing (remote writer working for an RBF station)
REM-PAGES	Number of pages sent for remote printing (remote writer working for an RBF station)
REMCARDP	Reserved Word

Remarks on the Accounting of Writers

- The field CPU-WRITER gives the CPU time usage of all writers'activities which may be accounted to the current job: local, remote or magnetic writers.
- When the magnetic writer feature is used, only the CPU time of the magnetic writer is accounted to the job which generated the output.
- When the saved output is actually printed, the CPU time, lines and pages printed are accounted to the job which submitted the corresponding WRITER statement.
- The submitter fields are filled with blanks if the job is started by a system service job.

D.3.2 JOBEX Record

At job execution termination, a record is added to the accounting file.

COBOL Declaration

```
01      JOBEX-RECORD.
   02   RECORD-TYPE          PICTURE X(2).
   02   USER-NAME            PICTURE X(12).
   02   PROJECT              PICTURE X(12).
   02   BILLING              PICTURE X(12).
   02   JOBID                PICTURE X(8).
   02   RON                   PICTURE X(4).
   02   REPEATED-JOB         PICTURE X(1).
   02   DSN                   PICTURE X(3).
   02   DATE-START           PICTURE X(6).
   02   TIME-START           PICTURE X(6).
   02   DATE-STOP            PICTURE X(6).
   02   TIME-STOP            PICTURE X(6).
   02   CLASS                PICTURE X(2).
   02   PRIORITY             PICTURE X(1).
   02   CRASHFLAG            PICTURE X(1).
   02   FILLER               PICTURE X(1).
   02   CPU                   USAGE IS COMP-2.
   02   ELAPSE               USAGE IS COMP-2.
   02   STATUS-X             USAGE IS COMP-2.
   02   MACHINE-ID           PICTURE X(21).
   02   FILLER               PICTURE X(4).
```

Total record length: 120 bytes.

GPL Declaration

```
DCL  1    JOBEX_RECORD,
     2    RECORD_TYPE          CHAR(2),  /*"01"*/
     2    USER_NAME            CHAR(12),
     2    PROJECT              CHAR(12),
     2    BILLING              CHAR(12),
     2    JOBID                CHAR(8),
     2    RON                  CHAR(4),
     2    REPEATED_JOB         CHAR(1),
     2    DSN                  CHAR(3),
     2    DATE_START           CHAR(6),
     2    TIME_START           CHAR(6),
     2    DATE_STOP            CHAR(6),
     2    TIME_STOP            CHAR(6),
     2    CLASS                CHAR(2),
     2    PRIORITY             CHAR(1),
     2    CRASHFLAG            CHAR(1),
     2    FILLER               CHAR(1),
     2    CPU                  FIXED BIN (31),
     2    ELAPSE               FIXED BIN (31),
     2    STATUS               FIXED BIN (31),
     2    MACHINE_ID           CHAR(21),
     2    RFU                  CHAR(4);
```

Total record length: 120 bytes.

Description of the Fields in the JOBEX Record

RECORD-TYPE	Type of the record ("01")
USER-NAME	User name declared for the job (or log-on id)
PROJECT	Project identification
BILLING	Billing identification
JOBID	Job identification
RON	Run occurrence number - leading zeros
REPEATED-JOB	Same as for JOBOUT record
DSN	"000"
DATE-START	Date of effective job execution start
TIME-START	Time of job start
DATE-STOP	Date of effective job execution end
TIME-STOP	Time of job execution end
CLASS	Class of the job
PRIORITY	Scheduling priority of the job
CRASHFLAG	Flag equal to space except when the job was terminated by a system crash: "C"
CPU	CPU used by the job execution (unit: 1/1000 minute)
ELAPSE	Elapsed time of the job execution (unit: 1/1000 minute). After 24 days this field is set to 0
STATUS	Step termination status of the last executed step of the job
MACHINE-ID	Name of the CPU on which the job has been executed - as it appears on the job occurrence report.

NOTE:

If the job has been interrupted by a system crash, the CPU time of the current step at crash time is not accounted, except if the step uses the checkpoint mechanism. In this case, the job's CPU time includes the step's CPU time recorded at the last checkpoint before the crash.

D.3.3 STEP Record

At step termination time, a variable length record is added to the accounting file.

