ASG-MethodManager®
User's Guide

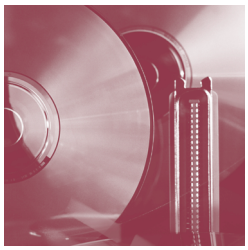
Version: 2.5

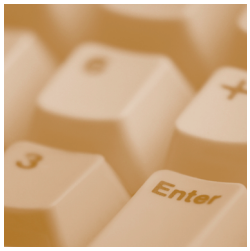
Publication Number: MMR0200-25

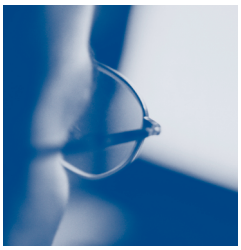
Publication Date: November 1996

The information contained herein is the confidential and proprietary information of Allen Systems Group, Inc. Unauthorized use of this information and disclosure to third parties is expressly prohibited. This technical publication may not be reproduced in whole or in part, by any means, without the express written consent of Allen Systems Group, Inc.

© 1998-2001 Allen Systems Group, Inc. All rights reserved.
All names and products contained herein are the trademarks or registered trademarks of their respective holders.







ASG Worldwide Headquarters Naples, Florida USA | asg.com
1333 Third Avenue South, Naples, Florida 34102 USA Tel: 941.435.2200 Fax: 941.263.3692 Toll Free: 1.800.932.5536

ASG Documentation/Product Enhancement Fax Form

Please FAX comments regarding ASG products and/or documentation to (941) 263-3692.

Company Name	Telephone Number	Site ID	Contact name

Product Name/Publication	Version #	Publication Date
Product:		
Publication:		
Tape VOLSER:		

Enhancement Request:

© 2001 Allen Systems Group, Inc.
All names and products are trademarks or registered trademarks of their respective holders.

ASG Support Numbers

ASG provides support throughout the world to resolve questions or problems regarding installation, operation, or use of our products. We provide all levels of support during normal business hours and emergency support during non-business hours. To expedite response time, please follow these procedures.

Please have this information ready:

- Product name, version number, and release number
- List of any fixes currently applied
- Any alphanumeric error codes or messages written precisely or displayed
- A description of the specific steps that immediately preceded the problem
- The severity code (ASG Support uses an escalated severity system to prioritize service to our clients. The severity codes and their meanings are listed below.)

If You Receive a Voice Mail Message:

1. Follow the instructions to report a production-down or critical problem.
2. Leave a detailed message including your name and phone number. A Support representative will be paged and will return your call as soon as possible.
3. Please have the information described above ready for when you are contacted by the Support representative.

Severity Codes and Expected Support Response Times

Severity	Meaning	Expected Support Response Time
1	Production down, critical situation	Within 30 minutes
2	Major component of product disabled	Within 2 hours
3	Problem with the product, but customer has work-around solution	Within 4 hours
4	"How-to" questions and enhancement requests	Within 4 hours

ASG provides software products that run in a number of third-party vendor environments. Support for all non-ASG products is the responsibility of the respective vendor. In the event a vendor discontinues support for a hardware and/or software product, ASG cannot be held responsible for problems arising from the use of that unsupported version.

Business Hours Support

Your Location	Phone	Fax	E-mail
United States and Canada	800.354.3578 1.941.435.2201 **Secondary Numbers:** 800.227.7774 800.525.7775	941.263.2883	support@asg.com
Australia	61.2.9460.0411	61.2.9460.0280	support.au@asg.com
England	44.1727.736305	44.1727.812018	support.uk@asg.com
France	33.141.028590	33.141.028589	support.fr@asg.com
Germany	49.89.45716.300	49.89.45716.400	support.de@asg.com
Singapore	65.224.3080	65.224.8516	support.sg@asg.com
All other countries:	1.941.435.2201		support@asg.com

Non-Business Hours - Emergency Support

Your Location	Phone	Your Location	Phone
United States and Canada	800.354.3578 1.941.435.2201 **Secondary Numbers:** 800.227.7774 800.525.7775 **Fax:** 941.263.2883		
Asia	011.65.224.3080	**Japan/Telecom**	0041.800.9932.5536
Australia	0011.800.9932.5536	**New Zealand**	00.800.9932.5536
Denmark	00.800.9932.5536	**South Korea**	001.800.9932.5536
France	00.800.9932.5536	**Sweden/Telia**	009.800.9932.5536
Germany	00.800.9932.5536	**Switzerland**	00.800.9932.5536
Hong Kong	001.800.9932.5536	**Thailand**	001.800.9932.5536
Ireland	00.800.9932.5536	**United Kingdom**	00.800.9932.5536
Israel/Bezeq	014.800.9932.5536		
Japan/IDC	0061.800.9932.5536	**All other countries**	1.941.435.2201

ASG Web Site

Visit http://www.asg.com, ASG's World Wide Web site.

Submit all product and documentation suggestions to ASG's product management team at http://www.asg.com/products/suggestions.asp

If you do not have access to the web, FAX your suggestions to product management at (941) 263-3692. Please include your name, company, work phone, e-mail ID, and the name of the ASG product you are using. For documentation suggestions include the publication number located on the publication's front cover.

Contents

Preface v
About this Publication v
Publication Conventions vi

1 Introduction to MethodManager 1
What is MethodManager? 1
Development Using Methodologies 2
Integrated Case Tools 2
The Repository Information Model 3
The Advantages 4

2 Modeling Systems and Information 5
What a Repository Does 5
The System Development Life Cycle 5
The Stages of the Life Cycle 6
How to Store Information in the Repository 7
Entities and Relationships 7
Modeling Techniques 7
How to Store Different Versions of a Model 7
How Incomplete Models are Stored 8
How to Enter and Alter Models 8
How to Enter Models 8
Protecting Models Against Unauthorized Access 9
Grouping Together Parts of a Model 9
Finding Out the Properties of a Model 9

3 The User Interface 11
Introduction 11
Accessing the Panel Interface 14
Leaving the Panel Interface 16
The Structure of the Panel Interface 16

4 Using the Panel Interface 21
Moving Around 21
Function Key Settings 26
Types of Panel 27
Menu Panels 27
Input Panels 28
In-context Help Panels 29

Expert Input Panels 30
Selection Panels 32
Assisted Display Panels 33
Assisted Update Panels 34
Getting Help 34

5 The Documentation Facility 37
Document Definition Contents 37
The Stages in Document Production 38
Writing Document Definitions 38
Creating and Updating Document Definitions 39
Attributes of a DOCUMENT Member 40
Types of Entry in the CONTENTS 40
Escape Characters 41
Static Text 41
Using Commands 42
Document Assembly Commands 42
Manager Products Commands 71
Formatting Commands 72
Commentary Lines 93
Document Assembly and Output 94
The DOC Command 94
The SDOC Command 95
Formatting Your Document 95
The FDOC Command 96
Other Documentation Facilities 97
The DOCUQUERY Command 97
Tailoring the Documentation Facility 97
The DCUPD Command 98

6 Summary of Commands 99
Panel Interface Commands 99
Editing Commands 99
Line Commands 100
Help Commands 102
Routing Commands 102
Primary Commands 102
Additional Note: The RELABEL Command 104
Document Commands 104
Additional Commands 104
TOAUPD 104

7 Input Panels and Help 105

8 Navigation Charts 229
How To Use the Charts 229
Lifecycle Services 231
Toolset Services 233
Tutorial 242
Strategic Information Planning 244

Index 247

Preface

This *ASG-MethodManager User's Guide* is one of a series describing the Manager family of Program Products developed by ASG for organizations seeking to automate and manage their application development and maintenance effort.

This publication introduces the concept of computer aided software engineering (CASE) tools and development life cycles. It goes on to describe how to use the panel interface, its structure and how to find your way around. ASG-MethodManager (herein called MethodManager) provides a complete application development environment, tailorable to your own requirements.

ASG welcomes your comments, as a preferred or prospective customer, on this publication or on the MethodManager product.

About this Publication

The *ASG-MethodManager User's Guide* consists of these chapters:

- Chapter 1, "Introduction to MethodManager," provides an overview of the MethodManager product.
- Chapter 2, "Modeling Systems and Information," describes the repository and life cycle management.
- Chapter 3, "The User Interface," describes the three kinds of user interface used to communicate with the repository.
- Chapter 4, "Using the Panel Interface," provides information for using the panel interface.
- Chapter 5, "The Documentation Facility," describes the documentation facility in MethodManager.
- Chapter 6, "Summary of Commands," summarizes the commands that you can use in the panel interface.
- Chapter 7, "Input Panels and Help," provides a reference to help you use input and expert panels in ToolSet SERVICES.
- Chapter 8, "Navigation Charts," provides a list of the panels that you access using the menus, to help you navigate more quickly and efficiently.

Publication Conventions

The following conventions apply to syntax diagrams that appear in this publication.

Diagrams are read from left to right along a continuous line (the "main path"). Keywords and variables appear on, above, or below the main path.

Convention	Represents
➤➤	At the beginning of a line indicates the start of a statement.
➤◄	At the end of a line indicates the end of a statement.
———➤	At the end of a line indicates that the statement continues on the line below.
➤———	At the beginning of a line indicates that the statement continues from the line above.

Keywords are in upper-case characters. Keywords and any required punctuation characters or symbols are highlighted. Permitted truncations are not indicated.

Variables are in lower-case characters.

Statement identifiers appear on the main path of the diagram:

```
➤———— COMMAND ————————————————————————————————➤
```

A required keyword appears on the main path:

```
➤———— COMMAND ——— KEYWORD ————————————————————➤
```

An optional keyword appears below the main path:

```
➤———— COMMAND ————————————————————————————————➤
              └── KEYWORD ──┘
```

Where there is a choice of required keywords, the keywords appear in a vertical list; one of them is on the main path:

```
➤———— COMMAND ———┌── KEYWORD1 ──┐————————————➤
                 ├── KEYWORD2 ──┤
                 └── KEYWORD3 ──┘
```

or

```
➤———— COMMAND ———┬── KEYWORD1 ──┐————————————➤
                 ├── KEYWORD2 ──┤
                 └── KEYWORD3 ──┘
```

Where there is a choice of optional keywords, the keywords appear in a vertical list, below the main path:

```
➤———— COMMAND ———————————————————————————————➤
                 ┌── KEYWORD1 ──┐
                 └── KEYWORD2 ──┘
```

Convention	Represents

The repeat symbol, <<<<<<, above a keyword or variable, or above a whole clause, indicates that the keyword, variable, or clause may be specified more than once:

```
                    <<<<<<<<
➤──────COMMAND ──── variable ──────────────────────────────────➤
```

A repeat symbol broken by a comma indicates that if the keyword, variable, or clause is specified more than once, a comma must separate each instance of the keyword, variable, or clause:

```
                    <<<,<<
➤──────COMMAND ──── variable ──────────────────────────────────➤
```

The repeat symbol above a list of keywords (one of which appears on the main path) indicates that any one or more of the keywords may be specified; at least one must be specified:

```
                                <<<<<<<<<<<<<<<<
➤──────COMMAND──────────┬──KEYWORD1──┬──────────────────────────➤
                        └──KEYWORD2──┘
```

The repeat symbol above a list of keywords (all of which are below the main path) indicates that any one or more of the keywords maybe specified, but they are all optional:

```
                                <<<<<<<<<<<<<<<<
➤──────COMMAND──────────┬──┬──KEYWORD1──┬──┬──────────────────────➤
                        │  └──KEYWORD2──┘  │
                        └──────────────────┘
```

Allen Systems Group, Inc. uses these conventions in publications:

Convention	Represents
ALL CAPITALS	Directory, path, file, dataset, member, database, program, command, and parameter names.
Initial Capitals on Each Word	Window, field, field group, check box, button, panel (or screen), option names, and names of keys. A plus sign (+) is inserted for key combinations (e.g., Alt+Tab).
`lowercase italic monospace`	Information that you provide according to your particular situation. For example, you would replace *`filename`* with the actual name of the file.
`Monospace`	Characters you must type exactly as they are shown. Code, JCL, file listings, or command/statement syntax. Also used for denoting brief examples in a paragraph.

1 Introduction to MethodManager

What is MethodManager?

MethodManager is a repository-based software product that enables your organization to integrate the following components into one Information Engineering (IE) environment:

- Industry standard development methodologies (such as SSADM), ASG-supplied methodologies and organization-specific working methods
- Computer Aided Software Engineering (CASE) tools provided by ASG and by other vendors
- The information your organization has stored in the repositories, catalogs, directories, libraries and/or databases supplied by ASG and other vendors.

Using MethodManager, your organization can create menu-driven dialogues that harness the power of Computer Aided Software Engineering (CASE) tools into working methodologies. ASG provides you with predefined tools and methodologies but, thanks to MethodManager's open-architecture, you can also integrate your own or those provided by other vendors.

The major components of MethodManager are:

- A panel-driven interface for methods and project management called LifeCycle SERVICES
- Development and productivity tools called ToolSet SERVICES
- A repository interface called Repository SERVICES
- A PWS Graphical Workbench
- A repository.

Once populated, the MethodManager SERVICES are a flexible, easy-to-use Information Engineering environment. The SERVICES offer a user interface to MethodManager's own functions and other vendor's tools. There are various ways of using the SERVICES, depending on your level of experience or the task you want to perform. You can choose between a graphical view thought MethodManager's Programmable Workstation Graphical Workbench (PGW), menu-driven panel interaction or command-oriented expert access.

Development Using Methodologies

LifeCycle SERVICES provide a sophisticated method-to-tool interface that enables you to harness the power of CASE tools into methodology-oriented dialogues called *Life Cycle Models*. A Life Cycle Model presents a series of panels that guide you through the steps required to complete a project according to a defined method and invoke the right tool at the right time.

Each project is scoped, so that you get a project-oriented view of the repository. This means the amount of information you have to cope with is limited, but you still maintain a corporate repository across all projects.

Thus LifeCycle SERVICES enable your organization to achieve high quality, standardized results.

ASG supplies you with predefined and tailorable Life Cycle Models for particular Information Engineering tasks: for example, ASG supplies a Life Cycle Model to guide you through a strategic information planning exercise. You can also create your own Life Cycle Models to support your own procedures or proprietary methodologies.

Refer to *ASG-MethodManager Administration* for details of LifeCycle SERVICES.

Integrated Case Tools

ToolSet SERVICES enable your organization to maximize productivity and individual output by providing access to CASE tools. ASG supplies the following tools:

- Flexible and extensive query, analysis, and documentation facilities for all entries in the repository
- A data modeling and design tool. This enables sophisticated data analysis and design, supporting top-down entity relationship modeling through to normalized database structures for DB2 and IMS
- Import and export tools for various platforms, especially System Application Architecture (SAA) compliant ones such as COBOL structures.

These tools can be accessed and used via a Life Cycle Model, as described above, or directly via ToolSet SERVICES. In both cases you access the tools and functions using interactive panels. The panels can be tailored so that you can add support for CASE tools supplied to you by other vendors.

The Repository Information Model

All the tools ASG provides operate on information stored in the repository. The structure of information held in the repository is defined in the MethodManager Dictionary/Repository Information Model (MDRIM). The MDRIM includes descriptions of:

- Information Engineering models, stored by the administrator
- Entities and relationships defined according to those models, stored by users.

The MDRIM also supports the definition of the major database systems (DB2 and IMS for example) as well as generic data, processes, business and methodology related objects.

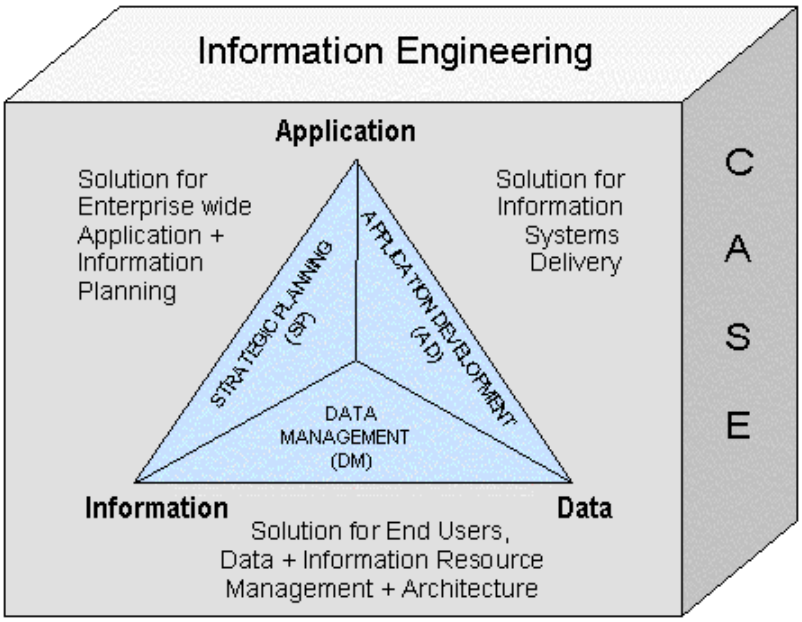


Figure 1. Comprehensive Information Engineering Solutions

MethodManager comes complete with a ready-to-use Information Engineering model. This information model is tailorable so you can build repository information models consisting of your own entity types, attribute types and relationship types.

Repository SERVICES provide the services necessary to create, maintain and use the contents of the repository. You access Repository SERVICES via the Repository Functions branch of ToolSet SERVICES.

The Advantages

The advantages of using MethodManager are that it:

- Is comprehensive as it supports all the aspects of Information Engineering, that is:
  - information planning
  - application development
  - data management.
- Is tailorable as it helps to solve your Information Engineering problems with predefined facilities but it is easily tailored to suit your corporate culture, organization and standards
- Has open architecture so that the CASE tools and methodologies supplied by other vendors can be incorporated into the MethodManager environment. You are not locked into any one methodology, repository or set of CASE tools, but can exploit the best and most suitable of those available
- Is profitable because it combines productivity tools with an easy-to-use methodology interface. MethodManager provides a degree of automation, enforcement and validation that enables your organization to get return on investment from its development activities.

MethodManager delivers solutions today and it provides a framework in which to harness the methodology and productivity tools of the future.

2 Modeling Systems and Information

What a Repository Does

A repository stores models of the data, processes and design information within a company. The models are composed of meta-data (data about data) which can represent the structure of databases, program data, departments, or any system that uses recordable data, even card filing systems.

By centralizing the models, the business and data processing activities of the company are simplified. Information can be shared across departments, and the effects of a corporate change can be analyzed.

The repository can be used simply as a database, but its real power lies in its ability to engineer new systems from models of existing systems. This is known as re-engineering. Such systems may be programs, databases, or even corporate methodologies.

The repository is the heart of a healthy information machine. Should the repository be neglected or poorly maintained, the limbs of the machine will be unable to perform essential tasks effectively.

The starting point for any repository work is the storage of a model. This model is derived from an analysis of the system of interest, such as a database that stores standards for program data structures.

The components of a model are stored in the repository as *members*. Members comprise keywords and clauses that represent attributes of the real world.

The System Development Life Cycle

The system development life cycle is a rigorous method by which new systems can be developed, and existing systems modified. The full power of the repository comes to the fore with life cycle management. Through all the stages of the life cycle, from planning and analysis to design and construction, the models you enter provide the driving force for turning the wheels of the systems project.

There are three different interfaces that you can use to enter models into the repository:

- The graphical interface
- The panel interface
- The command interface.

Refer to Chapter 3, "The User Interface," on page 11 for details.

When you use the graphical interface to enter models, the resulting diagrams can be readily shown to, and validated by, those involved in the project. Should you prefer to use the command or panel interface to enter the model directly into the repository, you can still transfer the members to the graphical interface in order to show the model diagrammatically. Conversely, you can transfer diagrams from the local repository of the graphical interface to the host repository, for inclusion in the life cycle.

The Stages of the Life Cycle

Each stage of the life cycle has a set of models that best represent the relevant information. The graphical interface provides you with several types of diagram that allow you to draw each of these models. The command interface and the panel interface provide you with an assortment of member types and relationship types with which you can build the required models.

Analysis

The first stage of the life cycle is concerned with analysis of the need for a new or modified system. In order to examine the effects on the enterprise of such an introduction or change, you need to produce models that show how the system would fit into the existing structure. Such models can also reveal whether the new system can be derived from an existing one.

For example, drawing an entity relationship diagram of the affected area of the enterprise allows you to examine the relationship between a new application and the existing entities. Alternatively, you can construct the model by directly entering ENTITY members into the repository, using the command or panel interface.

Design

On completion of the analysis stage, a detailed description of the new or modified application is generated. This paves the way for the second stage, design, at the end of which all the details of the system design should be stored in the repository, to be shared by members of the design team.

For example, you can use data flow diagrams to determine, for each element of the system, the required output for a given input. The data flow diagram can be derived from the entities that you described in the entity relationship diagram of the analysis stage. Such a model could also be constructed by using member types such as DATAFLOW, ENTITY, GROUP and ITEM.

Construction

Once the design is fully documented in the repository, construction of the system can begin. At this stage, you need easy access to the design information stored in the repository. The favoured diagram is therefore the repository-basic diagram. Obviously, you can also work directly with the repository via the panel or command interface.

Whichever interface you choose to use, you will appreciate the ease with which the models you use throughout the life cycle can work together to produce the end product. For example, you can readily expand the information used in the analysis stage for use in a model of the design stage. The rewards are greater consistency and productivity across the whole of the life cycle.

How to Store Information in the Repository

Entities and Relationships

A system can be represented by a model that comprises *entities* connected by *relationships*. An entity is any object or concept. For example, a program (entity) outputs (relationship) a file (entity). An entity can have any number of relationships.

Entities are represented in the repository as members. To represent the different types of entity, ASG supplies many types of members. For example, PROGRAM members represent programs. Every member must be of a particular member type, and can contain the clauses and keywords particular to that member type.

Modeling Techniques

Depending on the modeling technique in use, relationships are represented in the repository either as members or as clauses in members. With entity-relationship (ER) modeling you use members, whilst with entity-association (EA) modeling you use clauses, to represent relationships. For example, consider a program that outputs a file. With EA modeling, you represent the relationship between the program and the file using an OUTPUTS clause in a PROGRAM member. With ER modeling, you represent the same relationship using a discrete PROGRAM-OUTPUTS-FILE member.

Thus, there are two categories of member types: those that represent entities (entity member types) and those that represent relationships (relationship member types). The member types that you can use are determined by the Repository Information Model (RIM), which is defined by the administrator of the repository.

Members contain information about the characteristics, or *attributes*, of the entities they represent. For example, LANGUAGE "COBOL" and DATE-WRITTEN "7-APR-91" are clauses of a PROGRAM member that represent attributes of a program.

How to Store Different Versions of a Model

No system is static; all change with time. Models must vary accordingly. However, there is no need for you to delete a previous model or part of a model because of a change. You can store all versions of a model in the same repository by using *statuses*.

A status is a hierarchical partition of a repository. Statuses can be used to separate different models or different versions of the same model. They are set up by your repository administrator.

Your view of a model is determined by the status in which you are currently working. By changing your status, you can change the version of the model that you see.

How Incomplete Models are Stored

In developing a model, there may be times when you have insufficient information to define some or all of the clauses of a member. In these cases, you can still store the member in the repository, ready for updating once the full information becomes available.

Should you create a reference to a member that is not yet defined in the repository, it will exist as a *dummy member* until you provide the full definition at a later date.

The creation of dummy members is a significant feature. It enables you to create a useful model of a system before you have documented its every detail and thereby allows you to adopt a top-down approach to system modeling.

How to Enter and Alter Models

How to Enter Models

For entering all but the simplest of models, ASG recommends that you use the graphical interface. Diagrams thus created are automatically translated into members.

However, you can also enter models using the command or the panel interface, but it may take longer.

When you enter a member, you give it a name, specify its member type, state values for its clauses, and specify its relationship with other members. If you are uncertain about particular details, you may fill them in later. For the majority of member types, you need only specify the member name and member type.

When you have finished entering the member definition, its syntax is automatically checked. If the definition is correct, it is stored as an *encoded record*. Otherwise, it is stored as a *source record*, which you must correct before it can become part of a model. With ER modeling, additional checks are performed on the model, to ensure that it conforms to the properties defined in the ER schema.

There are a group of clauses that can be used with all member types. These *common clauses* allow you to record such things as alternative names (aliases) for a member, and the catalog classification to which the member belongs.

Mainframe Backup Facilities

If, whilst entering a model using the panel or command interface, there is a systems failure or you accidentally delete all or part of the model, do not despair. All your keystrokes in creating the model will have been recorded by the repository *logging facility*. Your administrator will be able to recover your lost data.

Also, if a process is prematurely aborted due to a system failure or human error, the *automatic recovery system* will restore the repository to its former state.

Protecting Models Against Unauthorized Access

If a model contains confidential or valuable information, the administrator may decide to prohibit certain users to display or alter all or part of the model, or even to deny their access to a whole repository or to particular statuses.

To access a repository, you must enter the password assigned to you by the administrator. However, entering the password does not automatically give you access to all parts of the repository. The administrator may have restricted your access by careful selection of your *security levels*. These protect the whole repository or particular members of the repository against unauthorized access. The administrator assigns to each user a *general security level*, which limits access to all members of the repository, and a *specific security level*, which limits access to specific members of the repository.

You will have separate security levels for the display, alteration, and removal of members. For example, if your general removal security level is too low, you will be unable to remove any members from the repository.

You may want to restrict other users' access to the model on which you are working. The administrator can arrange for you to be able to protect individual members against access by other users. There are two ways to protect members. The first is to specify an owner of the members; only users belonging to that owner can access the members. For example, you could specify an owner of ADMIN so that only administration personnel could process the members. The second way is to specify security levels for the members; only users with a sufficient general security level can process the members.

Grouping Together Parts of a Model

To simplify the processing of large parts of a model, you can store, under one name, a list of names of members. Such a list is known as a KEPT-DATA list. Dividing a model into groups allows you to change or interrogate specific parts of the model.

At the end of a session, KEPT-DATA lists are deleted, which may mean the loss of many hours of work. However, you can store them permanently in a storage area called the MP-AID. They can be deleted when no longer required.

Finding Out the Properties of a Model

Repository functions are provided to enable you to examine all or particular parts of a model by displaying the clauses and relationships of its constituent members.

The result of most interrogations is a list of names of members that satisfy the criteria you specify. These criteria can be almost anything to do with members, including their relationships, clauses and history.

For example, in developing a model, you may choose to remove a redundant member. Before doing so, you should find out which members will be affected by the removal. You can ask the repository "which members have a relationship with this member?", and you will obtain a list of member names, which you could keep in a KEPT-DATA list. Before you can remove the member, you must remove its relationships.

You may want to inform the departments responsible for these members that you are removing the member. The list of department names could be obtained by displaying the CATALOG clause of the members in the KEPT-DATA list. You could then check with those departments that it is OK to remove the member.

As a final check, you could find out who has ever processed the member, and inform them of your intentions. Thus assured, you can proceed with the member's removal.

3 The User Interface

Introduction

ASG provides three kinds of user interface, all of which can be used to communicate with the repository - to enter, alter, interrogate and manipulate information held there. These interfaces are the:

- Graphical interface
- Panel interface
- Command interface.

Each of the interfaces is suited to particular tasks and levels of user's expertise.

Figure 2 on page 12 is an example diagram from the graphical interface. This is aimed at users, such as business analysts and systems analysts, who need to produce complete models of systems or departments, and to be able to demonstrate them to other people. Using the graphical interface is the fastest way to set up new models.

The graphical interface has its own repository, called the local repository. The local repository is separate from the host repository, which is used with the command interface and the panel interface. However, information can be freely transferred between the local repository and the host repository. This enables you to display and modify, on the graphical interface, diagrams of members that were entered using the command or panel interface. Conversely, you can translate into members of the host repository, models that were entered using the graphical interface.

Refer to *ASG-ManagerView User's Guide* for details of using the graphical interface.