COBOL Declaration

```
01      STEP RECORD.
   02   RECORD-TYPE          PICTURE X(2).
   02   USER-NAME            PICTURE X(12).
   02   PROJECT              PICTURE X(12).
   02   BILLING              PICTURE X(12).
   02   JOBID                PICTURE X(8).
   02   RON                   PICTURE X(4).
   02   REPEATED-JOB         PICTURE X(1).
   02   DSN                   PICTURE X(3).
   02   DATE-START           PICTURE X(6).
   02   TIME-START           PICTURE X(6).
   02   DATE-STOP            PICTURE X(6).
   02   TIME-STOP            PICTURE X(6).
   02   CLASS                PICTURE X(2).
   02   XPRIORITY            PICTURE X(1).
   02   CRASHFLAG            PICTURE X(1).
   02   REPEATED-STEP        PICTURE X(1).
   02   CPU                   USAGE IS COMP-2.
   02   WAITING              USAGE IS COMP-2.
   02   READY                USAGE IS COMP-2.
   02   ELAPSE               USAGE IS COMP-2.
   02   STATUS               USAGE IS COMP-2.
   02   SYSOUT-WRITE         USAGE IS COMP-2.
   02   SYSOUT-PUNCH         USAGE IS COMP-2.
   02   MAXIMUM-IWS           USAGE IS COMP-2.
   02   BUFFER-SIZE           USAGE IS COMP-2.
   02   BACKST               USAGE IS COMP-2.
   02   PGMISSGNB            USAGE IS COMP-2.
   02   SYSMISSGNB           USAGE IS COMP-2.
   02   STACKOV              USAGE IS COMP-2.
   02   SSN                   PICTURE X(3).
   02   PROC-NUMBER           USAGE IS COMP-1.
   02   CHKPT-NB             USAGE IS COMP-1.
   02   CHKPT-MAXSIZE        USAGE IS COMP-2.
   02   TEMPORARY-SIZE       USAGE IS COMP-2.
   02   LM-NAME              PICTURE X(32).
   02   NB-OF-ENTRIES        USAGE IS COMP-1.
   02   ENTRY OCCURS 50 TIMES.
     03 IFN                    PICTURE X(8).
     03 MEDIA                PICTURE X(6).
     03 NB-OF-LOGEVENTS      USAGE IS COMP-1.
     03 NB-OF-CONNECTS       USAGE IS COMP-2.
     03 DEV-TYPE             PICTURE X(2).
     03 FILLER               USAGE IS COMP-1.
```

Length of the record: 184 + NB-OF-ENTRIES * 24.

There is one entry within ENTRY array per ifn (internal file name) assigned by the step and at most 50 assigned ifns per step record, the program must handle the variable number of entries. See GPL declaration for rules.

GPL Declaration

```
DCL 1      STEP_RECORD,
      2    RECORD_TYPE          CHAR(2),  /* ="02"*/
      2    USER_NAME            CHAR(12),
      2    PROJECT              CHAR(12),
      2    BILLING              CHAR(12),
      2    JOBID                CHAR(8),
      2    RON                  CHAR(4),
      2    REPEATED_JOB         CHAR(1),
      2    DSN                  CHAR(3),
      2    DATE_START           CHAR(6),
      2    TIME_START           CHAR(6),
      2    DATE_STOP            CHAR(6),
      2    TIME_STOP            CHAR(6),
      2    CLASS                CHAR(2),
      2    XPRIORITY            CHAR(1),
      2    CRASHFLAG            CHAR(1),
      2    REPEATED_STEP        CHAR(1),
      2    CPU                  FIXED BIN (31),
      2    WAITING              FIXED BIN (31),
      2    READY                FIXED BIN (31),
      2    ELAPSE               FIXED BIN (31),
      2    TERM_STATUS          FIXED BIN (31),
      2    SYSOUT_WRITE         FIXED BIN (31),
      2    SYSOUT_PUNCH         FIXED BIN (31),
      2    MAXIMUM_IWS          FIXED BIN (31),
      2    BUFFER_SIZE          FIXED BIN (31),
      2    BACKST               FIXED BIN (31),
      2    PGMISSGNB            FIXED BIN (31),
      2    SYSMISSGNB           FIXED BIN (31),
      2    STACKOV              FIXED BIN (31),
      2    SSN                  CHAR(3),
      2    PROC_NUMBER          FIXED BIN (15),
      2    CHKPT_NB             FIXED BIN (15),
      2    CHKPT_MAXSIZE        FIXED BIN (31),
      2    TEMPORARY_SIZE       FIXED BIN (31),
      2    LM_NAME              CHAR(32),
      2    NB_OF_ENTRIES        FIXED BIN (15),
      2    ENTRY                (NB_OF_ENTRIES),
        3  IFN                  CHAR(8),
        3  MEDIA                CHAR(6),
        3  NB_OF_LOGEVENTS      FIXED BIN (15),
        3  NB_OF_CONNECTS       FIXED BIN (31),
        3  DEV_TYPE             CHAR (2),
        3  RFU                  BIT (16) ;
```

Length of the record: 184 + NB_OF_ENTRIES * 24

There is one entry within the ENTRY array per ifn (internal file name), assigned by the step, and at most 50 assigned ifns per step record.

If more than 50 ifns were assigned to the step, as many step records as necessary are put in the file, each one containing 50 ifns (NB_OF_ENTRIES = 50), except for the last. For example, 103 ifns would be split into 2 records with 50 ifns each and 1 record with 3 ifns. For each of these records, all other information is the same.