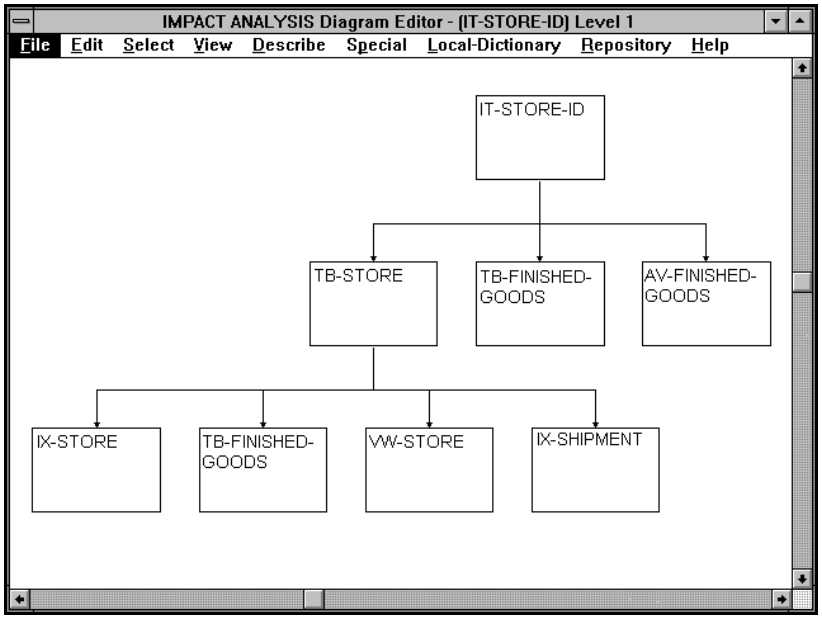


Figure 2. The Graphical Interface

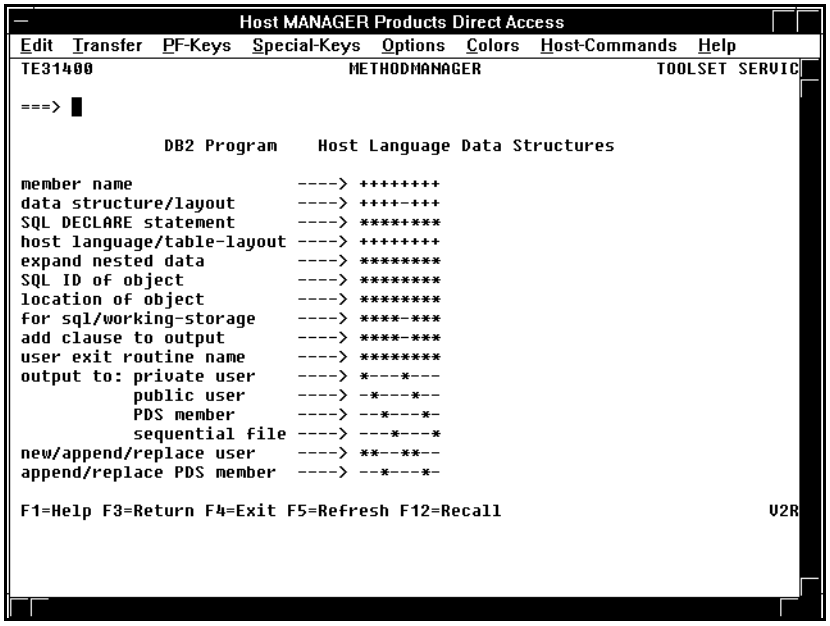


Figure 3. The Panel Interface

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
COMMAND MODE:                                          LINE: 00000 OF 00015 WAITI
      V...+...10....+...20....+...30....+...40....+...50....+...60....+...70.
***** *** TOP OF DATA ***
***** THE FOLLOWING SELECTED MEMBERS SATISFY THE GIVEN CRITERIA
*****                MODULE                          MOD-GET-EMP-MASTER
*****                MODULE                          MOD-GET-HIST-MASTER
*****                MODULE                          MOD-GET-SORT-TRANS
*****                MODULE                          MOD-STD-EDIT
*****                MODULE                          MOD-TRANS-MAIN-EDIT
*****                MODULE                          MOD-UPD-ERR-RPT
*****                MODULE                          MOD-UPD-HIST
*****                MODULE                          MOD-UPD-TRANS-MASTER
*****                MODULE                          MOD-VALIDATE-TRANS
*****                PROGRAM                         PROG-RPT-MASTER
*****                PROGRAM                         PROG-UPD-EMP-MASTER
*****                PROGRAM                         PROG-UPD-HIST
*****                PROGRAM                         PROG-VALIDATE-TRANS
*****            13  MEMBERS IN TOTAL
***** *** END OF DATA ***


====> which members have catalogue = "cobol"
                                                                        | V2R2
```

Figure 4. The Command Interface

Figure 3 on page 12 shows a typical panel from the panel interface. The panel interface presents you with menus to guide you through tasks; selections are entered via input panels. Although the use of panels is largely intuitive, in-context help is available.

The panel interface is aimed largely at end users who require an easy-to-use means to manipulate specific repository information; for example, a programmer updating particular clauses of repository members.

Refer to "Accessing the Panel Interface" on page 14 for details of accessing the panel interface and its structure.

Refer to Chapter 4, "Using the Panel Interface," on page 21 for details of using the panel interface.

Figure 4 on page 13 is an example screen from the command interface, which uses the traditional computer input line. The users of the command interface will tend to be the same as those of the panel interface, but with perhaps more experience. Although less friendly, the command interface can be the fastest way for an experienced user to perform a task. Repetitive tasks, such as changing the same clause in many members, can be performed in batches, leaving the user free to work on other tasks.

Refer to the *ASG-Manager Products Dictionary/Repository User's Guide* and the *ASG-ControlManager User's Guide* for details of using the command interface.

Accessing the Panel Interface

The panel interface provides you with easy access to the tools you use to develop applications.

The phases of development, as you have seen in Chapter 2, "Modeling Systems and Information," on page 5, are the analysis, design and construction of models representing the system and application that you are developing. To support this process, the tools in the panel interface enable you to:

- Design the models of a system
- Create and update definitions of models in the repository
- Check and query the definitions in the repository
- Import definitions from or export definitions to database systems such as DB2 or IMS
- Change your repository environment.

From Chapter 1, "Introduction to MethodManager," on page 1 you know that the two components of the panel interface are:

- ToolSet SERVICES, which provides the CASE tools that you use
- LifeCycle SERVICES, which provides a predefined method for development, so that no aspect of a development project can be neglected.

When you logon to ASG-Manager Products (herein called Manager Products) you usually access the command interface. Your systems administrator may have set up your environment so that you automatically access that part of the panel interface you most frequently use. If not, you need to enter either the LCS or TSS command.

To access LifeCycle SERVICES, enter:

```
LCS ;
```

To access a selected panel in LifeCycle SERVICES, enter either:

```
LCS panel-name ;
```

or:

```
LCS n ;
```

where:

panel-name is the name of a panel.

n is one or more options from LifeCycle SERVICES menus, separated by full stops.

To access ToolSet SERVICES, enter:

```
TSS ;
```

To access a selected panel in ToolSet SERVICES, enter either:

`TSS panel-name ;`

or:

`TSS n ;`

where:

panel-name is the name of a panel

n is one or more options from Toolset SERVICES menus, separated by full stops.

If you have your own logon profile or user member defined, you can add one of these commands, including the terminator, so that it is run as a part of a batch job when you logon. If you enter the commands interactively in the command interface you do not need to include the terminators.

Note:

Once you access the panel interface using the LCS command, you access TSS by selecting option 4 from the Information Engineering menu.

Refer to Chapter 4, "Using the Panel Interface," on page 21 for details of moving around the panel interface, and the types of panel you will find.

Refer to Chapter 5, "The Documentation Facility," on page 37 for details of the commands and line commands that you can use in the panel interface.

Refer to the *ASG-Manager Products Quick Reference* manual for the syntax of the LCS and TSS commands.

Refer to the *ASG-ControlManager User's Guide* and the *ASG-Manager Products Dictionary/Repository User's Guide* for details of the command interface.

Leaving the Panel Interface

You can leave the panel interface from the main menus, that is, either the TSS or the LCS menu.

When you select option X Exit, you are returned to the command interface. If your administrator has tailored your environment, you may exit Manager Products completely.

The Structure of the Panel Interface

The following charts show the options available from the:

- LifeCycle SERVICES main menu
- ToolSet SERVICES main menu
- ASG-supplied strategic information planning project
- Tutorial help menu.

In the structure charts:

- Each box represents a branch of the panel interface
- Each circle contains a reference to another chart where the structure of the branch is expanded.

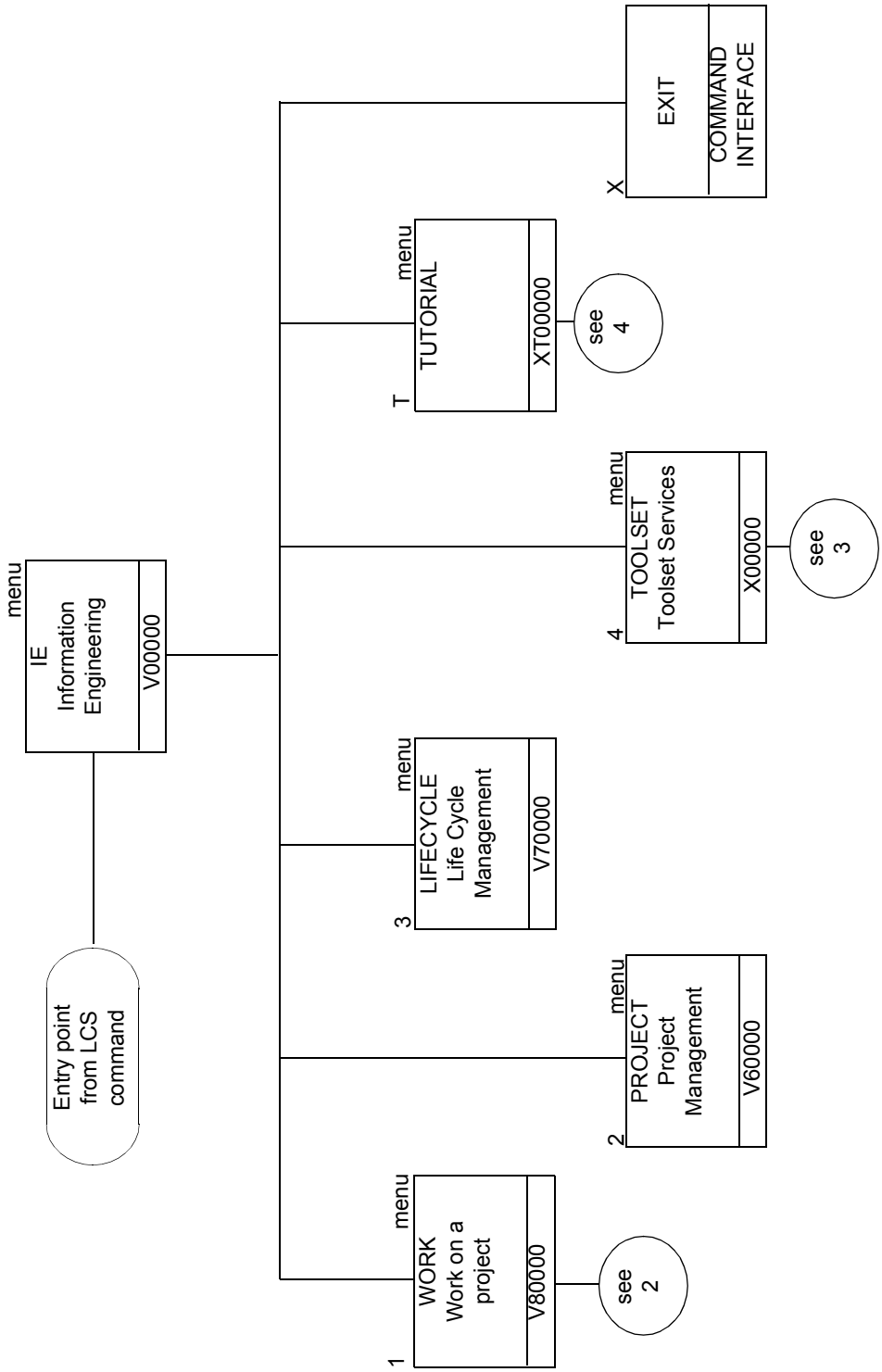


Figure 5. LifeCycle SERVICES

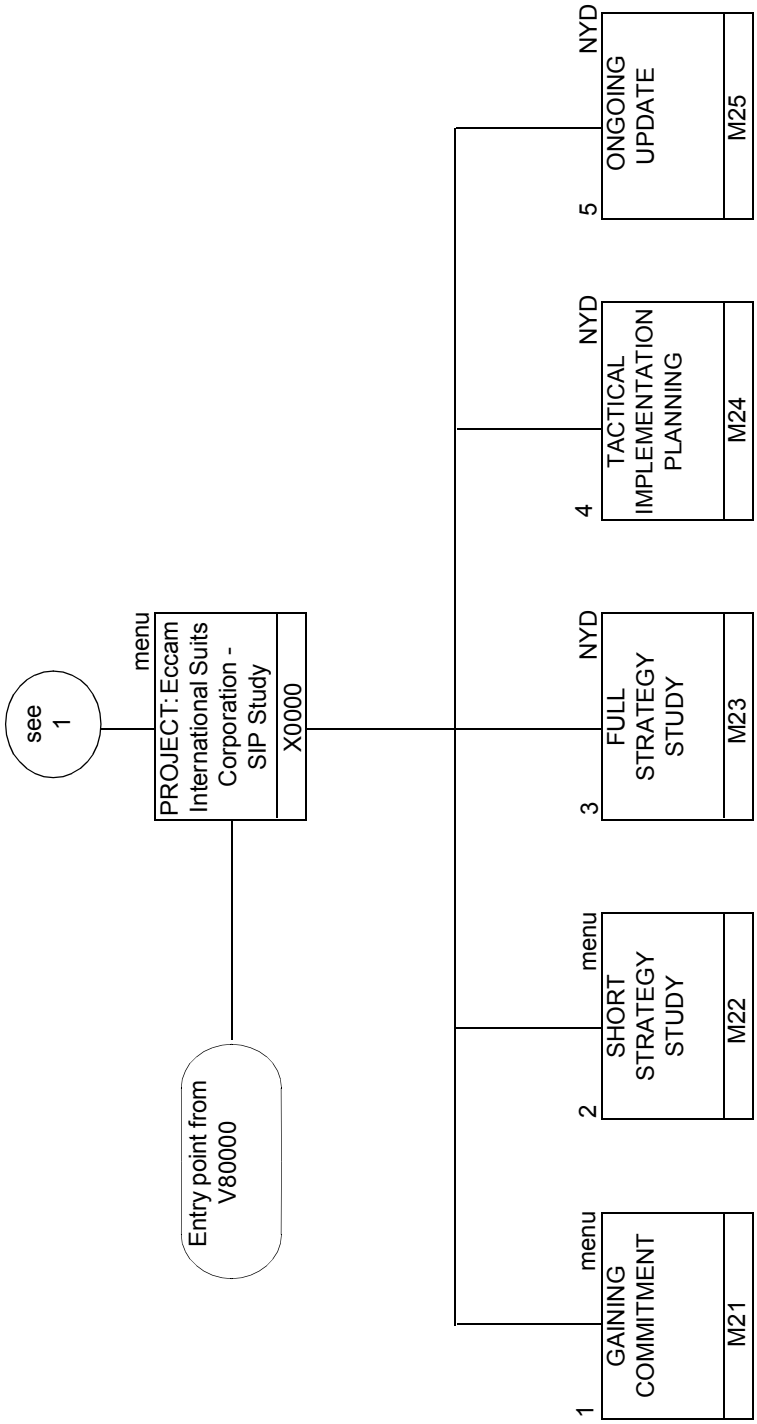


Figure 6. Strategic Information Planning

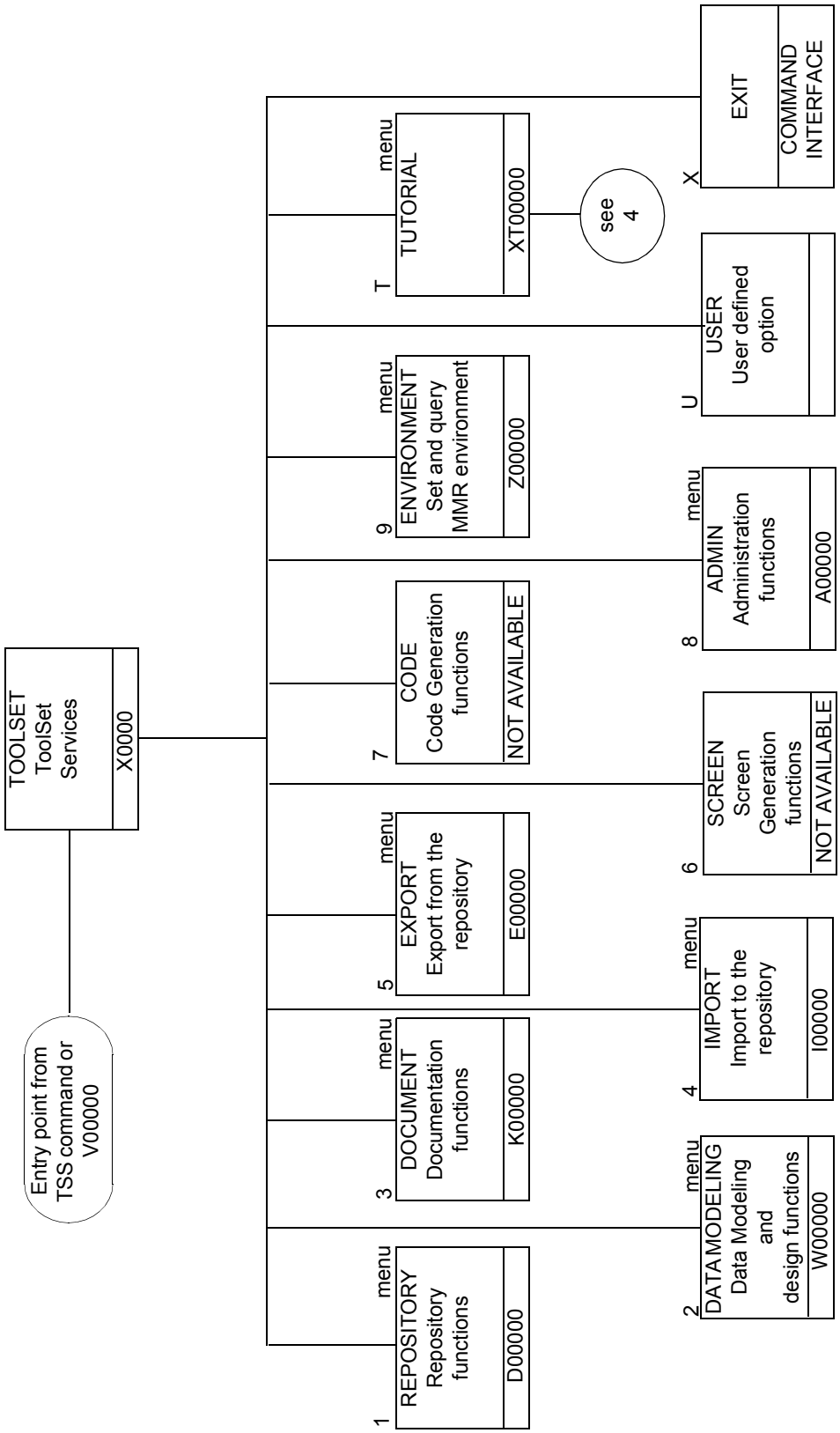


Figure 7. ToolSet SERVICES

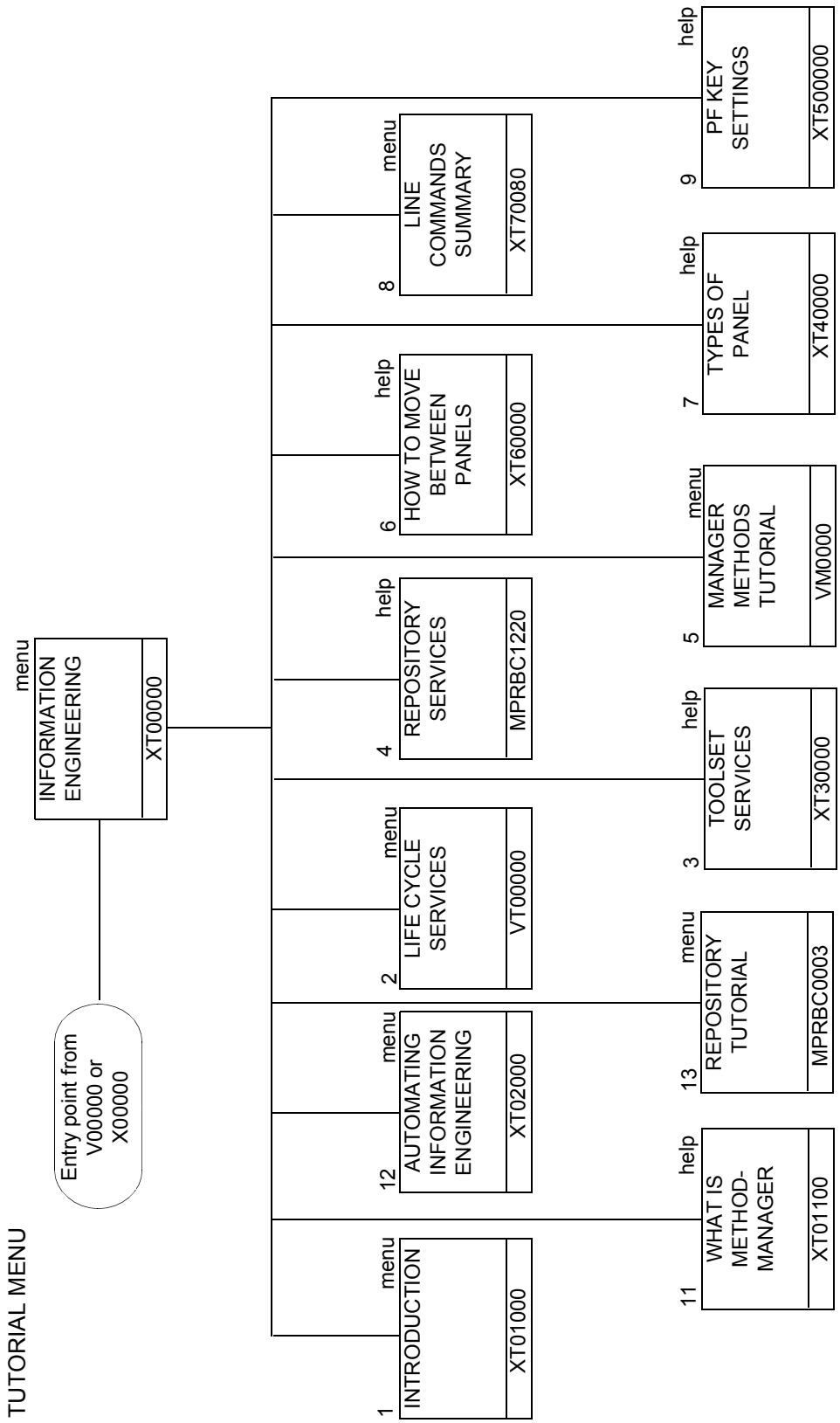


Figure 8. Tutorial Help

4 Using the Panel Interface

Moving Around

The figures in this chapter show you what the various types of panel look like. Panels may appear to be different in your installation because they can be tailored by your system administrator. For example the top line may contain information about which repository and status you are currently using and the function keys may be different.

The main menu that you use is either the Information Engineering menu or the ToolSet SERVICES menu, depending on the command that you use to access the panel interface. Moving around is simply a matter of selecting options from menus.

Figure 10 on page 22 shows an example of a menu. The different parts of the screen are labeled. To select menu options, enter the option number you require in the command area. For example, Figure 9 on page 21 represents a hierarchical structure of panels. To display panel 1.2, select:

- Option 1 from menu 0
- Option 2 from menu 1,

as shown by the solid double lines:

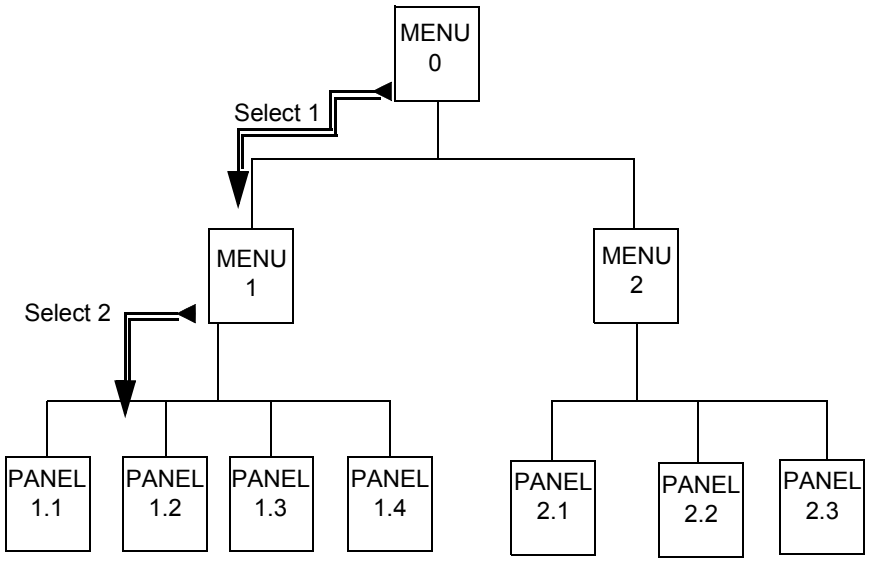


Figure 9

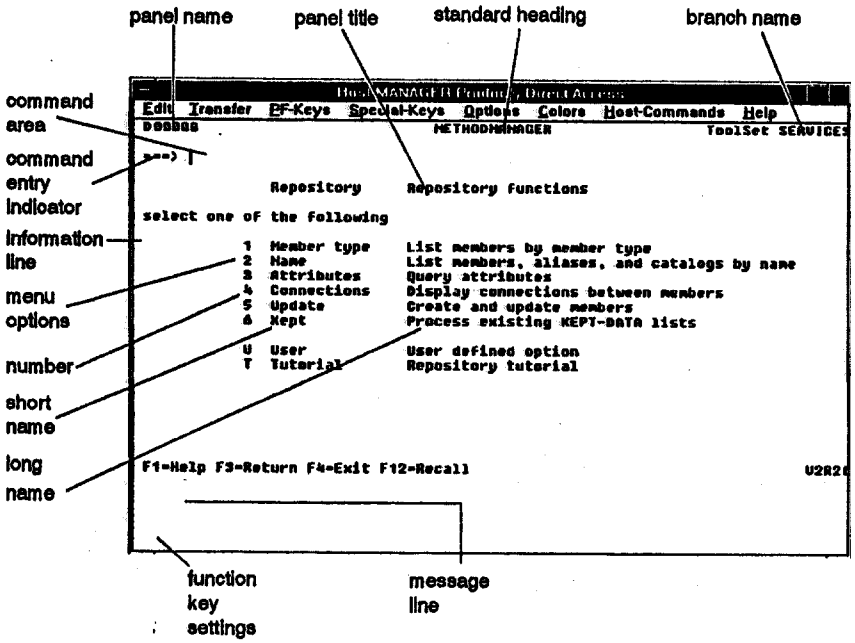


Figure 10. Repository Functions Menu

To return to the previous panel, use PF3. To return to the main menu, (either the ToolSet SERVICES or LifeCycle SERVICES menu) use PF4. The action of PF3 and PF4 is shown by the dotted lines in Figure 11 on page 23.

When you are familiar with the structure of the panel interface, you can use quicker, more direct methods of displaying panels.

You can display a panel, without displaying intervening menus, if you know the option numbers to select from each menu. Enter the option numbers as a continuous string, separated by full stops. For example, to display panel 2.3, enter:

```
2.3
```

in the command area of menu 0. The action of the command is shown by the solid double lines in Figure 11 on page 23.

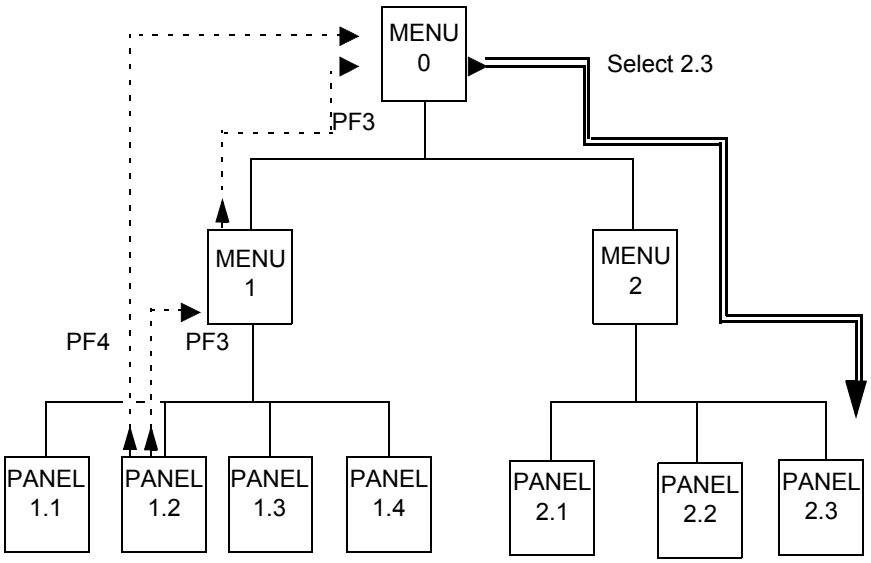


Figure 11

You can also display one panel from another panel without retracing to a menu. Enter either a plus (+) or equals (=) symbol followed by the option numbers you would select from the main menu. Depending on which symbol you use, you return to a different panel when you press PF3.

Note:

If you access a panel regularly, you can use the ToolSet SERVICES panel, TZ30000, to define that a function key executes the routing command, that is the plus (+) or equals (=) symbol, followed by the option number.

For example, if you are working on panel 1.2 and you want to display panel 2.3, but not any of the intervening menus, enter:

```
=2.3
```

in the command area of panel 1.2. The routing command is represented by solid double lines in the diagram. When you press PF3 to quit panel 2.3, you return to the main menu (menu 0). This is represented by the dotted line in Figure 12 on page 24.

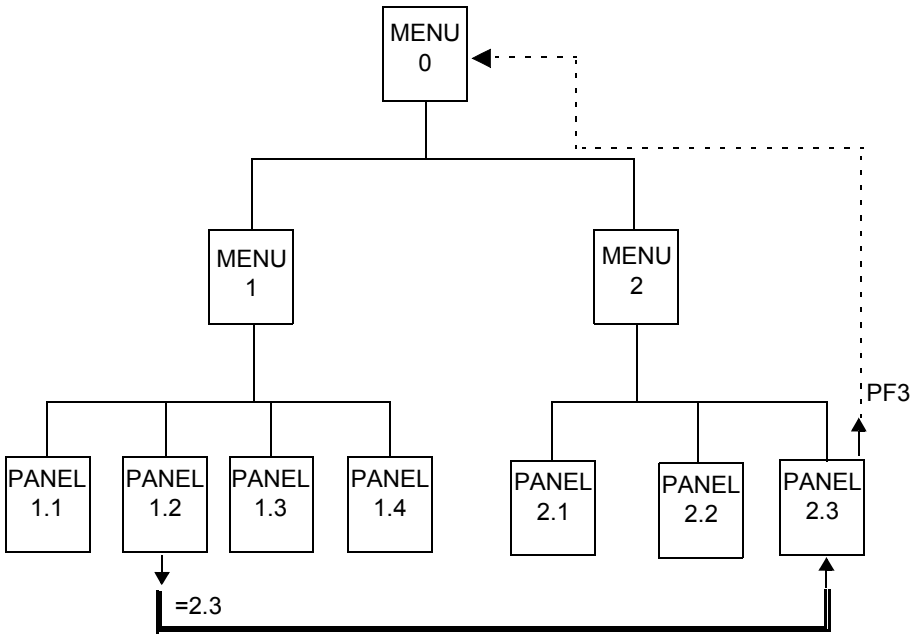


Figure 12

If, from panel 1.2, you enter:

```
+2.3
```

then you display panel 2.3, as shown by the double solid lines. But, when you press PF3 to quit panel 2.3, you return to the panel that you came from. This is shown by the dotted line in Figure 13 on page 25. You start and end with panel 1.2.

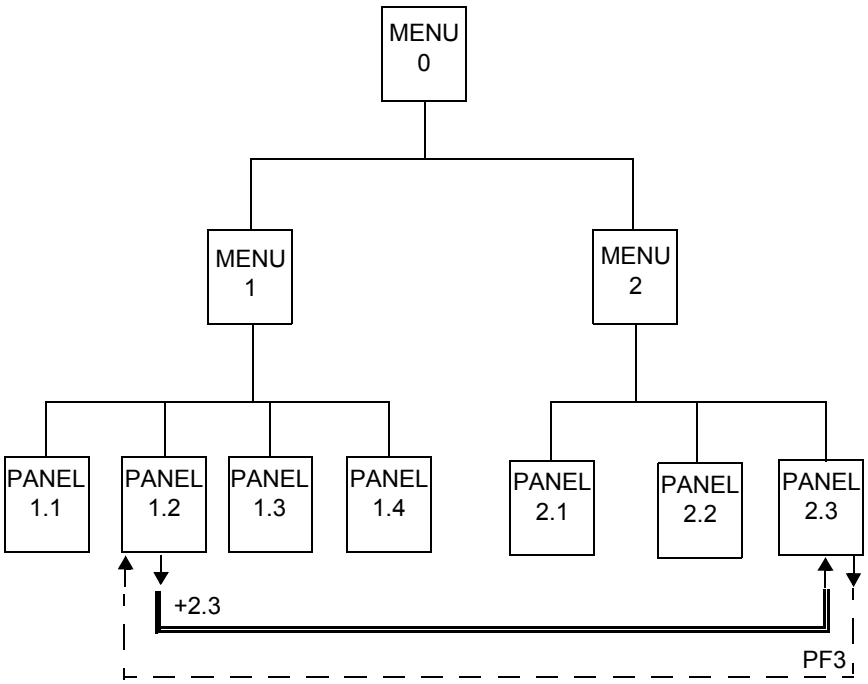

MENU
0
MENU
1
MENU
2
PANEL
1.1
PANEL
1.2
PANEL
1.3
PANEL
1.4
PANEL
2.1
PANEL
2.2
PANEL
2.3
+2.3
PF3

Figure 13

Function Key Settings

The function keys you can use in any menu, selection and input panels are displayed at the bottom of the screen. The standard, default settings are:

Function Key	Definition
PF1 & PF13	for the in-context help on a panel.
PF3 & PF15	to return to the previous panel.
PF4 & PF16	to return to the main menu.
PF5 & PF17	to refresh the screen, clearing messages and input.
PF7 & PF19	to scroll backwards.
PF8 & PF20	to scroll forwards.
PF12 & PF24	to recall commands entered in the command area.

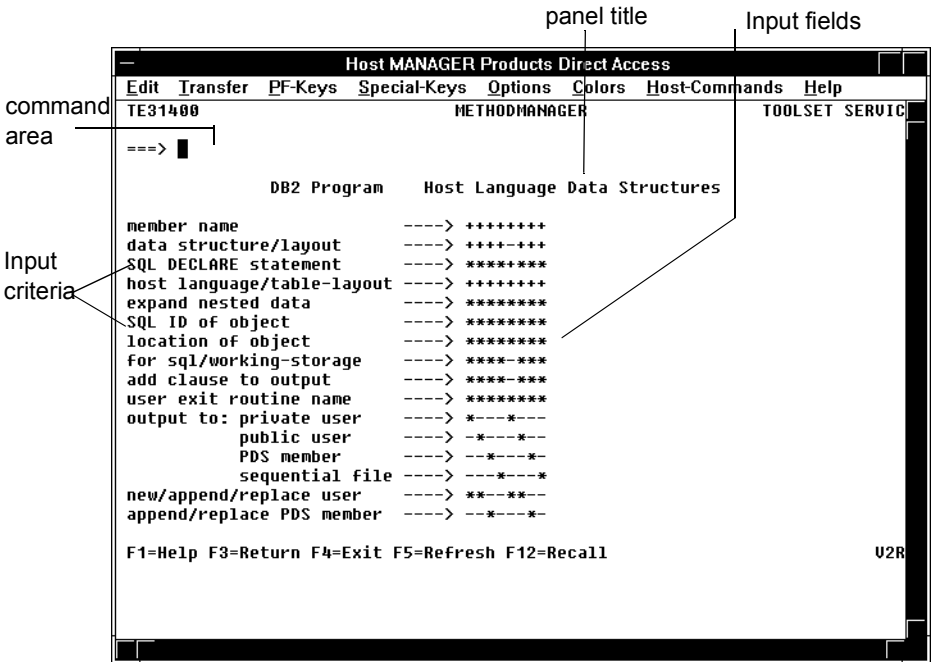


Figure 14. Input Panel

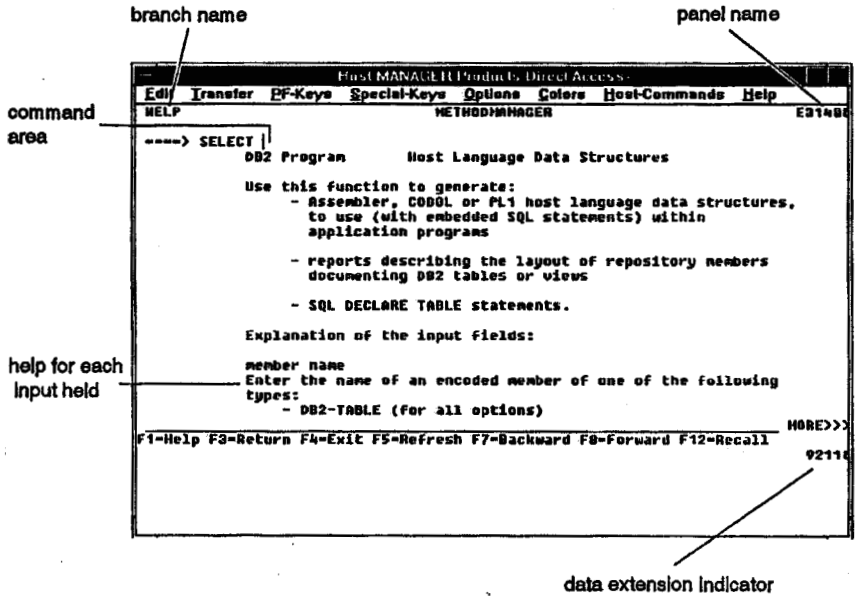


Figure 15. In-context Help Panel

You can use the PF key panel (TZ30000) to set PF2, PF6, PF9, PF10 and PF11 to your own values. For example, the following commands can be useful when set as function keys if you generate a long list of members:

FORWARD	TOP	UP	LOCATE	+ *option-number*
BACKWARD	BOTTOM	DOWN	NEXT	= *option-number*

Note:

Assisted update panels, expert panels and LOOKASIDE buffers use the function keys (except PF3 and PF4) set in the command interface. These may be different from the function keys set for the panel interface using TZ30000.

Refer to Chapter 6, "Summary of Commands," on page 99 for a summary of the Manager Products commands that you can enter in the panel interface, and refer to the *ASG-Manager Products Quick Reference* manual for details of the full syntax of all commands.

Types of Panel

Menu Panels

Menu panels direct you to specific panels that enable you to interrogate or change the repository or to import or export data. Figure 10 on page 22 shows a menu panel. To select an option from a menu, enter its number in the command area.

Input Panels

Input panels provide an easy means of entering complex interrogations or instructions. Figure 14 on page 26 shows a typical input panel.

If you are not sure what data to key into an input field, enter:

- PF1 for in-context help explaining what to enter in each input area
- ? in a specific input field to display a table showing the input fields that must be filled in, and those that are optional.

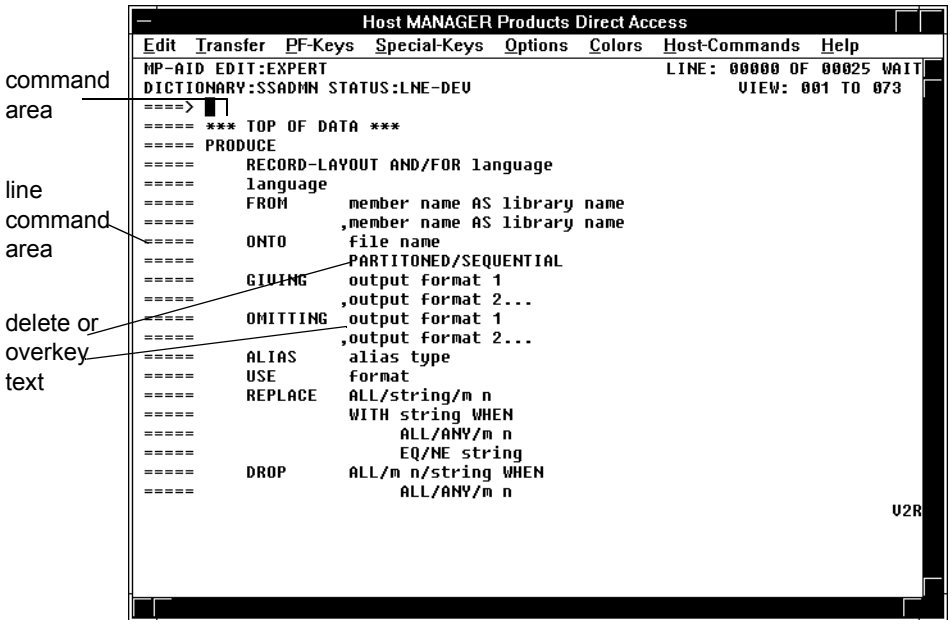


Figure 16. Expert Input Panel

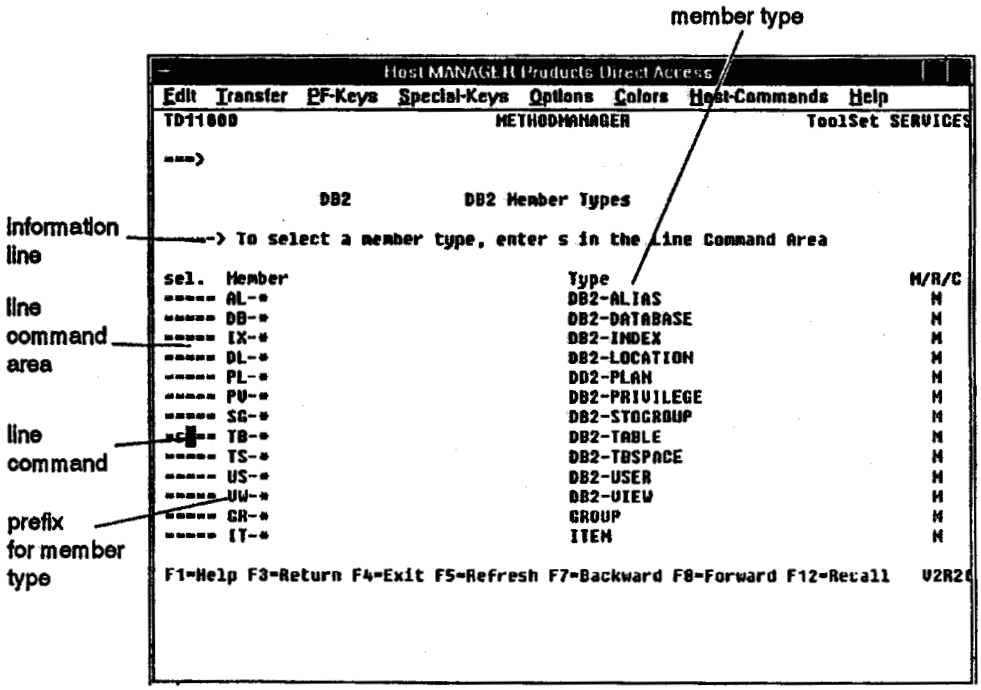


Figure 17. Selection Panel

When you have keyed in the data, press ENTER to action it and display any output resulting from the interrogation or instruction issued.

In-context Help Panels

In-context help panels are available for every panel and are accessed using PF1. Figure 15 on page 27 shows an in-context help panel that explains how to use the DB2 Program panel. If the help extends over more than one screen, the following symbols are displayed on the screen:

- MORE>>> in the bottom right hand corner indicates more information follows a screen
- <<<MORE in the top left hand corner indicates more information precedes a screen.

Use PF7 and PF8 to scroll backwards and forwards. Use PF3 to return to the panel from which you requested help.

Expert Input Panels

Expert input panels (also referred to as Expert Mode) are provided for those of you who are familiar with commands. They provide a skeleton of the keywords and clauses of the command and so save you having to remember and key in complex command syntax. You can use line commands to delete the clauses that you do not require, and you can enter specific values in other clauses. Figure 16 on page 28 shows an expert input panel.

You can use the panel shown in Figure 16 on page 28 to generate record layouts in a selected programming language.

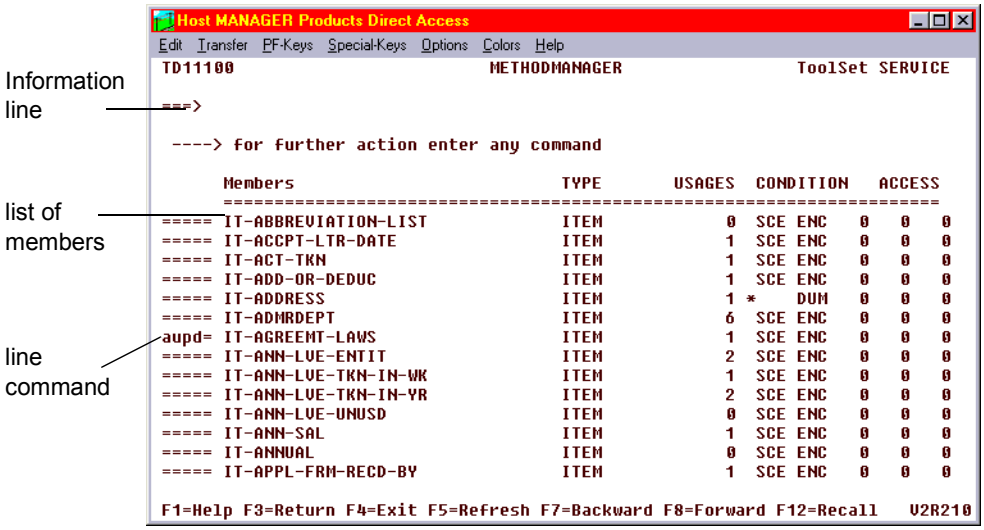


Figure 18. How to Display Member

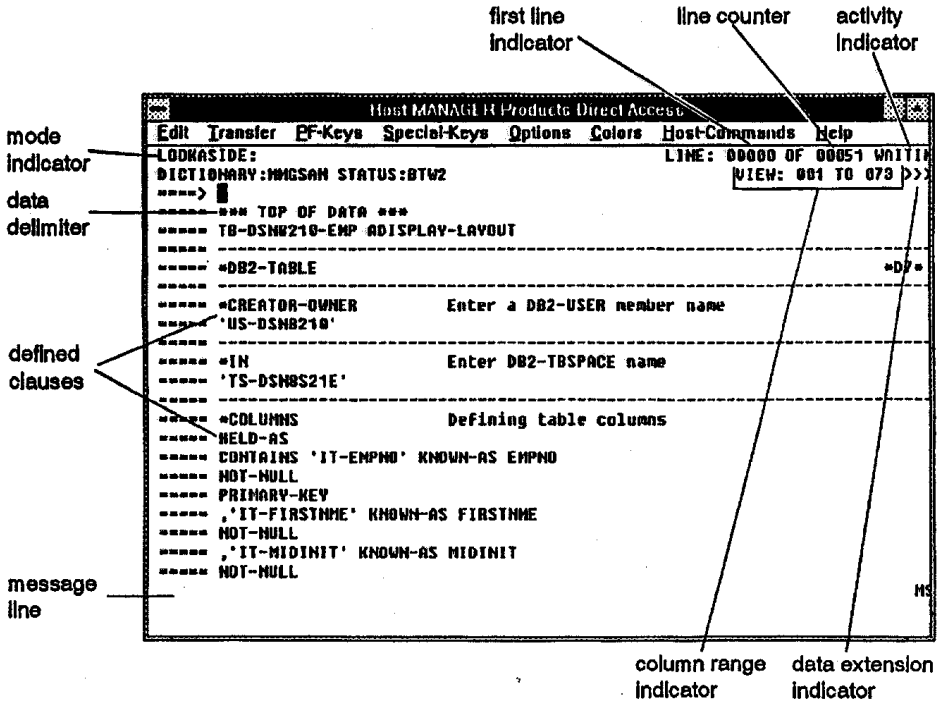


Figure 19. Assisted Display Panel

For example, to generate a COBOL record layout using the HELD-AS clause of a repository member and direct the generated output to a specific file, you would:

- Delete AND/ from the second line and overkey "language" with COBOL
- Overkey "member name" in the FROM clause with a member name, for example RC-XB4
- Overkey "file name" in the ONTO clause with the file name, for example XB_COB-4
- Overkey "format" in the USE clause with HELD AS
- Delete all the other lines and text, except the terminator at the end of the panel.

The terminator is not visible in Figure 16 on page 28 because the panel extends over more than one screen.

The remaining command is:

```
PRODUCE
    RECORD-LAYOUTS FOR COBOL
    FROM RC-XB4
    ONTO XB_COB_4
    USE  HELD-AS 3
;
```

To action the command, enter:

```
RUN
```

Refer to Chapter 6, "Summary of Commands," on page 99 for details of line commands.

Selection Panels

Selection panels display a list from which you can select a member. Figure 17 on page 29 shows a selection panel of a list of member types.

Selection panels have a range of functions. For example, you could select a member to update it, or to delete it from the repository. The information line explains what the selection list is for.

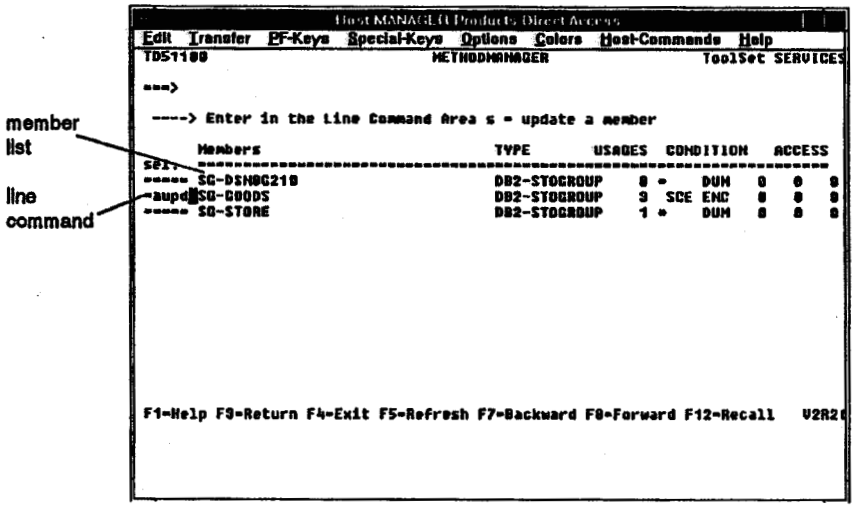


Figure 20. How to Update a Member

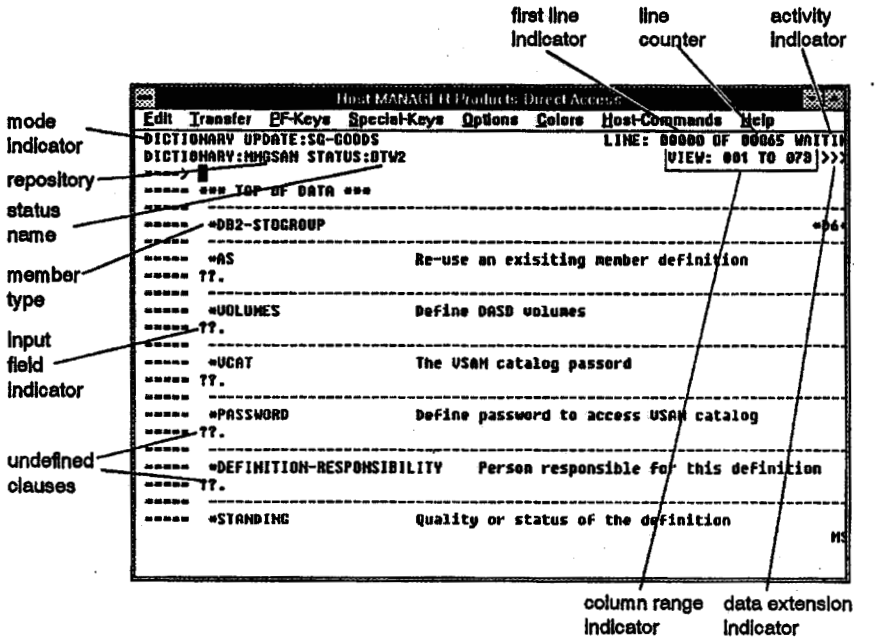


Figure 21. Assisted Update Panel

To select a member from a list, enter `S` in the line command area beside it. In Figure 17 on page 29, selecting a member type displays a list of all the existing members of that type.

The three character prefix for each different member type is provided to ensure that all member names meet the naming conventions in your installation.

On selection panels listing member types, symbols are displayed after the member type prefix. In Figure 17 on page 29, for example, all prefixes are followed by an asterisk (*), which means you can enter a member name with a maximum of 32 characters, including the prefix. However some symbols indicate a restriction on the number or type of characters that you can enter.

Naming Convention	Means	You Enter
EG-*	any string, up to 32 characters	EG-example
EG-_	one character	EG-x
EG-#	an integer	EG-1234
EG-_<	maximum length	EG-x or EG-xy
EG-__=	exact length	EG-xyz
EG-_>	minimum length	EG-xy or EG-longname

In some selection panels of members, the column heading M/R/C is displayed at the right hand side of the screen. For each member in the selection list, the column contains one of the following entries:

M means the member is an ordinary member type.
R means the member is a relationship member type.
C means the member is a collective member type.

A collective member type is one that represents several others, for example, DATABASES represents DB2-DATABASE, TOTAL-DATABASE or IMS-DATABASE.

Assisted Display Panels

Assisted Display panels display members in the repository that represent entities in a system.

To display a repository member, enter:

```
ADIS
```

in the line command area, as shown in Figure 18 on page 30. Figure 19 on page 31 shows a displayed member. The display is shown in a LOOKASIDE buffer.

As you work in the panel interface you will display LOOKASIDE buffers containing:

- Information in response to a command or interrogation that you have issued
- Processing error messages

as well as assisted displays of member definitions. LOOKASIDE buffers provide you with information and cannot be updated or edited.

To quit a LOOKASIDE buffer and return to the previous panel or menu, use PF3.

Assisted Update Panels

Assisted Update panels enable you to update repository members. They have a similar layout to assisted display panels. To update a member, either enter:

```
AUPD
```

or, where the information line states `S = assisted update`, enter:

```
S
```

in the line command area beside the member name.

Figure 20 on page 32 shows the AUPD line command against a member and Figure 21 on page 32 shows the assisted update panel. The symbols ??. indicate where you can input data. If you want help on what data to enter in the clauses, use PF1. To file the updated member, use PF3.

When you file an updated member, it is checked to make sure that its syntax is correct. If the member meets these checks it is encoded in the repository. If it does not meet the checks, the software outputs messages indicating where errors occur in a LOOKASIDE buffer. If you do not want to correct the errors, you can either abandon your updates by entering:

```
CANCEL ;
```

in the command area, or you can file the member definition without encoding it by entering either:

```
ACCEPT ;
```

or:

```
RETURN ;
```

Getting Help

For a general overview of what you can do with the functions in a particular branch of the panel interface, select option `T` for tutorial help. Access to tutorial help is available from the main menus.

To display help on a specific subject, enter:

```
HELP subject ;
```

in the command area.

To list all the help topics beginning with a specific character, enter:

`HELP character ;`

where:

character is usually a letter of the alphabet, but you can also enter a symbol, such as a colon (:) which is used as a naming prefix for macros.

The index of documented topics beginning with that letter is displayed. You can page through the list to find a topic, or use the LOCATE command. To display the help for a topic, enter:

`PAN`

in the line command area beside the topic.

To display help about the stage of a project that you are working on, enter `H` beside the option of the project menu.

To display in-context help about a menu or what you can do with a panel use PF1. Figure 15 on page 27 shows an in-context help panel.

Command	Displays
`INFOBANK`	InfoBank entry panel
`HELP`	top level help menu
`SELECT TOP-MENU`	top level help menu
`HELP` followed by:	
`INDEX`	subjects in help branch
command-name	menu for command description
command-name-`S`	command syntax
member-type	menu for member type description
topic	panel describing topic
letter	subjects beginning with a letter
`PF`	list of default PF key settings
`PANEL` followed by:	
`INDEX`*letter*	subjects beginning with letter
panel-name	displays that panel
message-name	displays that message panel

5 The Documentation Facility

This chapter describes the documentation facility in MethodManager. This facility allows you to produce documents such as reports and manuals automatically, incorporating information stored in your Manager Products repository.

Its use promotes an efficient and orderly approach to corporate documentation. Its philosophy follows that of all Manager Products in that DOCUMENTs are represented by a member type in the Manager Products repository which may refer to other members and to which other DOCUMENTs may refer. Reference chains up to 10 deep are supported.

If you follow the recommended practice of storing standard paragraphs, descriptions ('contextual units') and formats as individual DOCUMENTs to which other relevant DOCUMENTs refer, you not only ensure absolute consistency of corporate presentation, but also eliminate wasteful duplication of effort. Whenever you update a standard paragraph or DOCUMENT, the update is applied automatically and immediately in all DOCUMENTs which refer to it. You do not need to spend time and effort applying the same update manually to several affected documents.

To make them easier to read, your DOCUMENTs may include headers and footers. A table of contents is created automatically. Pages and headings are numbered, the number sequence being maintained automatically. Finished DOCUMENTs may be sent to the screen or output as files for printing. The output DOCUMENT can also be downloaded as a text file for processing in your PWS using a word processor or desktop publishing application.

This manual assumes that you are familiar with the major functions of MethodManager. If not, you are advised to acquaint yourself with these first.

Document Definition Contents

The content and format of each DOCUMENT that you produce is controlled by its *DOCUMENT definition*. This is stored in the repository as the member type DOCUMENT. It has a default naming convention prefix of DC-, although you may adopt a different naming convention if you prefer.

A DOCUMENT definition may contain:

- "Static" text which is normally incorporated in the DOCUMENT as it stands, but may contain variables
- DOCUMENT assembly commands, which extract material from other repository members
- Manager Products commands, whose results are incorporated into the document
- DOCUMENT formatting commands, which control the layout of the finished document
- Comments, which do not affect the finished document, but make the definition more intelligible.

By using documentation commands and Manager Products commands in your DOCUMENT definition, you may incorporate data from the repository into your finished document. This may consist of whole DOCUMENTs or individual "clauses" or attributes extracted from any member type.

Because this data is copied into the DOCUMENT when it is assembled, not when the definition is created, its content, unlike static text, varies depending on the current contents of those other repository members. So, for example, you could use the same DOCUMENT definition at regular intervals to produce up-to-date reports on the repository contents.

The Stages in Document Production

Document production normally involves the following four stages:

- Writing the DOCUMENT definition, as described in "Writing Document Definitions" on page 38.
- Using the DOC command to assemble the DOCUMENT, as described in "The DOC Command" on page 94.
- Using the HARDCOPY command to obtain printer output, as described in "The SDOC Command" on page 95.
- Formatting the assembled DOCUMENT for output to your screen, your printer or to a file for processing in your PWS. This is described in "The SDOC Command" on page 95.

Writing Document Definitions

The document definition controls the content and format of the document that you produce using the documentation facility in MethodManager. It is stored in the Manager Products repository as the member type DOCUMENT. Standard practice is to distinguish DOCUMENT members by the DC- prefix, but you may follow a different naming convention if you wish.

Creating and Updating Document Definitions

To create a new DOCUMENT definition proceed as follows:

- Display MethodManager's ToolSet SERVICES (TSS) main menu
- Choose option `3`, Documentation Services
- When the Documentation Services menu is displayed, in the command area type:

  `AUPD` *`document-name`*

where *`document-name`* is the name of the new DOCUMENT.

You must use a name that is not currently being used for a DOCUMENT member. If you enter the name of an existing member, its definition will be loaded into an assisted update buffer for amendment. If you enter a new name, an empty assisted update buffer will be created in which you may enter your new document definition.

If you cannot remember the names of your existing DOCUMENT members, you may obtain a list of them by using option `1` or `5` from the documentation menu. You will be required to specify the naming convention that you use, that is the prefix that you apply to the names of your DOCUMENT members. Type `S` in the line command area to select your naming convention.

Alternatively, if you already have in your repository a DOCUMENT member that is similar to the one that you wish to create, you may use the DCUPD command in the TSS command area to make a copy of that member and give it a new name. The syntax is:

`DCUPD` *`new-document old-document`*

where *`new-document`* and *`old-document`* are the names of the new and the existing DOCUMENT members respectively. The new DOCUMENT definition will appear in an assisted update buffer, ready for amending.

This saves considerable duplication of effort and is especially useful, for example, if your organization uses forms and other documents that have a standard layout and, perhaps, other material in common.

To update an existing DOCUMENT definition, you may enter `AUPD` *`document-name`* in the command area. Alternatively, use options `1` or `5` from the documentation menu to list the DOCUMENTs currently in your repository and type `AUPD` in the appropriate line command area.

The DOCUMENT definition will be loaded into a buffer for updating using the standard MethodManager editor. The AUPD command uses the assisted update buffer which includes all the clauses and attributes, even if they are empty. An alternative is the UPD command which uses the standard update buffer. This displays only those clauses and attributes to which you have made changes.

When you have finished updating your DOCUMENT definition, type `file` in the command area to store it in your repository.

Attributes of a DOCUMENT Member

The most important attributes of a DOCUMENT member are the NOTE, the TITLE and the CONTENTS.

- The NOTE attribute consists of one or more lines of text. It normally contains the DOCUMENT's title; it is used as an introductory heading when the DOCUMENT is reproduced in another DOCUMENT.
- The TITLE attribute is similar to the NOTE in that it consists of one or more lines of text and normally contains the DOCUMENT's title. When the DOCUMENT is incorporated into another DOCUMENT using the INCLUDE command, its TITLE is used as an entry in a list of contents which is compiled automatically; an error message is given at assembly time if this attribute is undefined in a DOCUMENT member that is referenced.
- The CONTENTS attribute contains the text and commands from which the body of the DOCUMENT is assembled.

There is also a SEE clause which lists the other repository members to which the DOCUMENT refers. It is created and maintained automatically from the INCLUDE, EXTRACT and BODY commands in the CONTENTS. Because of its special usage, its contents should not be changed by the user.

Besides these, a DOCUMENT member may have any of the common attributes or clauses; the assisted update (AUPD) template allows you to use as many of these as you wish. For further information on these, refer to Manager Products technical documentation.

Your Systems Administrator can change the attributes of DOCUMENT members if required.

Types of Entry in the CONTENTS

You may include the following types of entry in the CONTENTS attribute:

- "Static" text which is normally incorporated into the DOCUMENT as it stands, but may include variables whose values are set at assembly or formatting time
- DOCUMENT assembly commands, only available within DOCUMENT members, which control the finished DOCUMENTS's content
- Manager Products commands, executed at assembly time and whose results are incorporated into the DOCUMENT
- DOCUMENT formatting commands, executed when the assembled document is formatted, which control the finished DOCUMENT's layout
- Comments, not reproduced in the finished document, which make the definition more intelligible.

Escape Characters

Each entry must begin on a new line. You indicate which type of entry you are creating by placing an "escape character" or sometimes a sequence of two escape characters at the beginning of the entry, that is, in column 1 (and 2). By default the escape characters are defined as follows:

- ?? for DOCUMENT commands, including both assembly and formatting commands
- @ for Manager Products commands
- * for comment lines.

Static text does not begin with an escape character. Accordingly, any entry which does not begin with an escape character is treated as static text and is copied into the finished document as it stands.

Your Systems Administrator can redefine the escape characters in use on your system.

You may redefine the DOCUMENT command escape sequence (?? by default) from within a DOCUMENT definition using the ESCAPE command as described in "The ESCAPE Command" on page 71. This may be useful if the context requires a line of static text to begin with a text string which is currently defined as the escape sequence for DOCUMENT commands.

You can find which text strings are currently defined as escape characters (and also the settings of some other defaults) by entering DOCUQUERY in the command area. This facility is described in "Other Documentation Facilities" on page 97.

Static Text

You may enter lines of static text anywhere in the CONTENTS attribute of your DOCUMENT. No escape character is needed, since all lines which do not begin with an escape character are regarded as static text. If the context demands that a line should begin with a character currently defined as an escape character, your Systems Administrator can redefine the escape character.

Static text is normally reproduced in your document as it stands. It may, however, contain variables. See "Passing Values into Variables" on page 49, "The PARAMETER Command" on page 67, "User Variables" on page 92 and "Global and Command Variables" on page 93 for a desciption of how to use these.

You may use capitals or lower-case letters or a combination of both in your static text. The update buffer, however, may automatically convert lower-case letters to capitals when you press Enter. To prevent it from doing this you should enter `SET UPPER OFF` in the command area.

Each line of static text is an independent unit. The MethodManager Update routine does not provide a word-wrap facility of the kind found in word processors. Normally you will not be able to make a line so long that it runs over the right-hand edge of the screen. In the output document, however, lines can sometimes become over-long as a consequence of the expansion of variables or of the application of margin settings. Normally such lines are truncated and the surplus text is not reproduced. The formatter, however, does provide an optional word wrap routine. If this is enabled, it will relocate the surplus text on to one or more new lines below the current one. This is described in "Text Justification" on page 77 and "Word Wrap" on page 78.

You may arrange for blocks of text lines to be kept together if a page break would normally split them. For further information on this, see on the BYPASS and SHOW commands in "The BYPASS Command" on page 70 and "The SHOW Command" on page 70.

You can prevent static text from being output. The BYPASS command described in "The BYPASS Command" on page 70 allows you to treat groups of lines of text and commands as though they are lines of comment, so that text is not reproduced and commands are not executed.

Using Commands

You must begin each command with the appropriate escape characters, normally ?? for a DOCUMENT command or @ for a Manager Products repository command. You should end each command with a semicolon (;) terminator character.

Many commands fit on a single line. Some commands, however, are followed by a number of keywords and values. Consequently these may be too long to fit on a single line. You may continue your command on one or more subsequent lines if you wish. Indeed, you may find the DOCUMENT definition easier to understand if you deliberately spread longer commands over several lines, placing the main command on the first line and each subsequent keyword on a separate line.

A command may not extend beyond 32 lines. You may repeat the appropriate escape character at the beginning of each continuation line if you wish, but this is not strictly necessary; the absence of a terminator at the end of the previous line is sufficient to indicate that the current line is a continuation of the command.

You should ensure, however, that the terminator marking the end of the command does *not* fall in the first column, that is, at the start of a line. You may, however, terminate a command with a line consisting of a space in the first column followed by the semicolon terminator in the second column.

Note:

Some DOCUMENT commands, such as ??ARRAY, are followed by a block of data. Both the command and the block of data must be terminated by a semicolon terminator.

If you do not include a terminator within 32 lines of the start of a command, MethodManager will treat your command as a single-line command and will execute only its first line. Subsequent lines will be treated as static text.

Document Assembly Commands

Document assembly commands allow you to incorporate the whole or part of other repository members into your finished document. The generated output is incorporated at assembly time and so reflects the contents of those members at assembly time and not necessarily when the DOCUMENT definition is created or when the finished document is printed. Consequently, the same DOCUMENT definition assembled at different times may produce documents that are substantially different. If you wish, you can arrange for editorial changes to be applied automatically to the material that is incorporated into your document, as described in "User-defined Editing Routines" on page 52.

Document assembly commands are prefixed by an escape sequence (the default is ??) starting in column 1. You must enter the command itself immediately after the escape characters, without any intervening spaces. You must use upper-case characters and you should terminate the command with a terminator semicolon (;).

Document assembly commands are peculiar to DOCUMENT members. They cannot be executed in any other environment. The following DOCUMENT assembly commands are available:

- INCLUDE, EXTRACT and BODY all incorporate matter from the repository into your document. INCLUDE (described in "The INCLUDE Command" on page 43) incorporates an entire DOCUMENT member. EXTRACT (described in "The EXTRACT Command" on page 49) incorporates selected clauses or attributes from members of any type. BODY (described in "The BODY Command" on page 53) also incorporates selected clauses or attributes from members of any type and allows considerable flexibility in the way in which the extracted matter is formatted.
- CFPATH, CBPATH, CUPATH and PATH are all used to set up reference paths which the EXTRACT or BODY command will follow. They are described in "The CFPATH Command" on page 61 through "The PATH Command" on page 63.
- PARAMETER, ARRAY and FUNCTION all relate to user-defined variables or functions. PARAMETER (described in "The PARAMETER Command" on page 67) sets up variables in your DOCUMENT to which values are assigned at assembly time. ARRAY (described in "The ARRAY Command" on page 68) sets up an array whose values can be accessed from user-defined executive routines. FUNCTION (described in "The FUNCTION Command" on page 69) allows you to define an executive routine which will be accessed by the BODY command.
- BYPASS and SHOW allow the processing of portions of your document to be modified in various ways. BYPASS (described in "The BYPASS Command" on page 70) causes a selected portion of the document to be treated as comment. SHOW (described in "The SHOW Command" on page 70) causes a selected portion of the document to appear in the output, even if it includes comments or commands which do not normally appear.
- ESCAPE (described in "The ESCAPE Command" on page 71) allows you to define a different escape sequence for document commands.

The INCLUDE Command

Use the INCLUDE command if you wish to reproduce another whole DOCUMENT member from your repository as a part of the document that you are creating. DOCUMENTs referenced within documents in this way are called *sub-documents*, although they may be complete DOCUMENTs in their own right.

Its syntax is as follows:

```
>>--- ??INCLUDE MEMBER member-name ---------------------------------------->
                                  |--DECOMPOSE --+-YES-+--|
                                                 |-NO-|
>--------------------------------------------------------------------------->
   |--NUMBERING --+-YES-+--|   |--HEADER --+-header-text-+--|
                |-NO-|                     |-SUPPRESS-----|
>--------------------------------------------------------------------------->
   |--PATH --+-NO-+--|
           |-YES-------------------------------------------|
                                |--MENU --+-YES-+--|
                                        |-NO-|
>--------------------------------------------------------------------------->
   |--EDIT0 exec-name--|   |--EDIT1 exec-name--|
>--------------------------------------------------------------------------->
   |--EDIT2 exec-name--|   |--LOCK --+-YES-+--|
                                     |-NO-|
>--------------------------------------------------------------------------->
   |--------------------<<<<<<<<<<<<<<<<<<<<--------------------|
   |--TRANSLATE "old-char new-char"-----------------------------|
>--------------------------------------------------------------------------->
   |--------------------<<<<<<<<<--------------------|   ;   >>
   |--PASSING "parameter"-----------------------------|
```

For example, if you include in the CONTENTS attribute of your DOCUMENT the following line:

```
??INCLUDE MEMBER DC-002;
```

then, when your document is assembled, the DOCUMENT member named DC-002 in your repository will be assembled into your document at the position of the command. By default its NOTE attribute will be used as a heading and will be numbered in sequence with the other headings in your document. The static text in the CONTENTS clause of the sub-document will be reproduced and DOCUMENT commands in it will be executed.

By default, its TITLE attribute will be used as an entry in the table of contents.

Besides the name of the member to be included, the INCLUDE command may take a number of additional keywords which control the way in which the sub-document is assembled.

Sub-document Title

By default, when you assemble a sub-document into your document using the INCLUDE command, the sub-document's title, as defined in its NOTE attribute, becomes a numbered heading in your document. MethodManager automatically ensures that the number sequence is correctly maintained throughout your document's headings.

You may choose to suppress the numbering of the title. To do this append NUMBERING NO to the INCLUDE command. For example, when the line:

```
??INCLUDE MEMBER DC-002 NUMBERING NO;
```

is executed, the sub-document defined in member DC-002 will be assembled into your document as described previously, but its title will be reproduced as a heading without a number.

You may choose not to display the included sub-document's title. For example, if the sub-document's only purpose is to define a standard frame which controls the layout of your document, its title is probably irrelevant in the output document. To suppress its title in the output document, append HEADER SUPPRESS to the INCLUDE command.

So, when the command:

```
??INCLUDE MEMBER DC-FRAME1
   NUMBERING NO
   HEADER SUPPRESS;
```

is executed, the sub-document defined in member DC-FRAME1 will be assembled into your document, but its title will not be used as a header.

You may prefer to replace the sub-document's title with a new heading. To do this, append:

```
HEADER header-text
```

to the INCLUDE command, where *header-text* represents the string of text which you wish to use in place of the original title. If the new heading contains more than one word, you should enclose the whole heading in double quotes.

For example, when the command:

```
??INCLUDE MEMBER DC-002
   HEADER "Updated Files" ;
```

is executed, the sub-document defined in member DC-002 will be assembled into your document, but its title will be replaced with the text string "Updated Files".

Commands in the Sub-document

Normally, when the INCLUDE command in your document is processed, any DOCUMENT commands in the sub-document will be executed. The execution of commands in a sub- document is called "decomposing". If the sub-document itself contains an INCLUDE command referring to a further sub-document, this too will be assembled into your document at the position of the command in the sub-document. The number allocated to its heading will reflect the reference path. For example, the third INCLUDE command in the first sub-document will be allocated the number 1.3.

You may, however, prevent the decomposition of sub-documents, by appending DECOMPOSE NO to the INCLUDE command. For example, when the line:

```
??INCLUDE MEMBER DC-002 DECOMPOSE NO;
```

is executed, the text in the sub-document DC-002 will be incorporated into your document, but INCLUDE, EXTRACT and BODY commands in it will be ignored.

You may restrict the scope of decomposition to members which conform to a certain pattern of reference. For example, you could restrict decomposition to two levels of reference of DOCUMENT members by issuing the command CFPATH DOCUMENT DOCUMENT before the INCLUDE command and appending PATH YES to the INCLUDE command. You will find further information on the CFPATH command later, in "The CFPATH Command" on page 61.

If you use the PATH YES attribute, as described above, you may further restrict the decomposition to reproducing the titles, as stored in the NOTE attributes, of the DOCUMENT members concerned, but not their contents. To do this append MENU NO to the INCLUDE command.

Avoiding Recursive Loops

If the DOCUMENTs and other members in your repository refer to each other, as is likely, there is a danger in DOCUMENT assembly of entering an endless loop. For example, if your DOCUMENT contains an INCLUDE command referring to a sub-document which itself contains an INCLUDE or EXTRACT command referring back to your document, then at assembly time the system could in some circumstances enter an endless loop.

To prevent this, the documentation facility contains a lock routine which by default is enabled. This prevents any DOCUMENT which has already been processed from being decomposed again during the same document assembly. Sometimes, however, you may need to refer to the same sub-document more than once. To permit this, you may disable the lock by appending LOCK NO to your INCLUDE command. You should use this facility with care because of the danger of recursive loops.

User-defined Editing Routines

You may apply one or more user-defined editing routines to amend the text and commands in the sub-document that you are including. You must set up the editing routines to be used in advance as executive routines in the MP-AID. You should refer to the Manager Products technical documentation for further details of executive routines.

There are three stages in the INCLUDE process at which an editing process of this kind may be applied. The choice of stage is determined by your choice of parameter, EDIT0, EDIT1 or EDIT2:

EDIT0 applies the edit to the extracted NOTE attribute. Consequently, only the contents of the NOTE attribute, usually the sub-document's title, will be affected. Since the NOTE attribute is usually used as a heading in the document being assembled, EDIT0 may be used to format the heading in some way, such as converting it to upper case or truncating it to a certain number of words or characters.

EDIT1 applies the edit to the extracted CONTENTS attribute, but before any commands in it are executed. The edit may therefore change text and commands within the sub-document's CONTENTS, as the example below shows.

EDIT2 applies the edit after decomposition of the CONTENTS attribute, that is, after any commands in it have been executed. The edit may therefore change text that has been incorporated into your document from other repository members as a result of the execution of commands in the sub-document being included. For example, it could search for certain words or characters and replace them with preferred or amended values.

The following example of an EDIT1 (Figure 22 on page 47), the executive routine named EXPAND listed below is installed on the MP-AID:

```
EXECUTIVE
CONTENTS
mpxx 1iteral=#
/* TRACE L0GIC ASSIGNS RESULTS
start = 0
MPR #EXTRACT COMMAND 'LIST KEPT IN EXPAND DETAILS; '#
MPR #GENERATE 'NME';#
DO NME()
   NAME = NME(FDO(ARRAY))
   CALL PROCESS-NAME

EXIT
-PROCESS NAME
do document_body()
   start = start+1
   ct = fdo(array)
   new_array(start) = -repstr(document_body(ct),#?????#,name)
end
return
```

Figure 22

The routine searches the CONTENTS clause of the document being decomposed and for each occurrence of the string ????? it repeats the CONTENTS, replacing ????? with the name of the next member in the KEPT-DATA list named EXPAND.

The CONTENTS of the document DC-EXPAND are as follows:

```
@NOPRINT KEEP IN EXPAND LIST ONL EN-C
??INCLUDE MEMBER DC-EXPAND2 HEADER SUPPRESS EDIT1 'EXPAND' ;
```

The CONTENTS of the subdocument DC-EXPAND2 are as follows:

```
??CFPATH ENTITY ITEM ;
??BODY MEMBER ????? ;
```

```
═══════════════════════════
× ENTITY
- - - - - - - - - - - - - - - - - - - - - - - -
×                        ITEM
- - - - - - - - - - - - - - - - - - - - - - - -
 ;
```

The use of the BODY command is explained later, in "The BODY Command" on page 53.

When DC-EXPAND is assembled, first a KEPT-DATA list is created containing the names of five ENTITY members in the repository. Then the sub-document DC-EXPAND2 is incorporated into the document and the executive routine EXPAND is called. This replaces the string ????? which occurs in a BODY command in the included sub-document contents with the first name on the KEPT-DATA list, EN-CUST. Next the commands in the sub-document are executed and so the BODY command is applied to the member whose name has been substituted. The process is repeated for the remaining members on the KEPT-DATA list. The output from the assembled document is as follows:

```
═══════════════════════════
× EN-CUST
- - - - - - - - - - - - - - - - - - - - - - - -
×
- - - - - - - - - - - - - - - - - - - - - - - -

═══════════════════════════
×  EN-CUSTOMER
- - - - - - - - - - - - - - - - - - - - - - - -
×
- - - - - - - - - - - - - - - - - - - - - - - -

═══════════════════════════
×  EN-CUSTOMER-BACK-ORDER
- - - - - - - - - - - - - - - - - - - - - - - -
×
- - - - - - - - - - - - - - - - - - - - - - - -

═══════════════════════════
× EN-CUSTOMER-ORDER
- - - - - - - - - - - - - - - - - - - - - - - -
×          IT-CUSTOMER-ORDER-DATE
- - - - - - - - - - - - - - - - - - - - - - - -
×          IT-CUSTOMER-NUMBER1
- - - - - - - - - - - - - - - - - - - - - - - -
×          IT-CUSTOMER-ORDER-STATUS
- - - - - - - - - - - - - - - - - - - - - - - -
×          IT-CUSTOMER-ORDER-VALUE
- - - - - - - - - - - - - - - - - - - - - - - -
×          IT-CUSTOMER-NUMBER
- - - - - - - - - - - - - - - - - - - - - - - -

═══════════════════════════
× EN-CUSTOMER-ORDER-HISTORY
- - - - - - - - - - - - - - - - - - - - - - - -
×
- - - - - - - - - - - - - - - - - - - - - - - -
 ;
```

Translation of Characters

Sometimes the sub-document may contain characters which you wish to change in your document. You can do this by appending TRANSLATE '*old-char new-char*' to your INCLUDE command, where *old-char* and *new char* represent, respectively, the character which you wish to replace and the character you wish to insert in its place. You may specify more man one pair of translation characters. The whole chain of characters following the TRANSLATE attribute must be enclosed in single quotes. If you wish to specify a space as a translation character, you must enclose it between double quotes.

For example, when the line:

```
??INCLUDE MEMBER DC-002 TRANSLATE  '_- */';
```

is executed, the text from the sub-document DC-002 will be incorporated into the DOCUMENT you have defined, but any underscore characters in the sub-document will be replaced with hyphens and any asterisks will be replaced by obliques.

Passing Values into Variables

Both text lines and command lines in DOCUMENT contents may contain variables, to which values, that is strings of text, are assigned when the document is assembled. In the finished document the values appear in place of the variable names. If you wish to use variables in your document, you must set them up first using the PARAMETER command described in "The PARAMETER Command" on page 67.

If your document contains an INCLUDE command referring to a sub-document that contains variables, you may assign values to those variables at assembly time by appending to it the PASSING attribute followed by as many values, separated by spaces, as there are variables in the sub-document. Values may not contain the space character. When the sub-document is reproduced in your document, the values you passed with the INCLUDE command will appear in place of the variable names. If you wish to leave a variable null, you can pass a null value to it as two double quotes without any intervening characters, thus: "".

The order of the values following PASSING must correspond to the order in which the variables were set up by the PARAMETER command in the sub-document. If you append more values to the PASSING parameter than the number of variables in the sub-document, the surplus values are ignored. If you append fewer values than the number of variables in the document, the surplus variables will be null.

The EXTRACT Command

Use the EXTRACT command if you wish to incorporate into your document text from a selected attribute or clause from one or more members of any type in your repository. You can only extract textual attributes in this way.

Its syntax is as follows:

```
>>--- ??EXTRACT MEMBER member-name ---------------------------------------->
>--- ATTRIBUTE attribute-name ---+-------------------------------------+--->
                                 |  +-- YES --+                        |
                                 +--DECOMPOSE-+-- NO --+               |
>--+-------------------------------------------------------------------+--->
   |  +-- YES --+                  +-- header-text --+                |
   +--NUMBERING-+-- NO --+         +--HEADER-----------+-- SUPPRESS --+
>--+-------------------------------------------------------------------+--->
   |  +-- HEADING attribute --+  +-- NO --+                           |
   +--PATH-------------------+--+-- YES ---+-------------------------+
                                           |  +-- YES --+            |
                                           +--MENU------+-- NO ------+
>--+-------------------------------------------------------------------+--->
   |  +-- EDIT0 exec-name --+  +-- EDIT1 exec-name --+                |
>--+-------------------------------------------------------------------+--->
   |  +-- EDIT2 exec-name --+  +-- LOCK---+-- YES --+                 |
                                           +-- NO ---+                |
>--+-------------------------------------------------------------------+--->
   |                 <<<
```

is executed, the DESCRIPTION attribute of the member TB- CUSTOMER will be incorporated into the output document and the heading 'Description of Customer Table' will be attached to it.

Alternatively, you may use as a header an attribute or clause of your choice from the member being extracted. To do this append:

```
HEADING attribute
```

to the EXTRACT command, where *attribute* is the name of a clause in the member being extracted. Be careful to observe the distinction between HEADER and HEADING. If both HEADER and HEADING are included in the same command, the HEADER specification overrides the HEADING entry which is ignored.

If a HEADING keyword specifies the same attribute as the EXTRACT command itself, only the first line of the attribute is used as the heading, unless it is blank. If it is blank, the first non-blank line is used instead.

Headings or headers of text extracted using the EXTRACT command are not normally numbered. (This stands in contrast with headings to sub-documents extracted using the INCLUDE command, which are numbered by default.)

If you wish your attached header to be automatically numbered in sequence with other headings, include NUMBERING YES before the header text. For example,

```
??EXTRACT MEMBER TB-CUSTOMER
   ATTRIBUTE DESCRIPTION
   NUMBERING YES
   HEADER "Description of Customer Table" ;
```

would create the header described previously, but numbered in sequence with other headings in the document.

Document Commands in the Extracted Material

If you use the EXTRACT command to extract the CONTENTS attribute from a DOCUMENT member, by default only the static text is reproduced; DOCUMENT commands in it are ignored. This stands in contrast with the INCLUDE command which, by default, decomposes the sub-document.

You may force the EXTRACT command to decompose the extracted material by appending DECOMPOSE YES to the command.

For example, when the command:

```
??EXTRACT MEMBER DC-004
   ATTRIBUTE CONTENTS
   DECOMPOSE YES;
```

is executed, the contents of the DOCUMENT member will be incorporated into the output document and any commands in the contents will be executed.

Extracting from More Than one Member

You may use one EXTRACT command to extract clauses from several members by setting PATH YES within the EXTRACT command. First you must define the reference path to be followed using the CFPATH command described later in "The CFPATH Command" on page 61.

For example, the following pair of commands:

```
??CFPATH PROGRAM MODULE MODULE;
??EXTRACT MEMBER PG-PROCESS-EMPL-PAY
   ATTRIBUTE DESCRIPTION
   PATH YES;
```

will extract the DESCRIPTION attribute not only from the PROGRAM member named PG-PROCESS-EMPL-PAY, but also from the MODULE members to which it refers and in turn from further MODULE members to which the first MODULEs refer.

If you set a path in this way to extract the CONTENTS attributes of DOCUMENT members, you may restrict their decomposition to reproducing only their titles by appending MENU NO to the EXTRACT command.

Avoiding Recursive Loops

If the members in your repository refer to each other, as is likely, there is a danger in DOCUMENT assembly of entering an endless loop. A lock facility is provided which normally prevents this. In the EXTRACT command this works in exactly the same way as in the INCLUDE command. Refer to "Avoiding Recursive Loops" on page 46 for further information.

User-defined Editing Routines

You may apply user-defined editing routines to the attribute that you are extracting. The facility works in a similar way to the corresponding parameter of the INCLUDE command. Refer to "User-defined Editing Routines" on page 46 for further information.

Translation of Characters

If the material being extracted contains chacacters which you wish to change in your document, you may append TRANSLATE *old-char new-char* to your EXTRACT command. The facility works in exactly the same way as in the INCLUDE command, as described in "Translation of Characters" on page 49.

For example, if you wish to remove the highlight (!) and end-of-field (%) characters from the CONTENTS attribute of INFOBANK-PANEL KT21-00, you could use the following command:

```
??EXTRACT MEMBER KT-2100
 ATTRIBUTE CONTENTS
 TRANSLATE '!" " %" "';
```

When the command is executed, the highlight (!) characters and end-of-field (%) characters in the CONTENTS attribute of INFOBANK-PANEL KT-2100 will be changed to spaces, effectively removing them from the output document.

Passing Parameters

If you EXTRACT the CONTENTS clause of a member that contains variables, you may assign values to these by appending PASSING followed by as many values, separated by spaces, as there are variables in the sub-document. When the sub-document is reproduced in your document, the values you passed with the INCLUDE command will appear in place of the variable names. The facility works in the same way as in the INCLUDE command described in "Passing Values into Variables" on page 49.

The BODY Command

The BODY command is similar to the EXTRACT command in that it extracts data from members of various types in your repository, but it also allows access to user-defined executive routines. Moreover, it gives you precise control over the way in which the retrieved data is presented. Effectively, you provide a template into which the retrieved data, which may consist of the names of selected repository members or the contents of attributes, is inserted. You could use it, for example, to build up a graphical representation of a reference trail. For this reason, it is particularly useful if you wish to compile a report or a manual describing the relationships between members in your repository.

The BODY command follows a reference path. You must do one of the following:

- Specify the name of a KEPT-DATA list naming the members to be analyzed, or
- Before using the BODY command, define a reference path for it using the CFPATH, CBPATH, CUPATH or PATH commands described in "The CFPATH Command" on page 61 through "The PATH Command" on page 63 and, within the command itself; specify the name of the member on which analysis is to begin,
- Both specify the KEPT-DATA list and set up a path as described above. The path is then followed starting with each member named on the list in turn.

The BODY command may be followed by a number of keywords which determine how it operates. It is followed also by a number of definition, redefinition and text lines which control the format in which the retrieved data is presented.

Its syntax is as follows:

```
>>--- ??BODY ---+-------------------+-------------------+-------------------->
                 |  EOF eof-char     |  COMMENT com-char |
>---+-------------------+-------------------+-------------------+-------------->
    | MEMBER member-name |                   | OUTPUT ---+-YES-+ |
    | KEPT kept-name     |                   |           +-NO  + |
                                               |           +-SORT+ |
>---+-------------------+-------------------+-------------------+-------------->
    | SCOPY ---+-YES-+  |                   | CYCLE ---+-YES-+  |
             | +-NO  +  |                   |          +-NO  +  |
                                               |          +-NODE+  |
>---+-------------------+-------------------+-------------------+-------------->
    | DATA ---+-NEW-+   |                   | SUPPRESS-+-YES-+  |
           | +-OLD+   |                   |          +-NO  +  |
                                               |          +-NODE+  |
>---+-------------------+-------------------+-------------------+-------------->
    | TEXTID1 str1      |                   | TEXTID2 str2      | ; |
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
>---+-------------------+-------------------+-------------------+-------------->
    | definition line   |                   | redefinition line |  ; >><
    | text line                                                 |
```

Note:

There are *two* semicolon terminators. The one at the end of the command proper and before the text and definition lines is optional. Its presence does not normally affect the execution of the command, but many users prefer to include it as it makes the command easier to follow. The semicolon at the end of the text and definition lines is essential and must be on a separate line from the text, definition and redefinition lines. The keywords and the use of the text and definition lines are explained below.

Member/Kept

These are mutually exclusive options. You should specify either the name of the individual member on which you wish reference path analysis to begin or the name of a KEPT-DATA list.

End-of-field Character

EOF is followed by the character you wish to use in definition lines (explained in "Definition, Redefinition and Text Lines" on page 55) to indicate the end-of-field in the output from the BODY command. The hash (#) character is commonly used. Any character could be used, but you should avoid using the asterisk (*) which has a special meaning in definition lines.

The distance of the EOF character's position from the start of the preceding variable (such as a member type or attribute name) is used as the length of the field that will be available for the data, such as a member name, that will replace that variable. The length of field is especially important when the data returned by the BODY command is required to fit into the limited space available in a diagrammatic template. If the data is too long for the space available, it will be truncated and, in some circumstances, may be overwritten by other returned data.

Definition, Redefinition and Text Lines

The BODY command with its various keywords is followed by a number of definition, redefinition and text lines. These control the format in which the output from the command is presented. All lines following a BODY command until the next semicolon terminator are regarded as text, definition or redefinition lines unless they are preceded by the escape character assigned for comments in the scope of the BODY command (described in "Comment" on page 60). Such comment lines are ignored.

- Text lines consist entirely of static text. They are copied into the output document unchanged. A text line may be terminated by a TextID label (expained in "Controlling the Output" on page 55 and "Using and Changing the TextID Labels" on page 59). The TextID label itself is not reproduced but controls the circumstances in which the text line is reproduced.
- Definition lines are text lines which contain variables such as names of member types or attributes or the wildcard character (*) which represents any member type. A definition line is reproduced whenever a match is made between a variable it contains and a member on the path being followed. When a definition line is reproduced, the variables it contains are replaced by the names or types of the matching members or the contents of the specified attribute.
- Redefinition lines are similar to definition lines but are distinguished by the suffix R-. A redefinition line must follow immediately after a definition line and it controls the same line of output. Its output is overlaid over that of the preceding definition line so that it redefines it.

The reason for using a redefinition line is as follows. Definition lines control both the content and format of the data retrieved by the BODY command. Since sometimes the variables and their control data (described in "Controlling the Output" on page 55) take more space than the retrieved data that replaces them, there may be insufficient room to fit all the required variables, control data and static text into one definition line. By sharing the variables and control data between a definition line and a redefinition line, you should be able to accommodate them all and obtain the output in the required format.

Controlling the Output

Definition and redefinition lines allow you to control precisely the output obtained from members found on the reference path.

If you specify a member type, in the output document this is replaced by the name of the matching member found. If you specify a member name, this is replaced by its member type. If no match is made, a blank line is usually output, but such blank lines can be suppressed. "Other Keywords" on page 60 explains how to do this.

If you specify the name of an attribute, its contents are reproduced, in whole or in part as explained below. You may also use the terms USED-BYS, REFERENCES and EXTERNAL which behave in the same way as attributes.

You may control the quantity of text that is output by suffixing the attribute name with brackets containing three or four terms, separated by commas. These terms themselves are not reproduced. These terms are:

- The member name or type or the wildcard (*) character if any member type is to be returned
- The number of the first line in the attribute to be reproduced
- The number of the last line in the attribute to be reproduced or ALL if the whole attribute is to be reproduced irrespective of its length
- The fourth term is used when the data is being extracted not directly from a repository member, but from an executive routine in the MP-AID or when a user-defined relationship (UDR) is being accessed or when a Procedures Language expression is used. The fourth term is the name of the routine being called.

If you omit the second and third terms, they will be regarded as containing the value 1 so that only the first line of the extracted data will be reproduced. For example, the following definition line:

```
NOTE(ENTITY)
```

will reproduce the first line of the NOTE attribute of each ENTITY member found on the path.

The following definition line:

```
NOTE(ENTITY,1,5)
```

will reproduce the first five lines of the NOTE attributes of all ENTITY members found.

If you are following a KEPT-DATA list, you may use the wildcard (*) character to represent any member type in a definition or redefinition line. For example, the following definition line:

```
NOTE(*,1,ALL)
```

will reproduce the entire text of the NOTE attributes of all members of any type found on the KEPT-DATA list.

You may use the wildcard more than once in a line, as in the following definition line (which uses # as the EOF character):

```
*        #REFERENCES(*,1, ALL)
```

This will return, starting at the current left margin, the name of each member on the KEPT-DATA list and, after the position of the EOF character, the names of all the members of any type to which the previously mentioned member refers. If, for example, the KEPT-DATA list contained just one DIAGRAM member which referred to four ITEM members, the output might be as follows:

```
DG-FI-PERS-DETAILS                    IT-COMP-NAME
                                      IT-POSITION
                                      IT-ADDR1
                                      IT-POSTCODE
```

Since REFERENCES behaves as though it were an attribute, you can control the number of lines of output generated from it. For example, if the definition line had read:

```
*                                #REFERENCES(*)
```

the output would have been restricted to the first line of referenced members, as follows:

```
DG-FI-PERS-DETAILS               IT-COMP-NAME
```

Similarly:

```
*                                #USED-BYS(*,1,ALL)
```

will return, on the left, the names of the members in the path and, after the position of the EOF character, the names of members of any type which refer to the previously specified member.

The BODY command can also return data processed by user-defined executive routines. One way to do this is to store your executive routine as a USER member on the MP-AID and to call it from the BODY command using the keyword EXTERNAL as though it were an attribute whose contents were being extracted. An example of this is given in "The PATH Command" on page 63.

You can also define temporary executive routines within DOCUMENT definitions using the FUNCTION command. The function is called from within the BODY command using the keyword EXEC and its name as fourth data control codes. You will find further details and an example in "The FUNCTION Command" on page 69.

You may use Procedures Language expressions as the fourth term of the control data. These perform simple edits on the data returned. For example:

```
IN(*,,,EXPRESSION SUBSTR(IN_NAME,1,5)
```

will return just the first five characters of the IN clause of each member. Similarly,

```
DESCRIPTION(*,,,EXPRESSION UPPER(DESCRIPTION))
```

will return the DESCRIPTION clause of each member, translated into upper-case.

If you have used user-defined relationships, the BODY command can return their names by referencing the variable UDR_NAME as the fourth term of control data. For example:

```
@NOPRINT KEEP IN TB LIST ONLY TB-C;
??BODY KEPT IN TB EOF # ;
═══════════════════════════════════════════════════
| Member name                                     |
|-------------------------------------------      |
| *                        #DESRIPTION(*,) #      |
| First UDR  FIRST(*,,,VARIABLE UDR_NAME)         |
| Second UDR SECOND(*,,,VARIABLE UDR_NAME)        |
═══════════════════════════════════════════════════

 ;
```

On assembly the ouput from this document is as follows:

```
| Member name                                    |
|-----------------------------------------       |
| TB-CUST-COPY              CUST COPY            |
| First UDR  AA                                  |
| Second UDR BB                                  |
```

```
| Member name                                    |
|-----------------------------------------       |
| TB-CUSTOMER                                    |
| First UDR                                      |
| Second UDR                                     |
```

Note:

The same definition line may be output repeatedly. If more than one match is made on the reference path or KEPT-DATA list, the definition line containing the matching reference is reproduced once for each match.

An unlabeled text line following the BODY command but preceding the first definition line is reproduced once, at the start of the output from the command. (If a KEPT-DATA list is being followed, the line will be output once for each member on the list.) Unlabeled text lines between definition lines are assumed to be related to the previous definition line and are repeated following each appearance of that definition line.

You may, however, link a text line to a specific member type including one specified in a subsequent definition line; in this way the linked text line may be repeated preceding each appearance of the relevant definition line.

To link a text line, you must append a TextID label to it. This label most often takes the form:

++membertype

where *membertype* is the variable in the relevant definition line.

Note:

The label begins with a space; this is essential.

TextID-labeled text lines are useful if you wish to create an elaborate graphical representation of a reference trace. The following example shows how labels can be used to force the repetition of text lines which precede the definition lines for the object types to which they relate:

```
??CFPATH DIAGRAM ITEM ;
??BODY MEMBER DG-FI-PERS-DETAILS EOF#;
 -----------------------------
|DIAGRAM                     #|
 -----------------------------
              |
              |++ITEM
              |    ------------------------- ++ITEM
              |--->| ITEM                  #|
              |    -------------------------
;
```

When assembled this might produce the following graphical representation of a reference trace:

```
 ------------------------
| DG-FI-PERS-DETAILS     |
 ------------------------
            |
            |
            |     ---------------------
            |--->| IT-COMP-NAME        |
            |     ---------------------
            |
            |
            |     ---------------------
            |--->| IT-POSITION         |
            |     ---------------------
            |
            |
            |     ---------------------
            |--->| IT-ADDR1            |
            |     ---------------------
            |
            |
            |     ---------------------
            |--->| IT-POSTCODE         |
            |     ---------------------
```

Using and Changing the TextID Labels

You may change the string of characters used for TextID labels using the TEXTID1 and TEXTID2 keywords in the BODY command. This may be useful if you wish to use the + character elsewhere in your text or definition lines.

Attaching the TextID1 label (by default "++membertype") to a line of text ensures that that line of text (but not the TextID label itself) appears in the output whenever a change of control occurs at the same level or a higher level. Since a change of control is said to occur whenever the analysis of one member is completed and the analysis of another begins, appending the TextID1 label to a text line ensures that that line is reproduced whenever a member of the type specified in the label is found on the path or KEPT-DATA list.

A TextID1 label allows a text line to be reproduced preceding the definition line that relates to the same member type. This facility is especially useful when constructing graphical representations of reference traces, as the above example demonstrates.

Attaching the TextID2 label (by default "+membertype") to a line of text ensures that that line of text appears in the output when if a change of control occurs at a higher level. This means that the line will be reproduced only when the analysis of a member at a higher level than that specified ceases and the analysis of another member at a higher level begins. For example, a line labeled

```
+ITEM
```

will not be reproduced when the analysis of one ITEM member ends and another begins, but will be reproduced if the analysis of, say, the GROUP member containing the items ends and analysis of another GROUP begins.

Comment

COMMENT allows you to define a new escape character that will identify commentary lines amongst the text, definition and redefinition lines.

Elsewhere in the CONTENTS of DOCUMENT members, the default escape character for commentary lines is the asterisk (*). You should not use the * character itself for this purpose within the scope of the BODY command, because in this environment it is used as a 'wildcard' as described in "Controlling the Output" on page 55.

You may include commentary lines throughout the text, definition and redefinition lines to explain their purpose. They are particularly helpful for other users who may wish to use your document definitions. Comments have no effect on the execution of the command or the output document. The comment line must start in the first column. For further information about commentary lines see later in "Commentary Lines" on page 93.

Other Keywords

The other keywords in the BODY command are as follows:

Keyword	Description	
OUTPUT	Must be followed by YES, NO or SORT.	
	YES	gives normal output of lines.
	NO	suppresses normal output, but sends it instead to an array called PRINT_OUTPUT which can be processed subsequently by Procedures Language directives.
	SORT	enables output, but causes the lines being output by the command to be sorted into ascending order of EBCDIC code.

Keyword	Description
SCOPY	Controls the appearance of redefinition lines, which are explained in "Definition, Redefinition and Text Lines" on page 55. Its inclusion in a BODY command forces redefinition lines to appear in the output as separate lines following the definition lines to which they relate. It is a useful tool in sorting out problems, where the output obtained is not quite as expected. SCOPY must be followed by YES or NO. YES — Forces the entire redefinition line, including static text, to be reproduced. NO — Causes only the variable parts of the redefinition line to appear.
CYCLE	Must be followed by YES, NO or NODE. It determines whether a member can be decomposed more than once. NO — Means that a member can be decomposed only once in each path analysis. YES — Means that a member can decomposed more than once. NODE — The default; allows a member to be decomposed only once per node, that is once per member found on the reference path being traced.

SUPPRESS	Must be followed by YES, NO or NODE. It allows you to avoid creating empty lines of output. For example, the following lines:
	`??BODY KEPT ABC SUPPRESS YES;`
	`PROGRAM: NOTE(*)`
	`;`
	would create a line of output for every member in the KEPT- DATA list, but for those members which do not have a NOTE clause the line would be effectively empty (it would read simply PROGRAM:) if the SUPPRESS YES keyword were omitted. With SUPPRESS YES included these uninformative lines are suppressed. The NODE alternative allows one such empty line per node, that is per member on the reference path being traced.

DATA	is followed by NEW or OLD. If you are using complex path commands, the execution of the BODY command may take an appreciable time, because of the time taken to compile the list of members to be analyzed. If you have more than one BODY command following the same path, you can save time by specifying DATA OLD in the second BODY command. It will then use the existing list, instead of compiling a new identical one.

The CFPATH Command

The CFPATH command is used in conjunction with the EXTRACT command described in "The EXTRACT Command" on page 49 and the BODY command described in "The BODY Command" on page 53. CFPATH allows you to specify a hierarchical reference path which a subsequent EXTRACT or BODY command will follow.

The CFPATH command sets up a path of *forward* reference, that is, one in which each specified member type refers to the next specified type.

The syntax is:

```
>>──────??CFPATH──── <<<<<<<<<<< membertype ──;──────────────────────────><
```

Any appropriate number and sequence of member types may be specified, the types being separated by spaces. The member on which the reference trace begins is normally specified in the BODY or EXTRACT command or in a KEPT-DATA list. For example, the following command:

```
??CFPATH PROGRAM MODULE ;
```

will set a two-deep reference path of PROGRAMs and the MODULEs to which the PROGRAMs refer.

The following command:

```
??CFPATH ENTITY ENTITY ENTITY ENTITY
    ENTITY ;
```

will set up a five-deep reference trail of ENTITIES. The EXTRACT or BODY command will seek out only ENTITIES and other ENTITIES to which they refer and further ENTITIES to which these refer, and so on to a depth of five levels of reference.

For clarity, when the same member type is specified at more than one hierarchical level in the path, as in the above example, the level number, if it is greater than 1, is appended to the member type automatically and this is reflected in output from, for example, the BODY command. For example, an occurrence of ENTITY in the highest level will be represented as ENTITY, in the second level as ENTITY_2 and in the third level as ENTITY_3.

Alternatively, the first item following the CFPATH command may be a specific member name from which the path is to start. If so, the output from the BODY command relating to this first item will be the member type, and not the member name.

When a CFPATH command has been included in the DOCUMENT, an EXTRACT command with PATH YES set, or a BODY command, will search each example of the first member type specified for the path for references to the second member type specified. If more than two levels have been set, the search will continue for members which satisfy the trail specification.

The EXTRACT command will search for the attribute type named in the command and all those found will be processed according to the options specified in the command. The BODY command will retrieve the specified data and format it following the definition lines given.

When a CFPATH, CBPATH, CUPATH or PATH command has been included in a DOCUMENT, the path it has set remains in force and may be used by other EXTRACT and BODY commands until superseded by another CFPATH, CBPATH, CUPATH or PATH command.

The CBPATH Command

The CBPATH command is used in exactly the same way as the CFPATH command described above, but it sets up a path of *backward* reference. That is, from the first object or object type, it will seek objects of the specified type which refer to it and so on.

The CUPATH Command

The CUPATH command is used in a similar way to the CFPATH and CBPATH commands described above, but the direction of reference is undefined and must be specified explicitly for each level using -> to indicate forward reference and <- to indicate backward reference.

So the following command:

```
??CUPATH TX-CUSTENQ <- PROGRAM -> MODULE;
```

seeks a member named TX-CUSTENQ and then traces those PROGRAM members which refer to it and then those MODULEs to which each PROGRAM found in turn refers.

The PATH Command

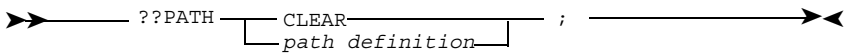
The PATH command is similar to the CFPATH command described above, but is more flexible and allows more complex structures to be created. The command only defines one node, that is, one step on the path at a time and, for this reason, a sequence of PATH commands is needed to define a full path.

Before defining a new path the command:

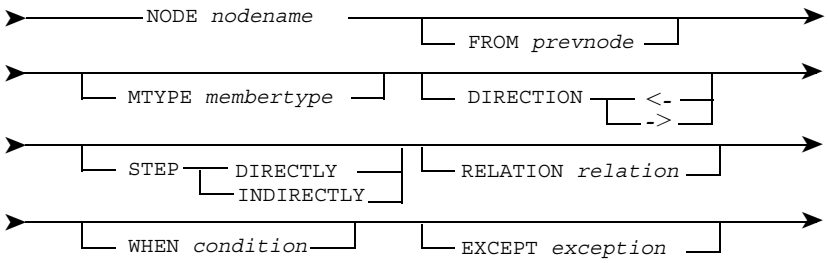
```
??PATH CLEAR;
```

should be issued to clear any previously set path, otherwise the command may attempt to link a new node to a path that was set previously.

The syntax of the PATH command is as follows:



where *path definition* is as follows:



where:

nodename allocates a name to the node being defined

prevnode is the name of the previous node

membertype is the type of member sought

`->` gives forward reference (as in USES) and <- gives backward reference (as in CONSTITUTES) command; the default is forward reference

DIRECTLY/INDIRECTLY controls the reference relationship between the current and the previous node

relation is the type of relationship between the current and previous node

condition or *exception* are a restriction on the relationship or a condition which excludes it, as in the HAS condition of the WHICH command.

The PATH command is most often used in conjunction with the BODY command to create graphical representations of data structures. The following pair of examples analyse the same KEPT-DATA of DB2-TABLES. The first example uses the PATH command to control the reference path followed. In the second example, no path is set, the output from the command being wholely dependent on the definition lines.

The relevant section of the first document is as follows:

```
??PATH CLEAR ' ';
??PATH NODE IT FROM START MTYPE ITEM;
??PATH NODE TBSP FROM START MTYPE DB2-TBSPACE;
??PATH NODE DB FROM TBSP MTYPE DB2-DATABASE;
??PATH NODE CR FROM START MTYPE DB2-USER;
??BODY KEPT TB;
```

```
Table name START
 Creator-Owner ALIAS                  Membername
               ALIAS (CR,1,1)         CR
 Columns       ALIAS                  Membername
               ALIAS (IT,1,1)         IT
   TABLESPACE  Alias ALIAS (TBSP,1,1) Membername TBSP
   TABLESPACE  is in DATABASE DB
;
```

This section of the DOCUMENT assembles as follows:

```
Table name TB-CUST-COPY
 Creator-Owner ALIAS          Membername
                              NEW-ONE
 Columns       ALIAS          Membername

   TABLESPACE  Alias          Membername TBSP
   TABLESPACE  is in DATABASE
```

```
Table name TB-CUSTOMER
 Creator-Owner ALIAS            Membername
               CUSTOMER_NAME    IT-CUSTOMER-NAME
               CUSTNO           IT-CUSTOMER-NUMBER
   TABLESPACE  Alias TSCUST     Member-name TS-CUMR
   TABLESPACE  is in DATABASE
```

```
Table name TB-DSN8210-DEPT
 Creator-Owner ALIAS            Membername
               DEPTNO           IT-DEPTNO
                                IT-DEPTNAME
               MGRNO            IT-MGRNO
                                US-DSN8210
 Columns       ALIAS            Membername
               ADMRDEPT         IT-ADMRDEPT
   TABLESPACE  Alias DSN8S21D   Member-name TS-DSN8
   TABLESPACE  is in DATABASE   DB-DSN8D21A
```

```
Table name TB-DSN8210-EMP
 Creator-Owner ALIAS            Membername
               EMPNO            IT-EMPNO
               FIRSTNME         IT-FIRSTNME
               MIDINIT          IT-MIDINIT
               LASTNAME         IT-LASTNAME
               WORKDEPT         IT-WORKDEPT
               PHONENO          IT-PHONENO
               HIREDATE         IT-HIREDATE
               JOB              IT-JOB
               EDLEVEL          IT-EDLEVEL
               SEX              IT-SEX
               BIRTHDATE        IT-BIRTHDATE
               SALARY           IT-SALARY
               BONUS            IT-BONUS
               COMM             IT-COMM
   TABLESPACE  Alias DSN8S21E   Member-name TS-DSNE
   TABLESPACE  is in DATABASE
```

The second example (Figure 23 on page 66) employs a user-defined executive routine called using the EXTERNAL keyword. The routine, which must be stored on the MP-AID, is as follows:

```
MPXX LITERAL=:
*RETAIN
*TRACE ALL RESULTS COMMANDS
DROP    RETURN_VALUE
COMMAND RETURN_VALUE
START = 1
MPR :DRETRIEVE COUNT CONSTRAINT;:
DO FOR COUNT_CONSTRAINT
   MPR :DRETRIEVE NEXT CONSTRAINT WITH ATTRIBUTES:;
```

```
      DROP FKEY_NAME
      MPR :DRETRIEVE ALL FOREIGN-KEY WITH ATTRIBUTES:;
      MPR :EXTRACT VARIABLE FKEY_NAME ALL:
      MPR :GENERATE 'RETURN VALUE' ARRAYNUM: START;
    START = START+ARRAYHI(:FKEY_NAME:)
   END
```

Figure 23

The significant section of the DOCUMENT is as follows:

```
??PATH CLEAR;
??BODY KEPT TB EOF #;
┌ — — —
```

```
DB2 TABLE COLUMNS                    FOREIGN KEYS
-------------------------------------------------------
TB-DSN8210-DEPT
   IT-DEPTNO                         IT-ADMRDEPT
   IT-DEPTNAME                       IT-MGRNO
   IT-MGRNO
   IT-ADMRDEPT
   -------------------------------------------------------
     CREATOR          TABLESPACE
     US-DSN8210       TS-DSN8
```

```
DB2 TABLE COLUMNS                    FOREIGN KEYS
-------------------------------------------------------
TB-DSN8210-EMP
           IT-EMPNO                  IT-WORKDEPT
           IT-FIRSTNME
           IT-MIDINIT
           IT-LASTNAME
           IT-WORKDEPT
           IT-PHONENO
           IT-HIREDATE
           IT-JOB
           IT-EDLEVEL
           IT-SEX
           IT-BIRTHDATE
           IT-SALARY
           IT-BONUS
           IT-COMM
           ---------------------------------------------
           CREATOR      TABLESPACE
           US-DSN8210   TS-DSNE
```

The PARAMETER Command

Use the PARAMETER command to define variables in your document. The syntax of the command is:

```
>>──────── ??PARAMETER var-name1 var-name2 var-name3... ──────;──────><
```

You may define as many variables as you wish by assigning their names with the PARAMETER command, using one or more spaces to separate the variable names. Any string of characters may be used as a variable name, providing that it begins with an alphabetic character, all alphabetic characters are capitals, and it does not include any spaces.

The values held by these variables consist of strings of text which must not include spaces. When they are first set up using the PARAMETER command, these variables contain null strings.

When a sub-document is reproduced using an INCLUDE or EXTRACT command, values may be assigned to the variables it contains using the PASSING keyword described in "Passing Values into Variables" on page 49. The order in which the values are passed in the PASSING clause must correspond to the order in which the variables were defined using the PARAMETER command. If fewer parameters are passed than the variables that have been defined, the surplus variables will have null values.

Variables are local to the DOCUMENT in which they are defined. Using the PASSING keyword, a DOCUMENT may pass the value of a variable on to a variable in a sub-documents to which it refers, but the name of the variable in the sub-document will not be recognized as a variable in the DOCUMENT which refers to them.

When the included or extracted document is output, the values of the variables are reproduced in place of the variable names.

The PARAMETER command must appear before the variable is first used in the DOCUMENT contents. If the variable's name occurs before the PARAMETER command, it will not be recognized as a variable and in the output document it will be reproduced as static text, that is, no substitution will take place.

Variables may be used in commands as well as text. For example, a DOCUMENT could include the following Manager Products command:

```
@KEEP LIST PROGRAM IF INTRODUCED AFTER 'START-DATE';
```

START-DATE is a variable set earlier in the sub-document. When the DOCUMENT is referenced a date value is passed to this variable using the PASSING facility and the command is then executed as though the date had been specified as hard data.

The ARRAY Command

The array command passes the subsequent lines until the semicolon (;) character into the named command array. The syntax is as follows:

```
>------- ??ARRAY --- array-name --- ; -------------------------->
>------- <<<<<< value ------------------------------------------>
>------- ; ----------------------------------------------------><
```

Note:

The two semicolon terminators. Both are essential.

For example:

```
??ARRAY WEEKDAY ;
Monday
Tuesday
Wednesday
Thursday
Friday
 ;
```

This sets up an array named WEEKDAY with Monday, Tuesday, Wednesday, Thursday and Friday as its elements 1 to 5. The elements may be accessed as, for example, WEEKDAY(2) by EDIT execs specified in INCLUDE and EXTRACT commands. They cannot be used directly in text as variables can.

The FUNCTION Command

The FUNCTION command allows you to introduce user-defined routines into document definitions. The routines, written in Procedures Language, are defined within the FUNCTION command and become temporary USER members on the MP-AID. They are called from within the BODY command. After use the USER member is automatically deleted.

The FUNCTION command must be placed before the BODY command that will call the routine it defines.

Its syntax is as follows:

```
➤──────── ??FUNCTION DEFINE routine-name ──── ; ──────────➤
➤──────── <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
          line of user routine definition ──────────────────➤
➤──────── ; ──────────────────────────────────────────────➤◄
```

Note:

The two semicolon terminators. Both are essential.

The following example shows how the FUNCTION command can be used to perform operations that would otherwise be impossible within a DOCUMENT definition.

```
??FUNCTION DEFINE SALIAS;
DO ALIAS_TYPE()
    I=FDO(ARRAY)
    IF ALIAS_TYPE(I)=COBOL THEN
        RETURN_VALUE=ALIAS_NAME(I)
END
 ;
??BODY KEPT A;
*                     ALIAS(*,,,EXEC SALIAS)
;
```

In this example the routine defined under the name SALIAS returns the COBOL alias for the member against which it is called. The BODY command works through the members named in the KEPT-DATA list A and returns the name of each member, together with its COBOL alias if it has one.

Note:

The FUNCTION command can return only one value. If you wish to employ a user-defined routine to return more than one value, you should use EXTERNAL, described in "Controlling the Output" on page 55.

The BYPASS Command

This command provides a flexible way of either reproducing lines as text or of treating them as comments. The command controls all the subsequent lines, whether they consist of text, comment or commands, until a line containing ??END is found.

This ??END is not a separate command, but is a delimiter for the BYPASS command. It does not need to be followed by a semicolon terminator.

The syntax of the BYPASS command is as follows:

```
>>---??BYPASS---+-WRITE-+---PARAGRAPH---+-YES-+---;--->>
                +-IGNORE-+             +-NO-+

>---<<
```

Its syntax is as follows:

```
>>---- ??SHOW ---- PARAGRAPH ---+-- YES --+---------- ; ----------->>
                                 +-- NO  --+

          <<<
```

outputs the menu of MethodManager's Documentation functions. If you incorporate this command in your DOCUMENT, that menu will be reproduced.

The following commands:

```
@KEEP IN LIST1 WHICH ITEMS CONSTITUTE DG-FI-PERS-DETAILS ;
@LIST KEPT IN LIST1;
```

will cause a list to appear in the output document of the names of the ITEM members to which the DIAGRAM member named DG-FI-PERS-DETAILS refers. The first command creates in the repository a KEPT-DATA list named "LIST1" consisting of the names of the referenced ITEMs. The second command outputs the members' names in this list. These are incorporated as text into the document being created.

Formatting Commands

Formatting commands control the layout of the assembled document.

The formatter assumes that the document will be printed in a fixed-pitch typeface as provided on a line printer. Commands are available, however, which issue predefined escape sequences in conjunction with headings or wherever you wish. These allow you to control typographical effects, such as bold and underline, if your document is printed on a laser or similar printer.

The pages of the document are normally placed in frames, which may include page headers and footers. You may define separate frames for left and right pages if your document is to be printed on both sides of the paper.

It is important to distinguish between the *physical page size*, which represents the area of the page on which your printer is capable of printing, and the *logical page size* which is the area defined by your frames. This is smaller than the physical page, since it is standard practice to leave some space at the head and foot of each page.

If your organization uses a standard format for its documentation, you can save much time and effort by creating a DOCUMENT member that contains just the one (or two) frame definitions needed to reproduce that format. You can then ensure that each document you produce conforms to that style by using the INCLUDE command to make your frame DOCUMENT the first sub-DOCUMENT referenced in the CONTENTS attribute. The frames defined in it will then be used throughout your document.

Each frame governs all subsequent pages in the document until another frame is defined. So you may use different styles for different sections of your document by executing another frame definition at the point where you wish the style to change. For example, you might use one style for the preliminary pages of a report and a different style for the main body of the text.

Formatting commands are prefixed by the same escape character as DOCUMENT assembly commands. By default this is ??. You must enter the command itself immediately after the escape characters, without any intervening space. You must use capital letters and you should terminate the command with a semicolon terminator (;).

Defining a Frame Layout

It is not essential that you define frames for your document, but you are recommended to do so as these give you greater control over the appearance of your finished document. They also allow you to define headers and footers which may include the page number.

The following commands allow you to define frame layouts:

- MASK, LMASK and RMASK signify the start of a frame definition.
- LINE determines the length and position of the frame.
- PROCEND signifies the end of a frame definition.

The MASK, LMASK and RMASK Commands

You must indicate the start of a frame definition by the command RMASK for a right-hand page, that is an odd-numbered page, or LMASK for a left-hand page, that is an even-numbered page. If the document is to be printed on one side of the paper only, all pages are assumed to be right-hand pages, so you could use RMASK, but you may use the generic command MASK if you prefer. Its function is identical to RMASK. The command does not have any keywords.

The LINE Command

The LINE command inserts blank lines into the frame until the specified line number is reached. A frame definition is assumed to begin at the top of the physical page, the first printable line being designated line 1. So, if you wish to leave five blank lines between the top of the page and the header, which marks the top of the logical page, your frame definition might begin:

```
??RMASK ;
??LINE 5 ;
```

The header is a line of text at the top of the logical page. As static text this is typed without any escape character. For example:

```
SYSTEM USER MANUAL
```

Many users like to include a delimiting line of hyphens, equals signs or other characters beneath the header. This helps to distinguish the header from the body text that will appear below it. This forms another line of static text.

You may also use the LINE command to determine the size of the printable area of the frame, by inserting as many blank lines for the body text area as you require. You should, of course, ascertain the number of lines that will fit on a page. For example, if you want 50 lines in your text area, you should now insert the command:

```
??LINE 50 ;
```

If you wish your frame to include a footer, this should now follow, preceded by a delimiting row of symbols if you wish. The footer often includes the page number, either in the centre of the line or on the right in a right-hand page or on the left in a left-hand page.

The first line and the last line within the frame on which text will be printed are determined by the PBOT and PTOP commands. These must be used within the main body of the document, that is outside frame definitions.

Ending a Frame Definition

Finally, you must use the PROCEND command to indicate the end of the frame definition.

The following examples show typical right-hand and left-hand page definitions:

```
* Right-hand page frame follows
??RMASK;
??LINE 5;
USER MANUAL
═══════════════════════════
??LINE 50;
═══════════════════════════
                    Page &PAG
??PROCEND;

* Left-hand page frame follows
??LMASK;
??LINE 5;
                OFFICE SYSTEM
═══════════════════════════
??LINE 50;
═══════════════════════════
Page &PAG
??PROCEND;
```

In the above examples the variable &PAG causes the current page number to be printed. This facility is described in detail in "Inserting Page and Title Numbers" on page 91.

General Text Formatting Commands

General text formatting commands available include:

- LMAR and RMAR set the left and right margins of the text respectively. PTOP and PBOT set the top and bottom position of text respectively.
- SKIP allows you to leave blank lines and LINCOUNT allows you to compensate for inaccuracies in the line count caused, perhaps, by print in a larger typesize. PAGE allows you to leave blank pages or to resume printing on a fresh page.

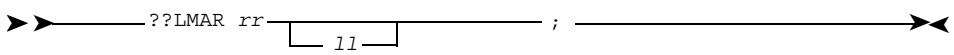
Because these commands apply to all pages, and not just frames, they should *not* be included in frame definitions.

The LMAR and RMAR Commands

These commands allow you to define the left and right margins. It is standard practice to make the logical page width narrower than the physical page width.

If your document uses separate left and right pages, you may use different settings on left and right pages, perhaps to allow extra room for the binding.

LMAR sets the left margin. The syntax is as follows:



RMAR sets the right margin. The syntax is as follows:

```
>>---??RMAR rr --+-------+-------------; ---------------------><
                 |  ll   |
```

rr and *ll* are the margin positions, measured in characters from the left-hand edge of the physical page, for right-hand pages (or standard pages) and left-hand pages respectively. The number for left-hand pages is omitted if the document uses one side of the paper only.

The difference between the left and right margin values defines the maximum number of characters that may appear on a line in the formatted document. If your document includes a line that is longer than this maximum, normally it is truncated and the surplus text is lost. You may, however, use FLOAT mode, described in "Word Wrap" on page 78. This automatically creates one or more new lines to contain text that would otherwise have been lost. In this way no text is lost.

The PTOP and PBOT Commands

The textual material created from the text and commands in the CONTENTS attribute of your document definition is inserted into the frames that you have defined. The PTOP and PBOT commands allow you to specify where the first and last lines of this material fall on the page.

The syntax is as follows:

```
>>---??PTOP rr --+-------+-------------; ---------------------><
                 |  ll   |
```

and

```
>>---??PBOT rr --+-------+-------------; ---------------------><
                 |  ll   |
```

where *rr* and *ll* are the numbers of the first (in the PTOP command) line and the last (in the PBOT command) line on the page on which text is to appear on right and left pages respectively.

The line numbers relate to the physical page and not the logical page. If you have defined a frame, you should ensure that your text starts below any header material and ends above any footers. If you have not defined a frame, you should leave some free space at the top and bottom of the pages.

The SKIP Command

The SKIP command inserts empty lines in your output text. The syntax is as follows:

```
>>---??SKIP--+-------+-------------nn---------; ---------------><
             |  TO   |
```

SKIP followed only by a number (*nn*) inserts that number of empty lines into the text. You might use the SKIP command in this way to create empty space where a diagram will be subsequently placed.

SKIP TO followed by a number (*nn*) leaves a variably sized gap in the page. Printing is resumed at line number *nn* on the physical page.

The LINCOUNT Command

Sometimes the lines being output may occupy more or less space than the physical lines. For example, if you use a laser printer, you could use the TITLE DEFINE CHAPTER-LASER-CONTROL command to reproduce your chapter headings in a larger type size than the body text. This may lead to disparities in the document formatting. The LINCOUNT command allows you to adjust the current line count so that pages are correctly formatted.

The syntax is as follows:

```
>>---??LINCOUNT --+-- SET --------------+-- nn ---- ; ----------><
                  |-- ADD --------------|
                  |-- SUBTRACT ---------|
```

The SET option sets the current line count to the specified value *nn*. The ADD option adds the value of *nn* to the current line count. The SUBTRACT option subtracts the value of *nn* from the current line count.

A similar command, TLINCOUNT, is used in the Table of Contents.

The PAGE Command

The PAGE command forces the start of a new page and may insert a number of empty pages. The syntax is as follows:

```
>>---??PAGE ---+-------------------------+-------------+-- ; ----------><
               |-- TO ----------------|  |-- nn --|
               |-- LEFT ---------------|
               |-- RIGHT --------------|
```

Used alone, thus:

```
??PAGE ;
```

the command forces printing to the start of the next page, unless the printing position is already at the top of a page. Printing will resume at the top of the next page, which may be a left or right page.

Followed by LEFT or RIGHT, thus:

```
??PAGE LEFT;
```

or

```
??PAGE RIGHT ;
```

the command forces the next text on to the next left or right page respectively, unless the printing position is already at the start of the next left or right page, as specified.

Followed by TO *nn*, where *nn* is the number of a page, for example:

```
??PAGE TO 6 ;
```

the command forces the next printed text to appear at the top of the specified page, inserting as many blank, intermediate pages as necessary.

You may also force the insertion of a specified number of blank pages by following the command with the number of blank pages required. For example:

```
??PAGE 4 ;
```

would force the insertion of four blank pages at this point, printing being resumed at the top of the fifth page after the current position.

A variant of this command, TPAGE, is used in the Table of Contents. It works in the same way.

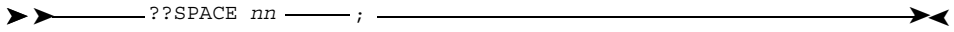
Special Formatting Effects

A number of special text formatting facilities are available. These make use of the following commands:

Command	Description
SPACE	Controls line spacing.
JUSTIFY, FLOAT, ASIS and FIXED	Are concerned with line width and word wrap.
PRINT	Allows you to select the range of pages printed.
PCTL	Allows you to highlight sections of text using the effects available in laser and similar printers.
PARA	Prevents paragraphs from becoming separated by page breaks and UPPER forces text into upper-case (capital letters).

Space Limit

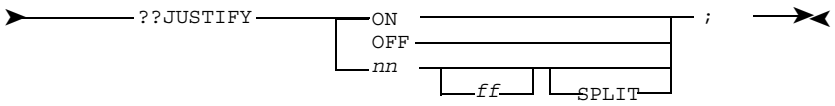
This command limits the amount of space that may be taken by the following command. It is often useful if the amount of text generated by the command is not predictable. The syntax is:



where *nn* is the maximum number of lines of output that will be allocated to the command. If the output from the command exceeds *nn* lines, the excess will not be reproduced.

Text Justification

You can arrange for the text in your document to be justified, that is for each line to be expanded to fill the logical page width, like text in most books and newspapers. The JUSTIFY command has the following syntax:



JUSTIFY ON starts justfying text and JUSTIFY OFF turns justification off.

Justification is achieved by inserting additional spaces between words as necessary. The additional spaces are inserted from the right to the left and more than one additional space may be inserted if necessary.

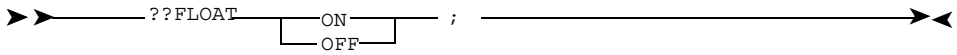
You can tailor the justification process to suit your preferences. Since excessively wide spaces between words can make a document untidy, you may set a limit to the number of spaces that will be inserted into a line. This limit is set by inserting the number *nn* immediately following the JUSTIFY command. If justification would cause this limit to be exceeded, no justification takes place and the line is reproduced at its 'natural' length. You may wish to experiment with different values of this number to see its effect. A suitable starting point is twice the average number of words per line, which will limit word spaces to three times their normal width.

You may also set a weighting factor (*ff*) which controls the proportion of the word spaces that may be expanded. By default this is 100% so that all word spaces are liable to expansion. If you set this to 75%, only the last 75% of the word spaces in the line will be available for expansion.

If the SPLIT option is set, a word-wrap routine is enabled. If a line is too long to fit between the margins, normally it is truncated. As much of it as will fit is output (and justified, if justification is turned on), but the remainder is lost. The word-wrap routine, however, saves the text that would otherwise have been discarded in the truncation process and creates from it a new line below the current one.

Word Wrap

If a line of text is too long to fit in the space available, normally it is truncated. As much of it as possible is printed, but the surplus text is discarded. You may, however, force the text that would otherwise be discarded to be printed as one or more new lines below the current one. The word wrap routine that handles this is called the "float facility". The syntax of the command which controls this facility is as follows:



To turn word wrap on use the following command:

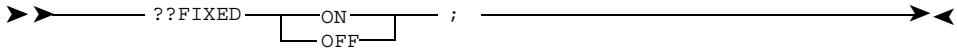
```
??FLOAT ON ;
```

To turn the facility off, use the following command:

```
??FLOAT OFF ;
```

If the float facility is turned off, each line is simply truncated, the surplus matter being discarded.

A separate command is available which controls both justification and word wrap. Its syntax is:



The following command:

```
??FIXED ON ;
```

turns both justification and word wrap off. Any special justification parameters that had been in use are stored. The command:

```
??FIXED OFF ;
```

resumes justification and float with the previously used settings.

Another command with the syntax:

```
>>---??ASIS---;---------------------------------------------><
```

also turns off justification and float and forces subsequent text to be reproduced as it is.

Print Range

You can restrict printing to left or right pages or to a certain spread of pages using the PRINT command.

The command has the following syntax:

```
>>---??PRINT-----------------------------------------------;---><
              |--- LEFT ---|     |-------------- nn --------------|
              |--- RIGHT ---|     |--- FROM ff TO tt ---|
```

LEFT or RIGHT restricts printing to the even- or odd-numbered pages respectively. Followed by a single number (*nn*), PRINT will only print the page having that number.

Followed by FROM *ff* TO *tt*, where *ff* and *tt* are page numbers, printing will be restricted to that range of page numbers, inclusively. This form of the command can be combined with the LEFT or RIGHT options. For example,

```
??PRINT LEFT FROM 4 TO 10;
```

will print only the left-hand pages from page 4 to page 10 inclusive.

A variant of this command, TPRINT, is available for use in the Table of Contents.

Printer Control Characters

This facility, intended primarily for use when output is to a laser or similar printer, allows you to switch between normal print and bold or other effects. The syntax is as follows:

```
>>---??PCTLn---+--- ON ---------------+---;---------------><
               |--- OFF --------------|
               |--- DEFINE -----------|
```

where *n* is blank, 1 or 2, allowing you to control up to three different effects independently.

You must include the following command sequence early in your document, before the first line to which you wish to apply the effect:

```
??PCTLn DEFINE ;
control sequence a
control sequence b
```

The two control sequences *must* be on separate lines *following* the semicolon terminator. `Control sequence a` is the printer control sequence needed to switch to the highlighting effect and `control sequence b` is the sequence needed to restore normal text.

The control sequences, which you should obtain from your printers User Manual, must be entered as characters. If, as is likely, the sequence begins with the Esc character (EBCDIC hex 27) which is not available from the keyboard, you must first redefine one of your keyboard characters as this character. You should choose one that you do not intend to use elsewhere in your document. If, for example, you decide to redefine the NOT (¬) character which has EBCDIC hex code 5F to represent the Esc character, you should issue the following command from the TSS command area:

```
SET CHARACTER-TRANSLATION INPUT 5F 27 27
```

You can then type the (¬) character in your document wherever you require the Esc character. If you use a Hewlett-Packard Laserjet series II or compatible printer, the following escape sequences may be useful to you:

¬(s3B turns on bold print
¬(s0B turns off bold print

¬&dD turns on continuous underlining
¬&3D turns on word underlining
¬&d@ turns off underlining

¬(s 1S turns on italic print
¬(s0S turns off italic print

When you first type these control sequences, the ¬ character will appear on screen as normal. When you press Enter in preparation for filing your document definition, the update buffer contents will be rewritten and the ¬ characters will be replaced with Esc characters, represented on screen by blank spaces. If your control sequences, like most of those above, include lower-case characters, you will need to issue the command SET UPPER OFF at the command line, otherwise the lower-case characters will be translated automatically to upper-case and the control codes will not work.

After the control code definition sequence, the following command:

```
??PCTLn ON ;
```

causes the printer to switch to the chosen highlighting effect. The effect is applied to all subsequent printer output until the control sequence needed to restore normal text is sent to the printer.

The following command:

```
??PCTLn OFF ;
```

sends the control sequence that restores normal text.

Note:

Other commands are available which automatically send the printer control codes needed to switch to a highlighted style when reproducing headings. These are described in "Subheading Style" on page 86 and "Chapter Heading Style" on page 88. In contrast, you may use the PCTL command to add emphasis to any line or any group of lines in your document contents.

Since each PCTL command of necessity occupies a whole line in the document contents, the effect which it controls may be applied only to a whole line or a group of lines. If you wish to apply an effect to one or more individual words in a line, you may do so using user variables to store the printer control codes: see "User Variables" on page 92.

Paragraph Protection

This facility controls the way in which paragraphs are handled if a page break occurs in them. The syntax is:

```
>>---??PARA nn---;-----------------------------------------------><
```

where *nn* is the minimum number of lines in the paragraph that must appear on the current page before a page break in the paragraph is permitted.

For example, if you set *nn* as 2 (a commonly used value), then a page break will be permitted within the paragraph only if there is enough room for its first two (or more) lines to be printed on the current page. The remainder of the paragraph will be printed on the following page. If there is not enough room on the current page for the first two lines of the paragraph, the whole paragraph will be kept together and printed on the following page. This eliminates the untidy appearance in documents of 'widows', that is, first lines of paragraphs that have become separated by page breaks from the rest of their paragraphs.

Swap to Upper Case

You can control the use of upper-case in alphabetical text in your document using the UPPER command:

```
>>---??UPPER---+-ON-+---------------;----------------><
               +-OFF-+
```

UPPER ON changes all subsequent alphabetical text to uppercase, that is capital letters. UPPER OFF cancels this effect and outputs subsequent text without any change of case. You may use these commands to change certain sections of your document contents to capitals only.

Global Printer Control Commands

Two global printer control commands are provided. These commands are issued just once during the output of each document to set up the printer. They should be located as close to the start of the document as possible, to ensure that the printer is correctly set up before any text is printed.

The commands are as follows:

SHIFT determines the physical page width and LPAGE determines the physical page length.

Horizontal Placing on the Page

SHIFT controls the position page across the page at which printing begins. It is applied before margin settings and allows for, for example, the effect of punched holes for binding. Its syntax is:

```
>>---- ??SHIFT rr ---+-------+---------- ; ----------><
                     +- ll -+
```

where *rr* and *ll* are the number of columns that text is to be shifted to the right for a right-hand (or standard) page and a left-hand page respectively.

A similiar command, TSHIFT, defines the physical page width in the Table of Contents.

Page length

LPAGE defines the physical page length in lines. The syntax is:

```
>>---- ??LPAGE nn ---- ; ----------------------------------------><
```

where *nn* is the number of lines per page. You only need to use this command if you have not defined any frames for your document.

A similar command, TLPAGE, defines the physical page length in the Table of Contents.

Controlling Headings and Sections

By default your document will have numbered headings, using a hierarchical numbering system similar to the one in this manual. The headings are created automatically from the titles and headings in the sub-documents to which your document refers. This hierarchical numbering system is created and maintained automatically.

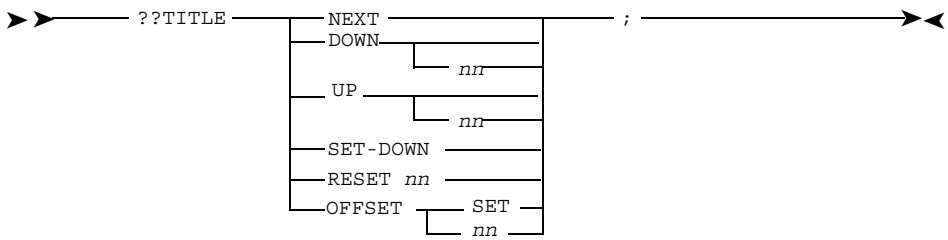
The title of the DOCUMENT being assembled is excluded from the numbering system, since inevitably it remains unchanged throughout the document and therefore is not informative. The titles of the included sub-documents form the first level of the hierarchy and their headings are sometimes called *chapter headings*. Documents referenced by these form the second level. The maximum depth of the hierarchy is 10 levels.

You can use the hierarchical system to control a wide range of effects through the TITLE command and the TITLE DEFINE command. These are followed by a number of keywords which determine how they are used.

There are also some special TITLE DEFINE commands used in laying out the Table of Contents. These are described later in "Page Number Position" on page 89 to "Chapter Heading Style" on page 91.

Reproducing Heading Numbers

The TITLE command reproduces a heading number from your document. The heading number it reproduces is controlled using a number of attributes to move around the hierarchical structure. The syntax is as follows:



When you first use the TITLE command in your document, it will reproduce the number 1 which is the first number in the first hierarchical level.

The NEXT keyword prints the next heading number in the same hierarchical level as the last one printed. For example, if the last heading printed by the title command was 1.1, the command TITLE NEXT would print a heading numbered 1.2, even if the document contained a subheading numbered 1.1.1.

The DOWN attribute moves the pointer downwards in the hierarchical structure. For example, if the last heading number printed by the TITLE command was 1.1, the command TITLE DOWN would print your heading number 1.1.1.

Similarly, you may move the pointer up a level using the UP keyword. This automatically performs a NEXT move as well. For example, if the last heading number printed by the TITLE command was 1.1.2, the command TITLE UP would print your heading number 1.2, this being the next heading on the hierarchical level above that of the previously printed heading.

You may move the heading pointer up or down more than one hierarchical level by specifyng the number of levels. For example, if the last heading printed by the TITLE command was 1.2.2.1, the command TITLE UP 2 would print your heading number 1.3, this being the next heading two hierarchical levels above the previously printed one.

If you specify a heading level that does not exist - for example, if you use a TITLE UP command while at the highest level - an error message will be issued.

You can change your heading level without outputting a number. To change down to the next level, use TITLE SET-DOWN.

To change up a number of levels without outputting a number you may use TITLE RESET *nn*.

The TITLE OFFSET SET command allows you to set the current title level as a temporary default. When a level has been set in this way, you can change levels relative to the default using TITLE OFFSET *n*.

For example, in the same document:

```
??TITLE NEXT;
```

prints out as:

```
1.
```

Subsequently,

```
??TITLE DOWN;
```

prints out as:

```
1.1.
```

A further

```
??TITLE DOWN;
```

prints out as:

```
1.1.1.
```

At this point (on level 3)

```
??TITLE OFFSET SET;
```

sets level 3 as the current default (level 1) for TITLE OFFSET commands. It does not output any text.

So,

```
??TITLE OFFSET 3;
```

sets the level to two levels below the current level, but does not output any text.

At this point, the following command:

```
??TITLE NEXT;
```

will output the next available heading number on the current level, which is:

```
1.1.1.1.1.
```

The following command:

```
??TITLE OFFSET 1;
```

restores the current level to the current default for TITLE OFFSET commands, namely level 3, but does not output any text.

So, at this point the following command:

```
??TITLE NEXT;
```

will output the next available heading at that level, which is:

```
1.1.2.
```

Changing the Level Delimiter Sign

Normally a full stop (period) sign is used to differentiate hierarchical levels in the title numbers, thus: 1.2.4.1. You may, however, replace the full stop with any other single character using the command:

```
>>---??TITLE DEFINE PERIOD c ---------- ; ----------><
```

where *c* is the character you wish to use. The default is the full stop (.).

Space Before Headings

The space that appears before each heading is controlled by:

```
>>---??TITLE DEFINE BSKIP nn ---------- ; ----------><
```

where *nn* is the number of blank lines to be printed before the heading. *nn* may be set to 0 if you wish. Depending on the way in which your document facility has been tailored, a default value for BSKIP will have been set and will displayed automatically in a line at the start of your assembled DOCUMENT.

Space After Headings

The space that appears after each heading is controlled by:

```
>>---??TITLE DEFINE ASKIP nn ---------- ; ----------><
```

where *nn* is the number of blank lines to be printed after the heading. *nn* may be set to 0 if you wish. Depending on the way in which your document facility has been tailored, a default value for ASKIP will have been set and will be displayed in a line at the start of your assembled DOCUMENT.

Space Between Consecutive Headings

The space that separates two consecutive headings is controlled by:

```
>>---??TITLE DEFINE TTSKIP nn ---------- ; ----------><
```

where *nn* is the number of blank lines to be printed between the two headings. *nn* may be set to 0 if you wish. Depending on the way in which your document facility has been tailored, a default value for TTSKIP will have been set and will be displayed in a line at the start of your assembled DOCUMENT.

Title and Paragraph Relation

The minimum number of lines of a paragraph that must appear on the same page as the paragraph heading is controlled by:

```
>>---- ??TITLE DEFINE PARAGRAPH nn ---------- ; ----------><
```

where *nn* is the minimum number of lines. The purpose of this facility is to prevent a heading from being printed at the foot of a page while the paragraph to which it relates is on the following page, or while only its first one or two lines is reproduced on the same page. Instead the whole paragraph and its heading are forced on to the following page so that they remain together. If ASKIP and BSKIP are both defined, the value for PARAGRAPH should be at least ASKIP + BSKIP + 2. Depending on the way in which your document facility has been tailored, a default value for PARAGRAPH will have been set and will be displayed in a line at the start of your assembled DOCUMENT.

Subheading Style

If you use a laser or similar printer, you can automatically print subheadings in a different style, using the following command:

```
>>---- ??TITLE DEFINE HEADER-LASER-CONTROL ------- ; ---------->
>---------- control sequence a ------------------------------------->
>---------- control sequence b ------------------------------------><
```

The two control sequences *must* be on separate lines *following* the semicolon terminator. `Control sequence a` is the sequence of codes needed to turn on the effect, perhaps bold, underlined or italic type, that you wish to use for your subheadings. `Control sequence b` is the sequence of codes needed to restore normal print. When this command has been executed, `control sequence a` will be sent automatically to your printer before it prints each subheading and `control sequence b` after each subheading to return the printer to normal print.

See "Printer Control Characters" on page 79 for further information on entering printer control sequences in DOCUMENT definitions.

A subheading, for the purposes of this command, is defined as a heading of any level below level 1. A separate command, TITLE DEFINE CHAPTER-LASER-CONTROL, described in "Chapter Heading Style" on page 88, allows you to use a different highlighting effect for chapter headings, that is level-1 headings.

Space Between Chapters

The following formatting commands relate to *chapters*. A chapter is a division in your document between two headings at the first hierarchical level. A chapter heading is therefore a first-level heading.

The space to be inserted at the end of a chapter is controlled by the following command:

```
>>---- ??TITLE DEFINE CHAPTER-SKIP ------------- nn ---------- ; ---><
                                   |-- PAGE ----|
                                   |-- LEFT ----|
                                   |-- RIGHT ---|
```

nn represents a number. If the value is a number, that number of blank lines is inserted after the end of the chapter and before the heading of the next chapter. If a page break occurs in the blank lines, the blank lines on the new page are cancelled and the new chapter will start at the top of the next page.

If PAGE is specified, the new chapter will start at the top of the next page, whether it is a left or a right page. If LEFT or RIGHT is specified, the new chapter will start at the top of the next even-numbered or odd-numbered page respectively. (Even-numbered pages are assumed to be left pages and odd-numbered pages are assumed to be right pages.)

Chapter Heading Prefix

You may insert a standard string of text in front of the new chapter heading. The string is defined using the following command:

```
>>---- ??TITLE DEFINE CHAPTER-PREFIX "text" ---------- ; ----><
```

where *text* is the string to be prefixed to the new chapter. The string should be enclosed in double quotes, for example: "Chapter", "Section" or "Part".

Chapter Heading Position

You can highlight a chapter heading by giving it a special left margin which may be inside the standard left margin or outside it, if there is enough room. The command is as follows:

```
>>---- ??TITLE DEFINE LMAR rr ---+-------+---------- ; ----><
                                  |  11   |
```

where *rr* and *11* are the title's absolute start positions (in characters) from the left-hand printing position on right-hand pages and left-hand pages respectively. If your document is printed on only one side of the paper, all pages are treated as right-hand pages and you omit the figure for left-hand pages.

You can also indent the chapter heading using the following command:

```
>>---- ??TITLE DEFINE HEADING-OFFSET nn ---------- ; ----><
```

The number *nn* is the new position relative to the current setting of the left margin.

Chapter Indenting

You may highlight a whole chapter by giving it a special left margin which may be inside the standard left margin or outside it, if there is enough room. The command is as follows:

```
>>---- ??TITLE DEFINE CHAPTER LMAR rr ---+-------+---------- ; ----><
                                          |  11   |
```

where *rr* and *11* are the chapter's new left margin (in character positions) from the left-hand printing position on right-hand pages and left-hand pages respectively. If your document is printed on only one side of the paper, all pages are treated as right-hand pages and you omit the figure for left-hand pages.

Chapter Heading Style

You can control the style in which laser and similar printers reproduce your chapter heading using the following command:

```
>>---??TITLE DEFINE CHAPTER-LASER-CONTROL--- ; ---------->
>--- control sequence a ----------------------------------->
>--- control sequence b ----------------------------------><
```

The two control sequences *must* appear on separate lines *following* the semicolon terminator. `Control sequence a` is the sequence of codes needed to turn on the effect, probably bold type, that you wish to use for your chapter headings. `Control sequence b` is the sequence of codes needed to restore normal print. When this command has been executed, `control sequence a` will be sent automatically to your printer before chapter headings and `control sequence b` afterwards to return the printer to normal print.

See "Printer Control Characters" on page 79 for further information on entering printer control sequences in DOCUMENT definitions.

Formatting the Table of Contents

The formatter which processes your document, fitting its content into frames, also has another task. It compiles the Table of Contents automatically from the TITLE attributes of referenced DOCUMENTs.

This Table of Contents is output as a separate document from the main body of text. It is created on a separate page or pages whose numbering is not continuous with that of the rest of the document and it can be given its own frames.

You can control its format using special formatting commands at the corresponding points in the main document definition. Many of these formatting commands are variants of commands used in the main document body. In general their names are the same as the regular commands, but they are prefixed with a "T". They are used in the same way as the corresponding regular commands and the following Table lists these commands and their equivalents and the numbers of the Sections where further details (of the regular commands) are given.

Table of Contents	Main Document	Description
??TLMASK	??LMASK	"The MASK, LMASK and RMASK Commands" on page 73
??TRMASK	??RMASK	"The MASK, LMASK and RMASK Commands" on page 73
??TLMAR	??LMAR	"The LMAR and RMAR Commands" on page 74
??TRMAR	??RMAR	"The LMAR and RMAR Commands" on page 74
??TPTOP	??PTOP	"The PTOP and PBOT Commands" on page 75
??TPBOT	??PBOT	"The PTOP and PBOT Commands" on page 75
??TSKIP	??SKIP	"The SKIP Command" on page 75

Table of Contents	Main Document	Description
??TLINCOUNT	??LINCOUNT	"The LINCOUNT Command" on page 76
??TPAGE	??PAGE	"The PAGE Command" on page 76
??TPRINT	??PRINT	"Print Range" on page 79
??TSHIFT	??SHIFT	"Horizontal Placing on the Page" on page 82
??TLPAGE	??LPAGE	"Page length" on page 82
??TPSET	??PSET	"Inserting Page and Title Numbers" on page 91
??TVSET	??VSET	"User Variables" on page 92

There are also some TITLE DEFINE format commands which are peculiar to the Table of Contents. These are:

TITLE DEFINE PAGE-OFFSET. Determines the position of the page number in the lines that make up the Table of Contents. TITLE DEFINE TITLE-OFFSET applies a separate left margin to entries.

TITLE DEFINE MAX-LEVEL. Determines which levels of title are included in the Table of Contents.

TITLE DEFINE TCHAPTER-SKIP. Allows you to distinguish chapter entries in the Table of Contents by introducing space at the end of a chapter. TITLE DEFINE TCHAPTER-PREFIX allows you to place a prefix (such as "Chapter") before Chapter entries and TITLE DEFINE TCHAPTER-LMAR allows you to apply a special left margin to Chapter entries. Finally, the style in which Chapter entries are printed on laser and similar printers can be controlled using TITLE DEFINE TCHAPTER-LASER CONTROL.

Descriptions of these follow.

Page Number Position

You can control the position of the page numbers in the topic lines in the Table of Contents using the following command:

```
>>---- ??TITLE DEFINE PAGE-OFFSET nn ------------------------ ; ----><
```

where *nn* is the character position for the page number, the currently set left margin being regarded as 1. The page numbers are retrieved and inserted automatically. Clearly you must make the number large enough to place the page number clear of the longest title, but not so large that the page number is forced off the page.

Entry Position

You can control the position of an entry in the Table of Contents using the command:

```
>>---- ??TITLE DEFINE TITLE-OFFSET nn ------------------------ ; ----><
```

where *nn* is the character position for the title concerned, the currently set left margin being regarded as 1. Clearly you must make the number large enough to place the page number clear of the longest hierarchical level number, but not so large that no room is left after the title for the page number.

Limit of Hierarchical Levels

The following command determines which hierarchical levels are included in the Table of Contents. The syntax is:

```
>>---- ??TITLE DEFINE MAX-LEVEL nn ------------------------- ; ---->><
```

where *nn* is the lowest level at which you wish titles included in the List of Contents. For example, if you set this level to 3, your Table of Contents will include all titles at levels 1, 2 and 3, but titles at levels 4 or below will be excluded. Thus entries such as 1.1.4 will be included, but not entries such as 1.1.2.1.

Chapter Breaks

You may distinguish chapter breaks in your Table of Contents by introducing space before them, or even forcing them to a new page. The command is:

```
>>---- ??TITLE DEFINE TCHAPTER-SKIP ---------- +- nn -------+- ; ---->><
                                               |- PAGE -----|
                                               |- LEFT -----|
                                               '- RIGHT ----'
```

This command works in the same way as TITLE DEFINE CHAPTER-SKIP described in "Space Between Chapters" on page 86. The default setting is 1, which leaves one blank line after each chapter.

Chapter Prefixes

You may distinguish chapter-level headings in the Table of Contents by inserting a prefix, such as "Chapter" in front of them. The command that controls this is:

```
>>---- ??TITLE DEFINE TCHAPTER-PREFIX "text" ------------- ; ---->><
```

This command works in the same way as TITLE DEFINE CHAPTER-PREFIX described in "Chapter Heading Prefix" on page 87.

Chapter Heading Position

You may distinguish chapter-level headings in the Table of Contents by giving them a different indentation, perhaps outside that of titles of other hierarchical levels. The command that controls this is:

```
>>---- ??TITLE DEFINE TCHAPTER LMAR rr ------+-------+- ; ---->><
                                             '- ll --'
```

It works in the same way as the TITLE DEFINE CHAPTERLMAR command described in "Chapter Indenting" on page 87. The numbers *rr* and *ll* define the new starting position of the title on right and left pages respectively the current left margin setting being taken as 1.

Chapter Heading Style

You may distinguish chapter-level headings in the Table of Contents by giving them a different style. This command allows laser and similar printers to use a different type style, perhaps bold type, for chapter-level headings. The command syntax is:

```
>>---??TITLE DEFINE TCHAPTER-LASER-CONTROL---;---------->
>---- control sequence a ----------------------------------->
>---- control sequence b ----------------------------------><
```

It works in the same way as the TITLE DEFINE CHAPTER-LASER-CONTROL command described in "Chapter Heading Style" on page 88.

See "Printer Control Characters" on page 79 for further information on entering printer control sequences in DOCUMENT definitions.

Inserting Page and Title Numbers

You may insert the current page number or the current heading number at any hierarchical level anywhere in the text. The most common usage would be in headers or in footers.

These numbers are stored in special variables used by the Documentation facility. Unlike the textual variables which you set up in your document using the PARAMETERS command, you do not need to set these up before you use them; they are set up and maintained automatically and are present whether you use them or not.

To reproduce the current page number, include &PAG. To reproduce the previous page number, include &LPA. To reproduce the next page number, include &NPA. In the finished document, these variable names will be replaced by the appropriate page numbers.

Page numbers are normally printed in numeric format, but you may specify the use of Roman numerals instead, by prefixing the variable name with the letter R. For example, &RPAG will reproduce the current page number in Roman numerals. This may be useful, for example, in the preliminary pages of a manual. Only upper-case Roman numerals are available. The highest page number that can be represented in Roman numerals is 3999.

You may reset the current page number if you wish. This may be useful if you are creating a second volume to an existing document and you wish its page numbering to follow on from that of the first volume or if pages from some other source are to be bound into your document. The command, normally issued from within a frame definition, is ??PSET. Its syntax is:

```
>>---??PSET nn ------------------- ; ---------------------><
```

For example, the command PSET 65 sets the current page number to 65. A variant of the command, TPSET, is available for use in the table of contents.

To reproduce the current heading number at a specific hierarchical level use the variable &T*nn* where *nn* is the level number. For example, &T05 will output the number of the current heading at level 5. If you use &T00, this will output the last used heading number, irrespective of the level.

User Variables

Two sets of user variables are permanently available in your DOCUMENT. One is associated with the Table of Contents and the other with the main body of the document. Both sets contain ten variables named &V0 to &V9.

To set a variable in the main body of the document, use the command:

```
>>---??VSET &Vn value ------------------- ; ----------><
```

To set a variable in the Table of Contents, use the command:

```
>>---??TVSET &Vn value ------------------- ; ----------><
```

The *value* may be any string of characters, but must not include any spaces. You may use the VSET and TVSET commands repeatedly so that the same variable has different values in different parts of the document.

To use a variable simply include its name in the document contents. During formatting the variable name is substituted by the value most recently assigned to that variable. The output line, however, follows the format of the original line in the document definition exactly. Consequently, if the value is longer than the variable name (which is always three characters long), it may be overwritten by any text following on the same line. If you know the length of all values to be used, you can allow for this by leaving extra spaces after the variable name. If you wish to use values of different or unknown lengths, you may need to contrive your text so that the variables always fall at the end of a line.

If your document is to be printed on a laser or similar printer, you can use the user variables to reproduce printer control codes. This allows you to change the style of individual words, for example to print one or more words in bold or italics in a line of otherwise normal print.

Since these control codes usually contain the Esc character which cannot be included in text lines, you cannot write printer control codes directly in text lines. You can, however, define them as variable values. The values replace the variable name when the document is formatted.

For example, if you are using a Hewlett-Packard LaserJet II or compatible printer, you could print individual words in italics as follows:

```
??VSET &V0 ¬(s1S ;
??VSET &V1 ¬(s0S ;
This line contains &V0 two words &V1 in italics.
```

In the above example the Esc character is represented by the NOT (¬) character. If the formatted file is downloaded to a suitable printer, the "two words" in the text line will be printed in italics, thus:

This line contains *two words* in italics.

Before you can enter these lines, you must configure your terminal to allow the entry of the Esc character and of lowercase characters. "Printer Control Characters" on page 79 explains how to do this.

Global and Command Variables

You may use global or command variables in your documents. You must define the variables using the VARIABLE command before their names first appear in the CONTENTS of your document. The syntax of the VARIABLE command is as follows:

```
>>---------- ??VARIABLE var-name1 var-name2 var-name3...  -------;-------><
```

The values of these variables must be set using procedures outside of the document definition. Substitution of the variable name by the value may take place at assembly time or at formatting time, depending on the nature of the variables. Procedures Language variables, such as &DATE and &TIME are substituted at assembly time. For example, the following lines in a document definition:

```
??VARIABLE &DATE &TIME ;
Files updated at &TIME on &DATE:
```

will at assembly change to:

```
Files updated at 15.15 on 6.6.1994:
```

Commentary Lines

Commentary lines allow you to incorporate explanatory notes into your DOCUMENT definition. They are particularly useful if other users need to understand your DOCUMENT definitions. They have no effect on the execution of commands and they do not appear in the output DOCUMENT; they are visible only in the DOCUMENT definition.

Commentary lines are distingnished by an escape character which by default is a single asterisk (*) character. Each comment is one line in length. If you wish your comment to run on to a second or subsequent line, it may do so, but each line of comment must begin with the appropriate escape character.

Within the scope of the BODY command, comments use an escape character separately defined in the BODY command itself; as described in "Comment" on page 60. You should not define the asterisk (*) character as the identifier for comments within the BODY command, because the (*) character has a special significance, as a wildcard, when used in the context of the BODY command.

You can force comment lines to appear in the output DOCUMENT if you wish. To do this you should use the BYPASS command described "The BYPASS Command" on page 70. The BYPASS command and the SHOW command described in "The SHOW Command" on page 70 can also be used to disable text and commands so that they do not affect the output DOCUMENT and are treated as though they were comments.

Document Assembly and Output

The DOCUMENT definition controls the content and format of the document. The first stage in the process that creates the finished document is *assembly* when MethodManager executes those commands in the DOCUMENT definition that, for example, copy material from other repository members. To assemble your document, you use the DOC command.

The DOC Command

You may issue the DOC command either by entering:

`DOC` *`document-name`*

in the TSS documentation menu command area or from any command area within MethodManager. Alternatively, you may type DOC in the line command area of the appropriate entry in a list of DOCUMENT members such as that obtained from options 1 of the Documentation menu. Alternatively, select item 3 (Print) from the menu, select the appropriate naming convention and enter the name of the DOCUMENT you wish to print.

You will only be able to assemble valid DOCUMENT members which have been encoded in your repository.

The DOC command can also be used in batch, to operate sequentially on several DOCUMENT members.

The DOC command performs these operations:

- It copies static text in the CONTENTS atrribute into the output document
- It executes DOCUMENT assembly commands and, in response to them, extracts matter from other repository members into the output document
- It replaces variables in sub-documents with values that are passed to them from the documents that refer to them
- It replaces system variables in sub-documents with values
- It executes Manager Products commands and integrates their results into the output document
- It removes comments.

DOCUMENT formatting commands are not executed at assembly time. These commands remain in place in the assembled document.

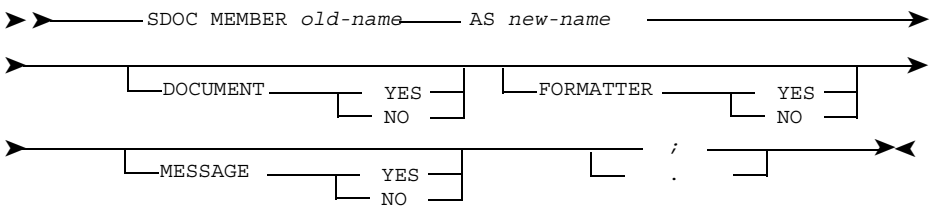
Variables whose values are set by formatting commands within the document itself also remain unchanged within the document contents.

Full formatting of the DOCUMENT, such as the arrangement of the contents in frames does not take place until the next stage of production.

The SDOC Command

The SDOC command assembles your document, as the DOC command does, but it also files a copy of the assembled document in your repository.

The syntax of the SDOC command is as follows:



where:

old-name is the name of the existing DOCUMENT definition

new-name is the name of the assembled document.

The DOCUMENT option specifies whether or not the assembled document is to be filed as a DOCUMENT member in your repository. If you include DOCUMENT YES in your SDOC command or omit the keyword DOCUMENT altogether, the assembled document will be filed as a DOCUMENT member. If you specifiy DOCUMENT NO, any information necessary to file the assembled document in the repository must be present in the assembled document, such as the ENCODE-KEYWORD of the MEMBER-TYPE and any mandatory clauses.

The FORMATTER option offers a facility for stripping all formatting commands out of the assembled document. This may be useful if you wish to decide how the document should be formatted only after it has been assembled (and therefore know, for instance, how long the finished document will be). To strip all formatting commands include FORMATTER NO in the command. To leave them in place include FORMATTER YES, or omit the FORMATTER keyword from the command.

The MESSAGE option determines whether error messages from the assembly process are incorporated into the assembled document. Error messages will be generated if, for instance, a reference is made to a repository member that no longer exists. If you include MESSAGE YES in your SDOC command, error messages arising during assembly will be saved as part of the assembled document. If you do not wish to save these messages, include MESSAGE NO or omit the MESSAGE keyword from the command.

Formatting Your Document

Before your assembled document can be formatted, it must be output as a file. When you have assembled your document as described above, having ensured that the first line of output is the "current" line in the buffer, enter HARDCOPY in the TSS command area. If you type HARDCOPY while the first line of output is not the current line, your document may not be properly processed.

The HARDCOPY command attaches a default header to the top of the contents list. Your Systems Administrator can redefine this header if it is considered inapproporiate.

The sequence of operations needed to initiate the formatter varies, depending on the operating system and on the way in which your system has been set up. You may need to refer to your Systems Administrator.

You will not be able to use the HARDCOPY command, however, unless you have set up a print job suitable for your system. Once you have set up your print job, it resides as a USER member in the MP-AlD and you can use it as often as you need.

The action of the formatter is as follows:

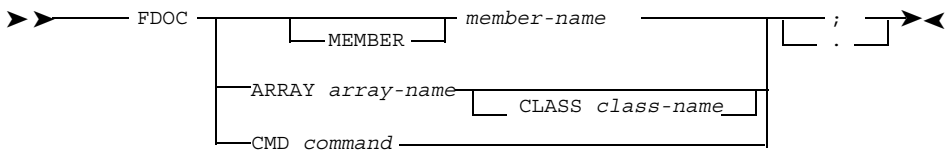
- It executes formatting commands and in response to them
- It substitutes remaining variable names by values
- It lays out the document in frames
- It inserts page numbers as appropriate
- If selected, it applies word wrap and justification to lines of text.

The result of this operation is a pair of text files, one containing the main body of the document and the other the Table of Contents. These may viewed on the screen using a text editor or sent to a printer for a hard-copy printout or converted to ASCII text files for downloading to your PWS, perhaps for further processing in a word processor or desktop publishing application.

The FDOC Command

The FDOC command assembles and formats a DOCUMENT member or formats the contents of an array, the output from a command or the current buffer.

The syntax of the command is as follows:



where:

member-name is the name of the single DOCUMENT member to be assembled and formatted

array-name is the name of an array holding the lines to be formatted

class-name is the class of the variable to which the specified array belongs. Valid values are COMMAND (the default) and GLOBAL

command is the command to be formatted. Multi-word commands must be enclosed in string delimiters.

If the command FDOC is invoked without parameters, the contents of the current buffer are formatted.

Other Documentation Facilities

The DOCUQUERY Command

You can see the Documentation defaults currently set in your system by issuing the DOCUQUERY command from the TSS command area. Because these defaults can be tailored, the actual settings will vary from installation to installation. A typical listing, however, is shown below:

```
Show document name                  : "NO"
Attribute for contents              : "CONTENTS"
Attribute for header                : "TITLE"
Numbering of headers                : "YES"
Echo                                : "NO"
Decompose subdocuments              : "YES"
Member wildcard (Reports)           : "*"
INCLUDE/Member barrier              : "YES"
EXTRACT/Member barrier              : "NO"
Message when member barrier         : "YES"
Document driver resident            : "YES"
Pre LUW statement                   : ""
Post LUW statement                  : ""
Maximum transactions / LUW          : "10"
Formatter escape symbol             : "??"
-------------------------------------------------------------------------------------------
Special character                   : "??"/* Document commands
   Action                           : "NOPRINT CEXEC £"
   max. number of lines             : "32"
-------------------------------------------------------------------------------------------
Special character                   : "£"/* Document commands
   Action                           : "NOPRINT CEXEC £"
   max. number of lines             : "32"
-------------------------------------------------------------------------------------------
Special character                   : "@" /* MPR Commands
   Action                           : "MPR"
   max. number of lines             : "32"
-------------------------------------------------------------------------------------------
Special character                   : "*" /* Comment
   Action                           . "NOP"
   max. number of lines             : "1"
-------------------------------------------------------------------------------------------
Special character                   : ""  /* Text
   Action                           : "WRITE"
   max. number of lines             : "1"
-------------------------------------------------------------------------------------------
```

Tailoring the Documentation Facility

The Documentation Facility can be tailored to suit individual requirements. Tailoring is normally undertaken by the Systems Administrator and is described in *ASG-MethodManager Administration*.

The DCUPD Command

The DCUPD command copies an existing DOCUMENT member in your repository, assigns it a new name and loads it into an assisted update buffer for amendment. This facility is useful if you wish to create a new document that is similar to an existing one, since, by amending the existing document definition, you can save yourself much time and effort.

To use the DCUPD command type:

`DCUPD` *`new-document old-document`*

in the command area of the Documentation services menu or any MethodManager menu; *`new-document`* and *`old-document`* represent the names of the new and the existing DOCUMENT members respectively.

6 Summary of Commands

Panel Interface Commands

Most of the time when you work in the panel interface you will not need to use commands. All the work is done for you when you select menu options or listed members. However you may need to use commands to:

- Cancel or accept updates
- Get help
- Find out about a member
- Edit expert panels.

This chapter summarizes the commands that you can use in the panel interface.

Refer to the *ASG-Manager Products Quick Reference* for the full syntax of the commands summarized in this chapter.

Editing Commands

Enter the following commands in the line command area of panels that you want to edit, for example expert input panels.

Note:

The Scratchpad is a temporary data storage area to which text can be copied or moved.

Command	Description
*	Repeats the previous Line Command
/	Makes any line the Current Line
A	Inserts the contents of the Scratchpad following a line
B	Inserts the contents of the Scratchpad before a line
C	Copies lines to the Scratchpad and overwrites the existing contents of the Scratchpad

Command	Description
CC	Copies block of text to the Scratchpad and overwrites the existing contents of the Scratchpad
D	Deletes lines starting with the line on which the command was entered
DD	Deletes block of text
I	Inserts one or more blank lines after the line on which the command is entered
K	Copies lines to the end of the contents of the Scratchpad
KK	Copies block of lines to the end of the contents of the Scratchpad
M	Moves lines to the Scratchpad and overwrites it
MM	Moves block of lines to the Scratchpad and overwrites it
Q	Moves lines to the end of the Scratchpad
QQ	Moves a block of lines to the end of the Scratchpad
R	Copies the line on which the command is entered and inserts the new line immediately after the copied line
X	Inserts a copy of the Scratchpad after this line

Line Commands

Use the following commands in the line command area, beside the name of the member to which you want to apply the command.

Note:

Some of the following commands apply to proposed members. These are members generated from the Workbench Design Area (WBDA), or the Workbench Translation Area (WBTA), and so do not exist in the repository.

Command	Description
ACT	Includes a repository member in the project view
ADIS	Displays a repository member in an assisted update format
ASM	Generates an Assembler data structure
AUPD	Updates a member in an assisted update skeleton
COB	Generates a COBOL data structure
DACT	Removes a repository member from the project view
DIS	Displays a repository member's source record
DOC	Processes a DOCUMENT member

Command	Description
MER	Loads a repository member into the Workbench Design Area
MERE	Loads a repository member into the Workbench Design Area: data groups belonging to the member are then expanded
MERU	Loads an unverified repository member into the Workbench Design Area, providing it is not a dummy
MERUE	Loads an unverified repository member into the Workbench Design Area, providing it is not a dummy: data groups belonging to the member are expanded.
PLI	Generates a PL/I data structure
PRP	Generates a valid name for a member from the standard abbreviation table
PRPD	Displays the entries in the standard abbreviation table

(Refer to *ASG-MethodManager Administration* for a description of the standard abbreviation table.)

Command	Description
PUR	Removes a repository member to which there are no references
REF	Lists all repository members directly referred to by a member
REFA	Lists all repository members directly or indirectly referred to by a member
RIGN	Specifies that you do not want a proposed member documenting an external object to be entered in the repository
RREN	Renames a proposed member documenting an external object
RREP	Specifies that you want a proposed member documenting an external object to replace an existing repository member
US	Lists all repository members that directly refer to a member
USA	Lists all repository members that directly or indirectly refer to a member
USR	Lists all relationships in which a member participates as source or target
WAL	Lists all repository members that have a specified alias
WF	Lists all members with a specified catalog classification

Help Commands

To display valid input combinations for an input panel, enter a question mark (?) in the command area.

To display user defined help on an input field, enter a question mark (?) in the input field.

To see the InfoBank Help Entry Panel or to request online help on a particular topic, use the HELP command.

To display the Top Level Entry Panel to InfoBank, use the INFOBANK command.

To display help on any member type available in the panel interface, use the MTHELP command.

To display a specific InfoBank panel, use the PANEL command.

To display or list the InfoBank panels you have previously seen, use the RETRACE command.

To display an InfoBank panel chosen from the panel you are viewing, use the SELECT command.

Routing Commands

To route to a panel in a different branch of the panel interface and, when you press PF3, to return to the panel you last saw, enter a plus sign (+) in front of a menu option number.

To route to a panel in a different branch of the panel interface and, when you press PF3, to return to the main menu, enter an equals sign (=) in front of a menu option number.

Primary Commands

Enter the following primary commands in the command area.

To save without encoding all updates made in an assisted update buffer, use the ACCEPT command.

To include a repository member in the project view, use the ACTIVATE command.

To display a repository member in an assisted update format, use the ADISPLAY command.

To insert an assisted update of a member's source record into the current buffer, below the current line, use the AGET command.

To replace the contents of the current update buffer with an assisted update of the source record of another repository member, use the AGETC command.

To generate an Assembler data structure, use the ASM command.

To update the source record of a repository member in an assisted update skeleton, use the AUPDATE command.

To cancel all updates made in an assisted update buffer, use the CANCEL command.

To generate a COBOL data structure, use the COBOL command.

To remove a repository member from the project view, use the DEACTIVATE command.

To display a repository member's source record, use the DISPLAY command.

To start LifeCycle SERVICES, use the LCS command.

To load a repository member into the Workbench Design Area, use the MERGE command.

To load a repository member into the Workbench Design Area, use the MERE command. The data groups belonging to the member are then expanded.

To load an unverified repository member into the Workbench Design Area, providing that the unverified member is not a dummy, use the MERU command.

To load an unverified repository member into the Workbench Design Area, providing that the unverified member is not a dummy, use the MERUE command. The data groups belonging to the member are then expanded.

To define, display, select or delete a view on a member-type, use the MMRVIEW command.

To generate a PL/I data structure, use the PLI command.

To generate a valid name for a member from the standard abbreviation table, use the PRP command.

To display the entries in the standard abbreviation table, use the PRPD command.

(Refer to *ASG-MethodManager Administration* for a description of the standard abbreviation table.)

To remove a repository member to which there are no references, use the PURGE command.

To list all repository members directly or indirectly referred to by a member, use the REFA command. To list all repository members directly referred to by a member, use the REFERENCES command.

To rename a member and change all references to refer to the new name, use the RELABEL command.

To return to the previous panel, use the RETURN command. (Any changes to an assisted update panel are filed.)

To specify that you do not want a proposed member documenting an external object to be entered in the repository, use the RIGN command.

To rename a proposed member documenting an external object, use the RREN command.

To specify that you want a proposed member documenting an external object to replace an existing repository member, use the RREP command.

To start ToolSet SERVICES, use the TSS command.

To list all repository members that directly or indirectly refer to a member, use the USA command.

To list all repository members that directly refer to a member, use the USAGE command.

To list all relationships in which a member participates as source or target, use the USR command.

To list all repository members that have a specified alias, use the WAL command.

To list all members with a specified catalog classification use the WF command.

Additional Note: The RELABEL Command

Member names specified in text clauses, such as NOTE and DESCRIPTION, will only be relabeled if they are both preceded and followed by a blank. Member names in mixed case will not be relabeled.

Document Commands

To process a DOCUMENT member, use the DOC command.

To create a new DOCUMENT member and copy into its CONTENTS attribute information from an existing member, use the DCUPD command.

For details of the commands available for use in the CONTENTS clause of DOCUMENT definitions, refer to Chapter 5, "The Documentation Facility," on page 37.

Additional Commands

TOAUPD

TOAUPD converts a native update (UPD) buffer into an assisted update (AUPD) buffer. This might be helpful in the event of an encode failure, allowing the member definition to be displayed in AUPD format.

7 Input Panels and Help

Help is easily accessible in the panel interface because pressing PF1 always displays help that is specific to the panel you are using. However for input and expert panels the information given in the in context help can be long and complex. You may find that you have to keep jumping from the help to the input panel to fill in each of the input fields correctly.

This chapter is provided as reference to help you use input and expert panels in ToolSet SERVICES. For each panel the following information is provided:

- The panel name forms a running header at the top of each page, to help you locate the panel help that you want
- The panel title
- The option number you enter to display the panel, in brackets below the panel title
- A captured screen of the panel, displaying the valid input combinations
- The in context help for the panel.

Panels are arranged in alphanumeric order of panel name, so that you can find the related help quickly. This means, for example, that panel TW2B100 appears before panel TW22000.

Valid input combinations indicate whether or not you have to enter information in a field. As there are several different combinations of input that you can enter, there are several different columns of symbols. The symbols are as follows:

+ mandatory input - you must fill in input fields marked with a plus symbol

* optional input - you can fill in input fields marked with an asterisk or you can leave them blank

- no input allowed - you must not fill in input fields marked with a dash.

It may seem a little odd to display an input field that you must not fill in, but the information required in the input panel depends upon the information you have already given.

For example, to find out what the following input combinations mean:

```
field 1  -----> + -
field 2  -----> - +
```

read down each column of symbols. When field 1 is mandatory (+) input is not allowed (-) in field 2, and visa versa. This means that you should fill in either one input field or the other, but not both.

To display the valid input combinations for an input panel, enter a question mark (?) in the command area. The key explaining what the symbols mean appears below the input fields.

If you want to find out the option numbers that take you directly to a specific panel or what type of panel to expect when you get there, refer to the navigation charts in Chapter 8, "Navigation Charts," on page 229.

Expert Expert Mode (Option 1.2.E)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
MP-AID EDIT:EXPERT                                   LINE: 00000 OF 00019 WAIT
DICTIONARY:SSADMN STATUS:LNE-DEV                           VIEW: 001 TO 073
====> |
===== *** TOP OF DATA ***
=====     KEEP IN list-name
=====     LIST
=====         HISTORY/ALPHABETICALLY
=====         MEMBERS member-name/member-type/EXCEPT member-type
=====         status-related-selection
=====         FROM 'aaa' TO 'bbb'/ONLY 'aaa'
=====         WHEN
=====             ANY
=====             p TO q
=====             END -q TO END -p
=====                EQ/NE string
=====             LENGTH
=====                EQ/NE/GT/GE/LT/LE
=====         IF AMENDED/INTRODUCED/VERIFIED
=====            ON/BEFORE/AFTER/BETWEEN date AND/ date
=====              AT/BEFORE/AFTER time
=====              BETWEEN time AND time
=====         BY USER user-name
=====     ;
                                                                        V2R
```

Use this mode to enter the syntax of the LIST command. The panel presents different input alternatives to guide you. To complete the template, you must.

- Overwrite all lower case expressions. For example, overwrite member name with the name(s) of actual member(s)
- Delete unwanted alternatives - options separated by a slash (/). For example to select EQ, delete NE/GT/GE/LT/LE
- Blank out any unwanted lines completely
- Terminate the command with a semicolon (;) in the first column position.

Not all alternatives of the LIST command have been used in the panel, as this would require more than one screen page.

Enter `1` in the Command Area for the full syntax of the LIST command. To execute the LIST command having completed the template, enter:

```
RUN
```

in the command line.

Expert Expert Mode (Option 1.3.E)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
MP-AID EDIT:EXPERT                                  LINE: 00000 OF 00028 WAIT
DICTIONARY:SSADMN STATUS:LNE-DEV                          VIEW: 001 TO 073
====> |
===== *** TOP OF DATA ***
=====     KEEP IN list-name
=====     WHICH
=====          status-related-selection
=====          MEMBERS/member-type/EXCEPT member-type
=====          ALPHABETICALLY
=====          FROM 'aaa' TO 'bbb'/ONLY 'aaa'
=====          WHEN
=====              ANY/nn TO mm/END-nn TO END-mm
=====                 EQ/NE string
=====              LENGTH
=====                 EQ/NE/GT/GE/LT/LE
=====              ON/BEFORE/AFTER/BETWEEN date AND/ date
=====                 AT/BEFORE/AFTER time
=====                 BETWEEN time AND time
=====              BY USER user-name
=====              DOES/DOES NOT
=====              HAVE attribute
=====                  SPECIFIED
=====                  EQ/NE/GT/GE/LT/LE value
                                                                      V2R
```

Use this mode to enter the syntax of the WHICH command. The panel presents different input alternatives to guide you. To complete the template, you must:

- Overwrite all lower case expressions. For example, overwrite *status-related-selection* with an actual status-related-selection keyword, such as CURRENT
- Delete all unwanted alternatives - options separated by a slash (/). For example to select EQ, delete NE/GT/GE/LT/LE
- Blank out any unwanted lines completely
- Terminate the command with a semicolon (;) in the first column position.

Not all alternatives of the WHICH command have been used in the panel, as this would require more than one screen page.

To execute the WHICH command having completed the template, enter:

```
RUN
```

in the command line.

Expert Expert Mode (Option 1.4.E)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
MP-AID EDIT:EXPERT                                  LINE: 00000 OF 00028 WAIT
DICTIONARY:SSADMN STATUS:LNE-DEV                          VIEW: 001 TO 073
====> |
===== *** TOP OF DATA ***
=====     KEEP IN list-name
=====     WHICH
=====          status-related-selection
=====          MEMBERS/member-type/EXCEPT member-type
=====          ALPHABETICALLY
=====          FROM 'aaa' TO 'bbb'/ONLY 'aaa'
=====          WHEN
=====              ANY/nn TO mm/END-nn TO END-mm
=====                 EQ/NE string
=====              LENGTH
=====                 EQ/NE/GT/GE/LT/LE
=====              ON/BEFORE/AFTER/BETWEEN date AND/ date
=====                 AT/BEFORE/AFTER time
=====                 BETWEEN time AND time
=====              BY USER user-name
=====              DOES/DOES NOT
=====              HAVE attribute
=====                  SPECIFIED
=====                  EQ/NE/GT/GE/LT/LE value
                                                                      V2R
```

Use this mode to enter the syntax of the WHICH command. The panel presents different input alternatives to guide you. To complete the template, you must:

- Overwrite all lower case expressions. For example, overwrite status-related-selection with an actual status-related-selection keyword, such as CURRENT
- Delete all unwanted alternatives - options separated by a slash (/). For example to select EQ, delete NE/GT/GE/LT/LE
- Blank out any unwanted lines completely
- Terminate the command with a semicolon (;) in the first column position.