Description of the Fields in the STEP Record

RECORD-TYPE	Type of record ("02")
USER-NAME	User name declared for the job (or logon id)
PROJECT	Project identification
BILLING	Billing identification
JOBID	Job identification
RON	Run occurrence number - leading zeros
REPEATED-JOB	Same as for JOBOUT record
DSN	Dynamic step number - leading zeros
DATE-START	Date of step execution start
TIME-START	Time of step execution start
DATE-STOP	Date of step execution stop
TIME-STOP	Time of step execution stop
CLASS	Class of the job
XPRIORITY	Dispatching priority of the step
CRASHFLAG	Flag equal to space except when the current step has been interrupted by a system crash: "C"
REPEATED-STEP	Flag equal to space except when the current step is being repeated: "R" (the DSN is kept unchanged at step repetition)
CPU	Running time of the step: CPU time used. See note below (unit = 1/1000 minute)
WAITING	Waiting time of the step: waiting for I/O completion or semaphore resources (unit = 1/1000 minute)
READY	Ready time of the step: waiting for CPU allocation (unit = 1/1000 minute)
ELAPSE	Elapsed time (unit = 1/1000 minute)
TERM-STATUS	Step termination status
SYSOUT-WRITE	Number of lines written into a SYS.OUT file (to be printed)

SYSOUT-PUNCH	Number of cards written into a SYS.OUT file (to be punched)
MAXIMUM-IWS	Maximum instantaneous working set in bytes. Gives the maximum memory (locked or not) used by the step.
BUFFER-SIZE	Buffer size effectively used in bytes
BACKST	Size of backing store used in bytes
PGMISSGNB	Number of program missing pages
SYSMISSGNB	System missing pages due to the program
STACKOV	Number of stack overflows
SSN	Static step number (leading zeros)
PROC-NUMBER	Number of processes of the process group. If greater than one, an MPROC record may optionally be issued
CHKPT-NB	Number of calls to checkpoint
CHKPT-MAXSIZE	Checkpoint largest snapshot size for the step (unit = byte)
TEMPORARY-SIZE	Size of temporary file space used by the step (unit = cylinder)
LM-NAME	Load module name of the step
NB-OF-ENTRIES	Number of entries in the array ENTRY (50 max per record)
ENTRY	One entry per ifn (internal file name) assigned by the step.
IFN	Internal file name
MEDIA	Volume serial number
NB-OF-LOGEVENTS	Number of event detected on this file and logged
NB-OF-CONNECTS	Number of I/Os related to the file.
DEV-TYPE	Type of the device related to the file.

NOTE:

For a multiprocess process group, CPU time is the sum of CPU time used in all processes. WAITING and READY apply only to the main process. See MPROC record for other values.

Records of steps in execution at crash time (CRASHFLAG = C) may not be fully recovered. Information that may be lost or that may be meaningless is contained in the following fields: CPU, ELAPSED, LINES, CARDS, BUFFER-SIZE, BACKST, PGMISSGNB, SYSMISSGNB, ENTRY-ARRAY, PROC-NUMBER.

Time Information

Time information such as the CPU time, the waiting time, the ready time, and the total elapsed time of the step is counted starting from the end of the step initiation (after having allocated all the resources and loaded the program into memory) to the end of the step termination (after having deallocated all the resources).

All the time spent within the user step space is counted, but the time spent for the step within centralized tasks of the system is not counted.

The difference between the job elapsed time and the sum of the different step elapsed times gives the time spent by the job in waiting for resources and media mounting and within centralized system tasks (resource allocation and program loading).

Accounting of VMM I/Os

The fields PGMISSGNB and SYSMISSGNB reflect the count of missing pages encountered in the global execution of the step. In fact VMM performs a number of physical I/Os associated with the execution of the step, which include:

- Swapping in from backing store to main memory all pages necessary for the step to execute
- Making room for these pages by swapping other pages out from main memory to backing store.

Even if the swapped-out pages may not belong to the current step, the corresponding I/Os are synchronous to it, and are necessary so that the step can execute.

All the VMM I/Os which are executed synchronous to the step are accounted in special entries of the ENTRY array, as explained below.

- System backing store (symbolic ifn: SYSBKST). This is divided into the backing store that is loaded at RESTORE time and that which is used as temporary paging backing store.
  - Permanent Virtual Memory File backing store (symbolic ifn: SYSPVMF) This entry contains the accounting of I/Os for all the files SYS.PVMF (SYS.PVMF1,...) where the permanent virtual memory files are located.
- Permanent Paging backing store (symbolic ifn: SYSLIB). This entry contains the accounting of I/Os for all the files SYS.LIB (SYS.LIB1,...).
- Temporary Paging backing store (symbolic ifn: SYSBKST*). This entry contains the accounting of I/Os for all the files SYS.BKST (SYS.BKST1, SYS.BKST2...).
- Temporary Virtual Memory File backing store (symbolic ifn: SYSTVMF). This entry contains the accounting of I/Os for all the files SYS.TVMF (SYS.TVMF1,...) where the temporary virtual memory files are located.

Since an entry may be related to several media, the media name of this entry is set to "****".

NOTE:

For VMM I/O accounting entries, the field NB_OF_LOGEVENTS is not used.

Only the entries which are significant are present in the entry array.

The same accounting data appears in the Job Occurrence Report, with the same rules.

Accounting of File Transfers

When a remotely assigned file is used by a File Transfer Facility (FTF) step (i.e., activated through the JCL statement FILTFR or the GCL command ENTER_FILETRANS_REQ), two entries are created in the ENTRY array.

First entry:

IFN	Ifn of the file.
MEDIA	Name of the media of the file.
NB-OF-LOGEVENTS	Filler.
NB-OF-CONNECTS	Count of data bytes transferred through telecommunications during the file transfer. This count includes the protocol headers and trailers contained in the data packs, but excludes the spaces.
DEV-TYPE	This field is set to "Rbb" in order to identify such an entry (and its complementary one).

Second entry:

IFN	This field contains the name of the site of the file (8 characters).
MEDIA	Filler.
NB-OF-LOGEVENTS	Filler.
NB-OF-CONNECTS	Filler.
DEV-TYPE	Actual device type of the media of the file.

D.3.4 MPROC Record

At step termination time, a variable length record is added to the accounting file, as well as the STEP record, if the process group is multiprocess.

COBOL Declaration

```
01        MPROC-RECORD.
  02    RECORD-TYPE                PIC X(2).
  02    USER-NAME                   PIC X(12).
  02    PROJECT                     PIC X(12).
  02    BILLING                     PIC X(12).
  02    JOBID                       PIC X(8).
  02    RON                        PIC X(4).
  02    REPEATED-JOB                PIC X(1).
  02    DSN                        PIC X(3).
  02    PROC-NUMBER                USAGE IS COMP-1.
  02    PROC-ENTRY OCCURS 20 TIMES.
    03  PCPU                       USAGE IS COMP-2.
    03  PWAITING                   USAGE IS COMP-2.
    03  PREADY                     USAGE IS COMP-2.
```

Record length: 56 + PROC-NUMBER *12 bytes

The number of PROC-ENTRY elements is variable; 20 is large enough for most uses. Any variation within 20 is handled by the program.

See GPL declaration for rules.

GPL Declaration

```
DCL 1   MPROC_RECORD,
     2   RECORD_TYPE        CHAR(2),  /*TYPE="03"*/
     2   USER_NAME          CHAR(12),
     2   PROJECT            CHAR(12),
     2   BILLING            CHAR(12),
     2   JOBID              CHAR(8),
     2   RON                CHAR(4),
     2   REPEATED_JOB       CHAR(1),
     2   DSN                CHAR(3),
     2   PROC_NUMBER        FIXED BIN (15),
     2   PROC_ENTRY         (PROC_NUMBER),
       3 PCPU               FIXED BIN (31),
       3 PWAITING           FIXED BIN (31),
       3 PREADY             FIXED BIN (31) ;
```

Length of the record: 56 + PROC_NUMBER *12 bytes.

There is one entry within the PROC_ENTRY array per physical process of the process group.

If more than 20 processes exist, as many MPROC records as necessary are put in the file, each one containing 20 process descriptions, and the last one containing the remainder (less than or equal to 20).

Description of the Fields in the MPROC Record

RECORD-TYPE	Type of the record ("03")
USER-NAME	User name declared for the job (or logon id)
PROJECT	Project identification
BILLING	Billing identification
JOBID	Job identification
REPEATED-JOB	Same as for JOBOUT record
DSN	Dynamic step number (leading zeros)
PROC-NUMBER	Number of processes described in the array PROC-ENTRY (20 maximum per record)
PROC-ENTRY	One entry per physical process of the process group (up to 20 per record)
PCPU	CPU time used by the process (unit = 1/1000 minute)
PWAITING	WAITING time of the process (unit = 1/1000 minute)
PREADY	READY time of the process (unit = 1/1000 minute)

NOTE:

The physical processes numbered p = 0 to p = pmax appear in this order in the MPROC records.

If the step was interrupted by system crash (CRASHFLAG = "C" in the corresponding step record), no MPROC record can be issued.

D.3.5 CRASH Record

After a crash or a shutdown the warm restart adds a system record to the accounting file.

COBOL Declaration

```
01          CRSH-RECORD.
    02      RECORD-TYPE                PICTURE X(2).
    02      DATE-STOP                  PICTURE X(6).
    02      TIME-STOP                  PICTURE X(6).
    02      DATE-START                 PICTURE X(6).
    02      TIME-START                 PICTURE X(6).
    02      TYPE-STOP                  PICTURE X.
    02      TYPE-START                 PICTURE X.
    02      JOB-LIST-NUMBER            USAGE IS COMP-1.
    02      JOB-LIST OCCURS 20 TIMES.
       03   USER-NAME                  PICTURE X(12).
       03   PROJECT                    PICTURE X(12).
       03   BILLING                    PICTURE X(12).
       03   JOBID                      PICTURE X(8).
       03   RON                        PICTURE X(4).
       03   FILLER                     PICTURE X.
       03   STAGE                      PICTURE X.
       03   SSN                        PICTURE X(3).
       03   DSN                        PICTURE X(3).
       03   STATUS                     USAGE IS COMP-1.
       03   CLASS                      PICTURE X(2).
```

Length of the record: 30 + 60 *JOB-LIST-NUMBER bytes

GPL Declaration

```
DCL 1      CRSH_RECORD,
      2    RECORD_TYPE          CHAR(2),  /*TYPE="05"*/
      2    DATE_STOP            CHAR(6),
      2    TIME_STOP            CHAR(6),
      2    DATE_START           CHAR(6),
      2    TIME_START           CHAR(6),
      2    TYPE_STOP            CHAR(1),
      2    TYPE_START           CHAR(1),
      2    JOB_LIST_NUMBER      FIXED BIN (15),
      2    JOB_LIST (20),
        3  USER_NAME            CHAR(12),
        3  PROJECT              CHAR(12),
        3  BILLING              CHAR(12),
        3  JOBID                CHAR(8),
        3  RON                  CHAR(4),
        3  RFU                  CHAR(1),
        3  STAGE                CHAR(1),
        3  SSN                  CHAR(3),
        3  DSN                  CHAR(3),
        3  STATUS               FIXED BIN (15),
        3  CLASS                CHAR (2);
```

Length of record: 30 + 60* JOB_LIST_NUMBER bytes. There is one entry within the JOB_LIST array per job in execution or suspended (EX or SUSP states). If there are more than 20 jobs, another record is created, so several crash records may be related to the same system interruption.

Description of the Fields in the CRASH Record

RECORD-TYPE	Type of the record ("05")
DATE-STOP	Date of system interruption
TIME-STOP	Time of system interruption
DATE-START	Date of system restart
TIME-START	Time of system restart
TYPE-STOP	Type of system interruption
TYPE-START	Type of system restart
JOB-LIST-NUMBER	Number of entries in the JOB-LIST array (a maximum of 20 per record)

JOB-LIST	One entry per job in execution at the time of system interrupt (executing or suspended)
USER-NAME	User name declared for the job
PROJECT	Project identification
BILLING	Billing identification
JOBID	Job identification
RON	Run occurrence number (leading zeros)
CLASS	Class of the job
STAGE	S if the job was suspended E if the job was in execution
SSN	Static step number (number assigned to the step at JCL translation time) of the step that was in execution (leading zeros)
DSN	Dynamic step number (number assigned to the step at execution time) of the step that was in execution (leading zeros)
STATUS	Step termination status of the step that was in execution.

Table D-2 gives the crash and shutdown information.

Table D-2. Crash/Shutdown Information

Type of Interruption			
Type of Restart	Shutdown TYPE-STOP = S	Crash TYPE-STOP = C	Unknown TYPE-STOP = U
Warm	One record is issued at shutdown time with: TYPE-START = U JOB-LIST given One record is issued at restart time with: TYPE-START = W No JOB-LIST given	One record is issued at restart time with: TYPE-START = U JOB-LIST given	
Cold	One record is issued at shutdown time with: TYPE-START = U JOB-LIST given One record is issued at restart time with: TYPE-START = C No JOB-LIST given	One record is issued at restart time with: TYPE-START = C No JOB-LIST given	
Clean	One record is issued at restart time with: TYPE-START = 0 No JOB-LIST given		

TYPE-STOP = S	Shutdown
C	Crash
U	Unknown (e.g., power off). DATE-STOP and TIME-STOP are set to 000000 in this case.
TYPE-START = W	Warm restart
C	Cold restart
0	Clean restart
U	Unknown (only in the record issued at shutdown time). DATE-START and TIME-START are set to 000000 in this case.

NOTE:

"JOB-LIST given" means that JOB-LIST-NUMBER may be other than zero, and that JOB-LIST-NUMBER = 0 reflects an empty machine at interruption time.

"No JOB-LIST given" means that JOB-LIST-NUMBER is equal to zero whatever the machine load may be at interruption time.

The type of interruption is "Unknown" each time the failure or action which interrupts the system does not enable the availability procedures to save the interruption information (for instance power failure, reload of the system without restart attention, etc).

D.3.6 SVIOL Record

Each time an attempt is made to access system data in violation of the protection afforded by the catalog mechanism, a record is added to the accounting file.

COBOL Declaration

```
01       SVIOL-RECORD.
    02   RECORD-TYPE         PICTURE X(2).
    02   USER-NAME           PICTURE X(12).
    02   PROJECT             PICTURE X(12).
    02   BILLING             PICTURE X(12).
    02   JOBID               PICTURE X(8).
    02   RON                 PICTURE X(4).
    02   REPEATED-JOB        PICTURE X(1).
    02   DSN                 PICTURE X(3).
    02   DATE-EVENT          PICTURE X(6).
    02   TIME-EVENT          PICTURE X(6).
    02   IA-REASON           PICTURE X(10).
    02   IA-USER             PICTURE X(12).
    02   IA-PROJECT          PICTURE X(12).
    02   IA-BILLING          PICTURE X(12).
    02   IA-TERMINAL         PICTURE X(24).
    02   IA-PASSWORD         PICTURE X(12).
    02   IA-APPLICATION      PICTURE X(12).
    02   IA-STATION          PICTURE X(8).
    02   IA-JOBID            PICTURE X(8).
```

Length of the record: 176 bytes

GPL Declaration

```
DCL 1    SVIOL_RECORD,
      2  RECORD_TYPE        CHAR(2),  /*TYPE="06"*/
      2  USER_NAME          CHAR(12),
      2  PROJECT            CHAR(12),
      2  BILLING            CHAR(12),
      2  JOBID              CHAR(8),
      2  RON                CHAR(4),
      2  REPEATED_JOB       CHAR(1),
      2  DSN                CHAR(3),
      2  DATE_EVENT         CHAR(6),
      2  TIME_EVENT         CHAR(6),
      2  IA_REASON          CHAR(10),
      2  IA_USER            CHAR(12),
      2  IA_PROJECT         CHAR(12),
      2  IA_BILLING         CHAR(12),
      2  IA_TERMINAL        CHAR(24),
      2  IA_PASSWORD        CHAR(12),
      2  IA_APPLICATION     CHAR(12),
      2  IA_STATION         CHAR(8),
      2  IA_JOBID           CHAR(8) ;
```

Length of the record: 176 bytes

Note that "IA" stands for Invalid Access (data)

Description of the Fields in the SVIOL Record

RECORD-TYPE	Type of the record ("6")
USER-NAME	User identification of the component which discovered the violation attempt. This component may be either the system itself or a Service job. In all cases, USER-NAME, PROJECT, and BILLING are those declared when the operator logs on. (The default values are OPERATOR, OPERATOR, and INSTALL respectively). The only exception to this is if a batch job is submitted from a user terminal, when the user's log-on information is given, since the input reader is working for him.
PROJECT	Project identification (see above)
BILLING	Billing identification (see above)

JOBID	Job identification of the component which discovered the violation attempt (SYSTEM for the system, job name for service jobs).
RON	Run occurrence number of the component which discovered the violation attempt ("0001" for the system, actual RON for service jobs).
DSN	Associated dynamic step number
REPEATED-JOB	Blank for this record
DATE-EVENT	Date of the violation attempt
TIME-EVENT	Time of the violation attempt
IA-REASON	Coded reason of the violation, the format is **xxnn.oocc** where **xxnn** is the valid key of a system message, in which **xx** is the abbreviation of a system component, (e.g., JB for JOB MANAGEMENT), and **nn** is the number of a message associated with this component (e.g., JB65). **oo** is a 2-digit object code describing the object on which an invalid access has been performed. The object codes are: 00: root 01: directory 02: file 03: file link 04: volume 05: catalog 06: user 07: project 08: billing 09: application 10: station 11: password 12: relation user/project 13: relation project/billing 14: relation project/station 15: relation project/application 16: transaction 17: site 18: main operator right

	cc is a 2-digit violation class describing the type of access violation performed. The violation classes are: *Classes 01 to 07: violation of access to cataloged objects* 01: simple access violation rights: LIST/EXECUTE/READ WRITE/RECOVERY 02: owner right violation 03: illegal setting of access rights 04: illegal deletion of access rights 05: violation of SYSADMIN access rights 06: access to a protected object when access to the system is not checked 07: volume access violation *Classes 11-13: violation of access to the system* 11: Batch access violation 12: Telecom access violation 13: Operator log-on cannot be checked because a system failure has been encountered during checking. The log-on is accepted in order to let the operation go on (otherwise the system would be deadlocked). E.g. I/O error in accessing the catalog. *Classes 21 to 23: attempt to use a protected service* 21: Use of TDS 22: Routing of output 23: Station-site mapping
IA-USER IA-PROJECT IA-BILLING	Valid or invalid information given when accessing the system. This depends on IA-REASON. The information is given either on the job card (for a batch job) or at log-on (telecom access or operator log-on).
IA-TERMINAL	Terminal identification at which the violation attempt was performed (if applicable)
IA-PASSWORD	Invalid password (if applicable)
IA-APPLICATION	Application to which an invalid access was attempted (if applicable)
IA-STATION	Station to which an invalid access (routing) was attempted (if applicable)

IA-JOBID	Identification of the job which caused the violation (if applicable)

NOTE:

The "IA" fields are filled depending on the reason for the violation.

The fields which are not applicable are filled with spaces.

D.3.7 FVIOL Record

Each time an attempt is made to violate the catalog protection of an object (file, volume), a record is added to the accounting file.

COBOL Declaration

```
01        FVIOL-RECORD.
     02   RECORD-TYPE          PICTURE X(2).
     02   USER-NAME            PICTURE X(12).
     02   PROJECT              PICTURE X(12).
     02   BILLING              PICTURE X(12).
     02   JOBID                PICTURE X(8).
     02   RON                  PICTURE X(4).
     02   REPEATED-JOB         PICTURE X(1).
     02   DSN                  PICTURE X(3).
     02   DATE-EVENT           PICTURE X(6).
     02   TIME-EVENT           PICTURE X(6).
     02   IA-REASON            PICTURE X(10).
     02   IA-CATALOG           PICTURE X(44).
     02   IA-MEDIA             PICTURE X(6).
     02   IA-DEVTYPE           PICTURE X(2).
     02   FILLER               PICTURE X(2).
     02   IA-EFN               PICTURE X(44).
```

Length of the record: 174 bytes

GPL Declaration

```
DCL 1    FVIOL_RECORD,
      2  RECORD_TYPE        CHAR(2),  /*TYPE="07"*/
      2  USER_NAME          CHAR(12),
      2  PROJECT            CHAR(12),
      2  BILLING            CHAR(12),
      2  JOBID              CHAR(8),
      2  RON                CHAR(4),
      2  REPEATED JOB       CHAR(1),
      2  DSN                CHAR(3),
      2  DATE_EVENT         CHAR(6),
      2  TIME_EVENT         CHAR(6),
      2  IA_REASON          CHAR(10),
      2  IA_CATALOG         CHAR(44),
      2  IA_MEDIA           CHAR(6),
      2  IA_DEVTYPE         CHAR(2),
      2  FILLER             CHAR(2),
      2  IA_EFN             CHAR(44);
```

Length of the record: 174 bytes

Note that "IA" stands for Invalid Access (data)

Description of the Fields in the FVIOL Record

RECORD-TYPE	Type of the record ("07")
USER-NAME	User identification of the job (or terminal session) which attempted to violate the protection set on an object (file or volume)
PROJECT	Related project identification
BILLING	Related billing identification
JOBID	Job identification of the job which attempted the violation
RON	Related run occurrence number
DSN	Dynamic step number of the current step at violation time.
REPEATED-JOB	Same as for JOBOUT record
DATE-EVENT	Date of the violation attempt

TIME-EVENT	Time of the violation attempt
IA-REASON	Coded reason of the invalid access to the protected object, the format of which is xxnn.ooccb. See SVIOL for explanation.
IA-CATALOG	Identification of the private catalog which protected the object (if applicable)
IA-MEDIA	Media name of the protected object
IA-DEVTYPE	Device type of the protected object
IA-EFN	External file name (if the protected object is a file)

D.3.8 FILEUPDATE Record

A FILEUPDATE record is sent when the last permanent file opened with a processing mode that is not INPUT is deassigned during the execution of a step. The file can be cataloged or not.

COBOL Declaration

```
01        FILEUPDATE-RECORD.
     02   RECORD-TYPE          PICTURE X(2).
     02   USER-NAME            PICTURE X(12).
     02   PROJECT              PICTURE X(12).
     02   BILLING              PICTURE X(12).
     02   JOBID                PICTURE X(8).
     02   RON                  PICTURE X(4).
     02   REPEATED-JOB         PICTURE X(1).
     02   DSN                  PICTURE X(3).
     02   DATE                 PICTURE X(6).
     02   TIME                 PICTURE X(6).
     02   SEPARATOR            PICTURE X(2).
     02   EFN                  PICTURE X(44).
     02   DVTYP                PICTURE X(2).
     02   NUMBVOL              USAGE IS COMP-1.
     02   VSN                  OCCURS 10 PICTURE X (6).
```

Length of the record: 176 bytes

GPL Declaration

```
DCL 1    FILEUPDATE_RECORD,
      2  RECORD_TYPE          CHAR(2),  /*TYPE="35"*/
      2  USER_NAME            CHAR(12),
      2  PROJECT              CHAR(12),
      2  BILLING              CHAR(12),
      2  JOBID                CHAR(8),
      2  RON                  CHAR(4),
      2  REPEATED JOB         CHAR(1),
      2  DSN                  CHAR(3),
      2  DATE                 CHAR(6),
      2  TIME                 CHAR(6),
      2  SEPARATOR            CHAR(2),
      2  EFN                  CHAR(44),
      2  DVTYP                CHAR(2),
      2  NUMBVOL              FIXED BIN (15),
      2  VSN(10)              CHAR(6);
```

Length of the record: 176 bytes

Description of the Fields in the FILEUPDATE Record

RECORD-TYPE	Type of the record ("35")
USER-NAME	User identification
PROJECT	Related project identification
BILLING	Related billing identification
JOBID	Job identification
RON	Related run occurrence number (leading zeros)
REPEATED-JOB	Same as for JOBOUT record
DSN	Dynamic step number (leading zeros)
DATE	Date of record
TIME	Time of record
SEPARATOR	Separator
EFN	External file name
DVTYP	Device type
NUMBVOL	Number of volumes in the file
VSN	Volume serial number

NOTE:

When a file is deassigned when the system is restarted following a crash, the fields REPEATED_JOB, DSN, DATE, and TIME are filled with zeros.

D.4 Application Records Description

The application record (i.e., GCOS records but not system records) types are in the range 20 to 49 and contain the standard header:

```
RECORD-TYPE
USER-NAME
PROJECT
BILLING
JOBID
RON
REPEATED-JOB
DSN
DATE
TIME
```

TDS Accounting Records

Accounting records written in the system accounting file are as follows:

- TDS session accounting record when the TDS session terminates.
- User session accounting record when a user disconnects normally (BYE) or is abnormally disconnected (CANCEL).
- Transaction accounting record at the end of a transaction. This record must be requested specifically in TDSGEN or dynamically at the master terminal.
- TPR accounting record at the end of a TPR. This record must be requested specifically in TDSGEN or dynamically at the master terminal.

Details of these records are given in the *TDS Administrator's Guide*.

E. Clean the IOF Mailbox

Purpose

The JCL command CLEANMBX is used to delete all messages of an IOF mailbox user. The user must specify the name of the user whose IOF mailbox is to be deleted.

Syntax

EJR CLEANMBX LIB=SYS.HSLLIB VALUES=(user_id)

Where:

user_id User name , **mandatory**.

Description of Options

User_id Name of the user whose IOF mailbox is to be deleted.

The user's name must be specified as positional value, up to 12 characters. **Mandatory**.

See example of use below.

Constraints

The user_id parameter is mandatory.

Only a MAIN operator or a SYSADMIN user can delete the IOF mailbox of an other user.
Other users can only delete their own mailbox.

The user whose IOF mailbox is to be deleted must not be connected to the IOF application.

Error Messages

No output will be produced except in case of fatal error with SEV 3.

The following error messages are sent to the submitter console and in the JOR.

FATAL	CLEAN IOF MAILBOX : UNKNOWN USER: DUPONT
FATAL	CLEAN IOF MAILBOX : USER PRESENTLY LOGGED
FATAL	CLEAN IOF MAILBOX : PERMISSION DENIED
FATAL	CLEAN IOF MAILBOX : ERRONEOUS USER NAME: DUPONTDENEMOURS
FATAL	CLEAN IOF MAILBOX : USER MISSING

EXAMPLES

EJR CLEANMBX LIB=SYS.HSLLIB VALUES=(DUPONT)

❑

Index

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Technical publication remarks form

Title :	DPS7000/XTA NOVASCALE 7000 GCOS7 System Administrator's Manual Operating System: Administration

Reference N° :	47 A2 54US 02	**Date :**	**June 2002**

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be promptly investigated by qualified technical personnel and action will be taken as required.
If you require a written reply, please include your complete mailing address below.

NAME : _____ Date : _____

COMPANY : _____

ADDRESS : _____

_____

Please give this technical publication remarks form to your BULL representative or mail to:

Bull - Documentation Dept.
1 Rue de Provence
BP 208
38432 ECHIROLLES CEDEX
FRANCE
info@frec.bull.fr

Technical publications ordering form

To order additional publications, please fill in a copy of this form and send it via mail to:

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

Phone: +33 (0) 2 41 73 72 66
FAX: +33 (0) 2 41 73 70 66
E-Mail: srv.Duplicopy@bull.net

CEDOC Reference #	Designation	Qty
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		
_ _ _ _ _ _ _ _ _ [_ _]		

[_ _] : The latest revision will be provided if no revision number is given.

NAME: ______________________________ Date: __________

COMPANY: ______________________________

ADDRESS: ______________________________

____

PHONE: ______________________ FAX: ______________

E-MAIL: ______________________________

For Bull Subsidiaries:

Identification: ______________________________

For Bull Affiliated Customers:

Customer Code: ______________________________

For Bull Internal Customers:

Budgetary Section: ______________________________

For Others: Please ask your Bull representative.

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

REFERENCE
47 A2 54US 02