Not all alternatives of the WHICH command have been used in the panel, as this would require more than one screen page.

Enter `1` for the full syntax of the WHICH command.

To execute the WHICH command having completed the template, enter:

```
RUN
```

in the RUN command line.

Expert Mode (Option 5.1.E)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
MP-AID EDIT:EXPERT                                  LINE: 00000 OF 00025 WAIT
DICTIONARY:SSADMN STATUS:LNE-DEV                          VIEW: 001 TO 073
====> 
===== *** TOP OF DATA ***
===== PRODUCE
=====     RECORD-LAYOUT AND/FOR language
=====     language
=====     FROM      member name AS library name
=====               ,member name AS library name
=====     ONTO      file name
=====               PARTITONED/SEQUENTIAL
=====     GIVING    output format 1
=====               ,output format 2...
=====     OMITTING  output format 1
=====               ,output format 2...
=====     ALIAS     alias type
=====     USE       format
=====     REPLACE   ALL/string/m n
=====               WITH string WHEN
=====                    ALL/ANY/m n
=====                    EQ/NE string
=====     DROP      ALL/m n/string WHEN
=====                    ALL/ANY/m n
                                                                      V2R
```

In this mode you have to complete the syntax of the PRODUCE command. The panel presents different input alternatives for guidance. You have to:

- Insert parameters by overwriting lower case expressions, e.g. overwrite string with the characters of the required string.
- Delete parts in the lines to exclude alternatives from execution. For example, delete AFTER to insert a string BEFORE certain data names.
- Terminate the command with a semicolon (;).

All these inputs can be made as primary commands in the Command Area as well (for detailed information see *ASG-Manager Products Quick Reference*). To activate the command enter:

```
RUN
```

in the command line.

Usage Frequency of Use (Option 1.2.1)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TD21000                        METHODMANAGER                ToolSet SERVIC

===>

              Usage          Frequency of Use

process existing KEPT-DATA list ----> ____________________________________
member type(s) / relationship(s)----> ____________________________________
member name(s)                  ----> ____________________________________
ER integrity indicator          ----> ____________
dummies                         ----> _
used n times                    ----> ________
uses n members                  ----> ________
keep results in KEPT-DATA list  ----> ____________________________________

 Use this panel to interrogate member types, members and/or dummies as
 to their frequency of use or their relationship with other members.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

process existing KEPT-DATA list

To interrogate only those members held in an existing KEPT-DATA list, enter the name of that KEPT-DATA list.

member type(s)

To interrogate members of a specific member type only, enter the name(s) of those member types. If you enter more than one member type, each must be separated by a comma (,).

member name(s)

To interrogate specific members only, enter the name(s) of those members. If you enter more than one member name, each must be separated by a comma (,).

ER integrity indicator

To select check-ok members, members that have passed integrity checks, enter CHECK-OK. To select check-needed members, members that need integrity checks, enter CHECK-NEEDED. To select both check-ok and check-needed members, enter nothing.

dummies

To search for dummy members (members whose condition is *DUM or *SCE DUM), enter any character other than a blank space.

used n times

To find out how many times the selected members are used via reference clauses from other members, enter a comparand and an integer, separated by a blank space. For example, to find out which members are used less than four times, enter:

```
LT 4
```

The available comparands are as follows:

Comparand	Explanation	Alternative
LT	Less Than	<
LE	Less than or Equal to	
EQ	Equal to	=
GE	Greater than or Equal to	
GT	Greater Than	>
NE	Not Equal to	

uses n member(s)

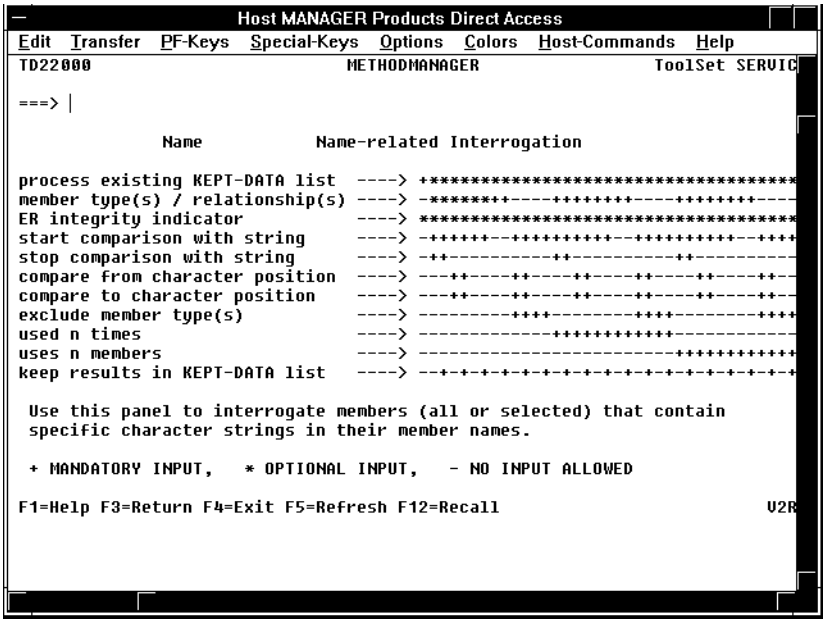
To find out which selected members refer to other members a specific number of times, enter a comparand and an integer, separated by a blank space. For example, to find out which members refer to three other members, enter:

```
=3
```

keep results in KEPT-DATA list

To keep the results of your interrogation in a new named KEPT-DATA list, enter a name for the KEPT-DATA list. If you enter the name of an existing KEPT-DATA list, its contents will be overwritten. If you do not enter a name in this field, the results of your interrogation are displayed on the screen.

Name Name-related Interrogation (Option 1.2.2)



process existing KEPT-DATA list

To interrogate only those members held in an existing KEPT-DATA list, enter the name of that KEPT-DATA list.

member type(s)

To interrogate members of a specific member type only, enter the name(s) of those member types. If you enter more than one member type, each must be separated by a comma (,).

ER integrity indicator

To select check-ok members, members that have passed integrity checks, enter CHECK-OK. To select check-needed members, members that need integrity checks, enter CHECK-NEEDED. To select both check-ok and check-needed members, enter nothing.

start comparison with string

To get an alphabetical list of all members whose member name contains a particular string, enter that string in this field. The string can represent the prefix or any other part of a member name. For example, if you enter the string 'XA' to query a KEPT-DATA list consisting of the members XA-AAA, XA-AAB1, XB-AAC1 and XC-AAD, the members XA-AAA and XA-AAB1 are displayed.

stop comparison with string

To get an alphabetical list of all members whose member name contains a particular string, enter that string in this field. The string can represent the prefix or any other part of a member name. For example, if you enter the string 'XA' for the start and 'XB' for the stop to query a KEPT-DATA list consisting of the members XA-AAA, XA-AAB1, XB-AAC1 and XC-AAD, the members XA-AAA, XA-AAB1, and XB-AAC1 are displayed.

start comparison at character position

To indicate the position in the member name where you want the comparison between string and member name to start, enter an integer in this field. For example, if you enter: 4 to query a member list consisting of the members XA-AAA, XA-AAB1, XB-AAC1, XC-AAD..., the prefixes XA-, XB- and XC- are ignored, and only the subsequent characters are compared with the entered string.

stop comparison at character position

To indicate the position in the member name where you want the comparison between string and member name to stop, enter an integer in this field. This integer must be equal to, or greater than, the integer entered at the start comparison at character position field. For example, if you enter: 4 for the start position, and 6 for the stop position, to query a member list consisting of the members XA-AAA, XA-AAB1, XB-AAC1 and XC-AAD, only the characters AAA, AAB, AAC and AAD are compared with the string.

exclude member type(s)

To exclude any members of specific member type(s) from your query, enter the name(s) of those member types. If you enter more than one member type, each must be separated by a comma (,).

used n times

To find out how many times the selected members are used via reference clauses from other members, enter a comparand and an integer, separated by a blank space. For example, to find out which members are used less than for times, enter:

```
LT 4
```

The available comparands are as follows:

Comparand	Explanation	Alternative
LT	Less Than	<
LE	Less than or Equal to	
EQ	Equal to	=
GE	Greater than or Equal to	
GT	Greater Than	>
NE	Not Equal to	

uses n member(s)

To find out which selected members refer to other members a specific number of times, enter a comparand and an integer, separated by a blank space. For example, to find out which members refer to three other members, enter:

```
= 3
```

keep results in KEPT-DATA list

To keep the results of your interrogation in a new named KEPT-DATA list, enter a name for the KEPT-DATA list.

Caution! If you enter the name of an existing KEPT-DATA list, its contents will be overwritten.

If you do not enter a name in this field, the results of your interrogation are displayed on the screen.

Attribute Interrogate Time, Status and User (Option 1.2.3)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TD23000                        METHODMANAGER                 ToolSet SERVIC

===> 

              Attribute      Interrogate Time, Status and User

process existing KEPT-DATA list ----> +******
member type(s)                  ----> -******
status-related-selection        ----> -******
dummies                         ----> -******
user                            ----> -******
( = < > ) date                  ----> --++-+-
from/till date                  ----> ----+-+
( = < > ) time                  ----> --+----
from/till time                  ----> ---++--
keep results in KEPT-DATA list  ----> -******

Use this panel to interrogate members (all or selective) and refine
the search further by including restrictions such as date of update
or logon-id of user making the change in the selection.
 ------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

process existing KEPT-DATA list

To interrogate only those members held in an existing KEPT-DATA list, enter the name of that KEPT-DATA list.

member type(s)

To interrogate members of a specific member type only, enter the name(s) of those member types. If you enter more than one member type, each must be separated by a comma (,).

status-related-selection

To get information about members in different statuses, enter one of the following status-related-selection keywords:

```
AMENDED            NEW
CHANGED            REVERIFIED
CURRENT            UNVERFIED
DIVERGING
```

A repository can be partitioned into different statuses, based upon one another e.g., a base status ('HIST') would have dependent statuses ('LIVE', 'DEV2'). Working in the dependent status 'LIVE' you can add new members to the repository and redefine members according to your specific needs without changing the version in the base status 'HIST'. Your view of the entries in the repository depends on the status you are working in. In 'HIST' you only see the members of the base status. In 'LIVE' your view includes the members in 'HIST' and the new and amended members in 'LIVE'.

dummies

To search for dummy members (members whose condition is *DUM or *SCE DUM), enter any character other than a blank space.

user
To list all the members last edited by a particular user, enter the authority password of that user. Your repository Controller has access to a complete list of the authority passwords for all the repository users.

(= < >) date
To list all the members that were inserted or encoded before (<), after (>) or on (=) a certain date, enter a comparand and a date (DD.MM.YY).

from/till date
To list all the members that were inserted or encoded within a specific period of time, enter the two dates separated by a space (DD.MM.YY DD.MM.YY).

(= < >) time
To list all the members that were inserted or encoded before (<), after (>) or at (=) a certain time, enter a comparand and a time (HH.MM.SS). If you use this interrogation, you must also complete the date field.

from/till time
To list all the members that were inserted or encoded in within a specific period of time, enter the two times separated by a space (HH.MM.SS HH.MM.SS). If you use this interrogation, you must also complete either the date or the from/till date field.

keep results in KEPT-DATA list
To keep the results of your interrogation in a new named KEPT-DATA list, enter a name for the KEPT-DATA list.

Caution! If you enter the name of an existing KEPT-DATA list, its contents will be overwritten.

If you do not enter a name in this field, the results of your interrogation are displayed on the screen.

Catalog Catalog-related Interrogation (Option 1.2.4)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TD24000                        METHODMANAGER                 ToolSet SERVIC

===> |

              Catalog        Display all CATALOG Classifications

process existing KEPT-DATA list  ----> +************************
start comparison with string     ----> -+++++++--++++++--++--++++
stop comparison with string      ----> -++------++--------++----
compare from character position  ----> ---++------++----++--++--
compare to character position    ----> ---++------++------++++++
used n times                     ----> ---------++++++++--------
keep results in KEPT-DATA list   ----> --+-+-+-+-+-+-+-++-+-+--+

 Use this panel to interrogate Catalogs(all or selected) that contain
 specific character strings in their names.

 ----------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ----------------------------------------------------------



F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

process existing KEPT-DATA list

To interrogate only those Catalogs held in an existing KEPT-DATA list, enter the name of that KEPT-DATA list.

start comparison with string

To get an alphabetical list of all Catalogs whose Catalog name contains a particular string, enter that string in this field. The string can represent any part of a Catalog name. For example, if you enter the string 'AB' to query a KEPT-DATA list consisting of the Catalogs ABA, ABB1, ACC1 and ADD, the Catalogs AAA and AAB1 are displayed.

stop comparison with string

To get an alphabetical list of all Catalogs whose Catalog name contains a particular string, enter that string in this field. The string can represent any part of a catalog name. For example, if you enter the string 'AA' for the start and 'AB' for the stop to query a KEPT-DATA list consisting of the Catalogs AAA, ABB1, ACC1, and ADD, the Catalogs AAA and ABB1 are displayed.

start comparison at character position

To indicate the position in the Catalog name where you want the comparison between string and Catalog name to start, enter an integer in this field. For example, if you enter: 2 to query a Catalog list consisting of the Catalogs AAA, AAB1, AAC1, AAD..., the first character A is ignored, and only the subsequent characters are compared with the entered string.

stop comparison at character position

To indicate the position in the Catalog name where you want the comparison between string and Catalog name to stop, enter an integer in this field. This integer must be equal to, or greater than, the integer entered at the start comparison at character position field. For example, if you enter: 2 for the start position, and 4 for the stop position, to query a Catalog list consisting of the Catalogs VAA, VAB1, VAC1 and VAD, only the characters AA, AB, AC and AD are compared with the string.

used n times

To find out how many members uses the selected Catalogs enter a comparand and an integer, separated by a blank space. For example, to find out which Catalogs are used only one time, enter:

```
EQ 1
```

The available comparands are as follows:

Comparand	Explanation	Alternative
LT	Less Than	<
LE	Less than or Equal to	
EQ	Equal to	=
GE	Greater than or Equal to	
GT	Greater Than	>
NE	Not Equal to	

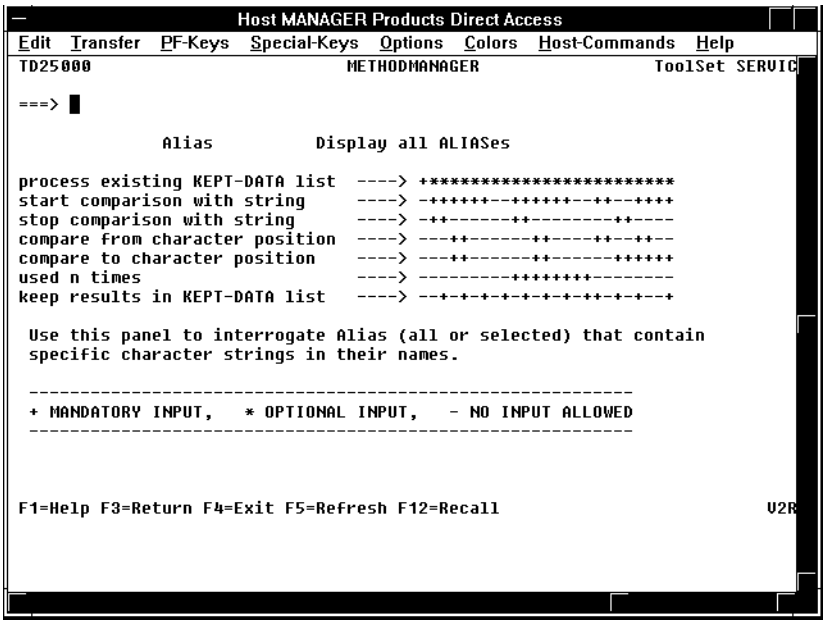
keep results in KEPT-DATA list

To keep the results of your interrogation in a new named KEPT-DATA list, enter a name for the KEPT-DATA list.

Caution! If you enter the name of an existing KEPT-DATA list, its contents will be overwritten.

If you do not enter a name in this field, the results of your interrogation are displayed on the screen.

Alias Alias-related Interrogation (Option 1.2.5)



process existing KEPT-DATA list

To interrogate only those Aliases held in an existing KEPT-DATA list, enter the name of that KEPT-DATA list.

start comparison with string

To get an alphabetical list of all Aliases whose Alias name contains a particular string, enter that string in this field. The string can represent any part of a Alias name. For example, if you enter the string 'AB' to query a KEPT-DATA list consisting of the Aliases ABA, ABB1, ACC1 and ADD, the Aliases ABA and ABB1 are displayed.

stop comparison with string

To get an alphabetical list of all Aliases whose Alias name contains a particular string, enter that string in this field. For example, if you enter the string 'XA' for the start and 'XB' for the stop to query a KEPT-DATA list consisting of the Aliases XAAAA, XAAB1, XBAAC1 and XCAAD, the Aliases XAAAA, XAAB1, and XBAAC1 are displayed.

start comparison at character position

To indicate the position in the Alias name where you want the comparison between string and Alias name to start, enter an integer in this field. For example, if you enter: 2 to query a Alias list consisting of the Aliases AAA, AAB1, AAC1, AAD..., the first character of each Alias will be ignored and only the subsequent characters are compared with the entered string.

stop comparison at character position

To indicate the position in the Alias name where you want the comparison between string and Alias name to stop, enter an integer in this field. This integer must be equal to, or greater than, the integer entered at the start comparison at character position field. For example, if you enter: 2 for the start position, and 4 for the stop position, to query a ALIAS list consisting of the ALIASES VAAA, VAAB1, VAAC1 and VAAD, only the characters AAA, AAB, AAC and AAD are compared with the string.

used n times

To find out how many members uses this Alias clause, enter a comparand and an integer, separated by a blank space. For example, to find out which ALIAS is used more than 1 time, enter `GT 1`.

The available comparands are as follows:

Comparand	Explanation	Alternative
LT	Less Than	<
LE	Less than or Equal to	
EQ	Equal to	=
GE	Greater than or Equal to	
GT	Greater Than	>
NE	Not Equal to	

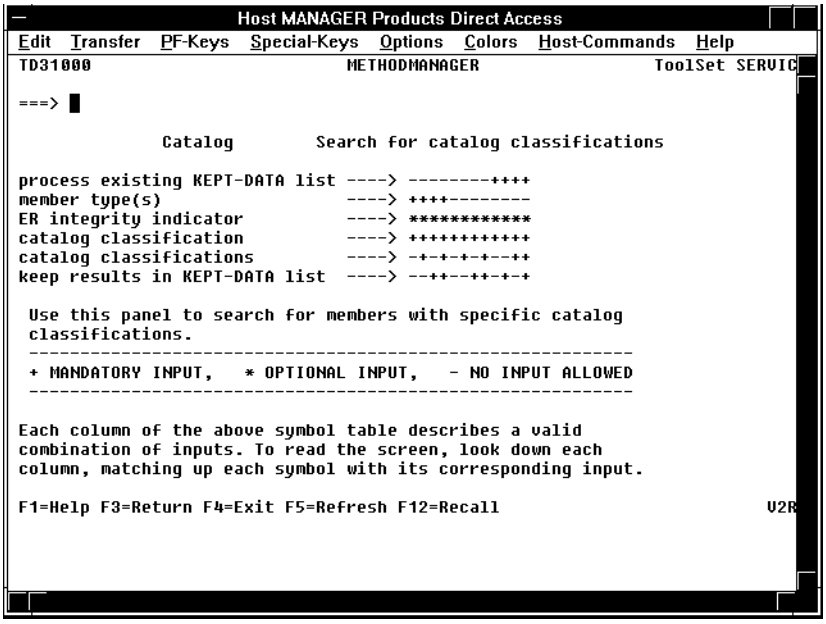
keep results in KEPT-DATA list

To keep the results of your interrogation in a new named KEPT-DATA list, enter a name for the KEPT-DATA list.

If you enter the name of an existing KEPT-DATA list, its contents will be overwritten.

Caution! If you do not enter a name in this field, the results of your interrogation are displayed on the screen.

Catalog Search for catalog classifications (Option 1.3.1)



process existing KEPT-DATA list

To interrogate only those members held in an existing KEPT-DATA list, enter the name of that KEPT-DATA list.

member type(s)

To interrogate members of a specific member type only, enter the name(s) of those member types. If you enter more than one member type, each must be separated by a comma (,).

ER integrity indicator

To select check-ok members, members that have passed integrity checks, enter CHECK-OK. To select check-needed members, members that need integrity checks, enter CHECK-NEEDED. To select both check-ok and check-needed members, enter nothing.

catalog classification

To list the members that contain a specific catalog classification in their member definition, enter that catalog in this field.

catalog classifications
To list the members that contain more than one specific catalog classifications in their member definition, enter those catalogs in this field. Each of the catalog classifications must be separated by comma (,).

keep results in KEPT-DATA list
To keep the results of your interrogation in a new named KEPT-DATA list, enter a name for the KEPT-DATA list.

Caution! If you enter the name of an existing KEPT-DATA list, its contents will be overwritten.

If you do not enter a name in this field, the results of your interrogation are displayed on the screen.

Alias Search for Aliases (Option 1.3.2)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TD32000                        METHODMANAGER                 ToolSet SERVIC

===> |

              Alias          Search for aliases

process existing KEPT-DATA list   ----> +--**
member type(s)                    ----> -++--
ER integrity indicator            ----> +****
alias                             ----> -++++
alias type(s)                     ----> -****
keep results in KEPT-DATA list    ----> --+-+

 Use this panel to search for members with specific aliases.

 ----------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ----------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

process existing KEPT-DATA list
To interrogate only those members held in an existing KEPT-DATA list, enter the name of that KEPT-DATA list.

member type(s)
To interrogate members of a specific member type only, enter the name(s) of those member types. If you enter more than one member type, each must be separated by a comma (,).

ER integrity indicator
To select check-ok members, members that have passed integrity checks, enter CHECK-OK. To select check-needed members, members that need integrity checks, enter CHECK-NEEDED. To select both check-ok and check-needed members, enter nothing.

alias
To list the members that contain a specific alias in their member definition, enter that alias in this field.

alias type(s)
To list the members that contain aliases of a specific alias type (such as SQL, COBOL or PL/I) in their member definition, enter that alias type in this field. If you enter more than one alias type, each must be separated by a comma (,). To get a list of all valid alias types according, enter:

```
SHOW ALIAS-TYPES
```

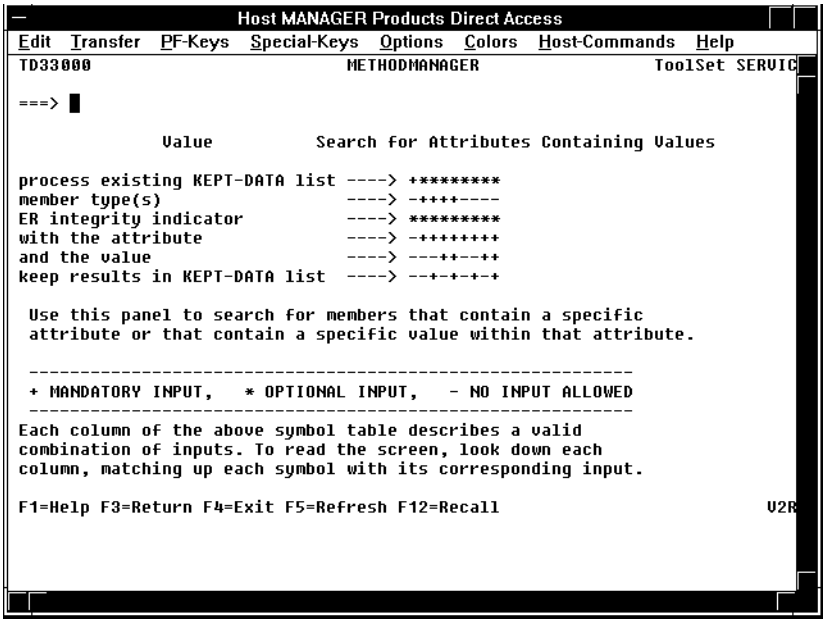
in the Command Area.

keep results in KEPT-DATA list
To keep the results of your interrogation in a new named KEPT-DATA list, enter a name for the KEPT-DATA list.

Caution! If you enter the name of an existing KEPT-DATA list, its contents will be overwritten.

If you do not enter a name in this field, the results of your interrogation are displayed on the screen.

Value Search for Attributes Containing Values (Option 1.3.3)



process existing KEPT-DATA list
To interrogate only those members held in an existing KEPT-DATA list, enter the name of that KEPT-DATA list.

member type(s)
To interrogate members of a specific member type only, enter the name(s) of those member types. If you enter more than one member type, each must be separated by a comma (,).

ER integrity indicator
To select check-ok members, members that have passed integrity checks, enter CHECK-OK. To select check-needed members, members that need integrity checks, enter CHECK-NEEDED. To select both check-ok and check-needed members, enter nothing.

with the attribute
To list all members that contain a specific attribute in their member definition, enter the name of that attribute in this field. Only attributes that do NOT contain text strings are valid in this interrogation. For example 'EFFECTIVE-DATE' and 'AUTHOR' are valid.

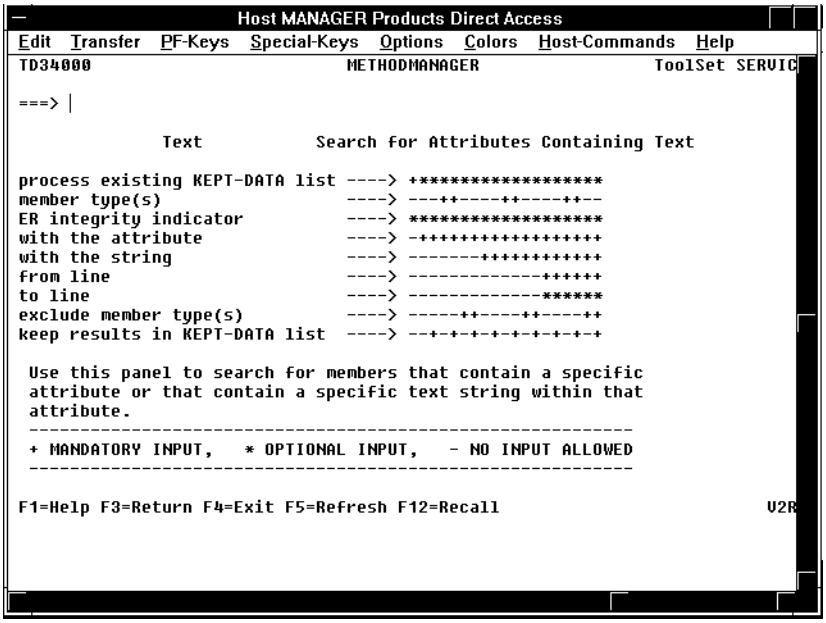
and the value
To limit your interrogation to list only those members that contain a specific value within the attribute entered in the *with the attribute field*, enter that specific value in this field. For example, '900601' for the 'ISSUE-DATE' clause.

keep results In KEPT-DATA list
To keep the results of your interrogation in a new named KEPT-DATA list, enter a name for the KEPT-DATA list.

Caution! If you enter the name of an existing KEPT-DATA list, its contents will be overwritten.

If you do not enter a name in this field, the results of your interrogation are displayed on the screen.

Text Search for Attributes Containing Text (Option 1.3.4)



KEPT-DATA list to process
To interrogate only those members held in an existing KEPT-DATA list, enter the name of that KEPT-DATA list.

member type(s)
To interrogate members of a specific member type only, enter the name(s) of those member types. If you enter more than one member type, each must be separated by a comma (,).

ER integrity indicator
To select check-ok members, members that have passed integrity checks, enter CHECK-OK. To select check-needed members, members that need integrity checks, enter CHECK-NEEDED. To select both check-ok and check-needed members, enter nothing.

with the attribute
To list all members that contain a specific attribute in their member definition, enter the name of that attribute in this field. Only attributes that DO contain text strings are valid in this interrogation. For example 'DESCRIPTION' and 'CONTENTS' are valid.

and the string
To limit your interrogation to list only those members that contain a specific character string within the attribute entered in the *with the attribute* field, enter that specific value in this field.

Note:

The spelling of this string must match exactly with the source in the attribute of the relevant member(s).

from line
To limit your interrogation to specific lines from the attribute entered in the with the attribute field, enter an integer in this field to represent the first line of the attribute that you want to compare with the text in the and the string field. This is only necessary if you do not want the comparison to start in the first line. For example, if the CONTENTS clause consists of several lines of text, enter 2 in this field to start the comparison at the second line of the text.

to line
To limit your interrogation to specific lines from the attribute entered in the with the attribute field, enter an integer in this field to represent the last line of the attribute that you want to compare with the text in the and the string field. This is only necessary if you do not want the comparison to finish on the last line. For example, to stop the comparison in the sixth line of an attribute, enter 6 in this field. This only makes sense if you know that the attribute consists of more than six lines of text.

exclude member type(s)
To exclude specific member type from your interrogation, enter the name(s) of those member types. If you enter more than one member type, each must be separated by a comma (,).

keep results in KEPT-DATA list
To keep the results of your interrogation in a new named KEPT-DATA list, enter a name for the KEPT-DATA list. If you enter the name of an existing KEPT-DATA list, its contents will be overwritten. If you do not enter a name in this field, the results of your interrogation are displayed on the screen.

Dir-Uses Direct References Upwards (Option 1.4.1)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TD41000                        METHODMANAGER                 ToolSet SERVIC

===>

              Dir-Uses       Direct References Upwards

process existing KEPT-DATA list ----> ----++++----++++--++--++
member name(s)                  ----> ++++----++++----++--++--
VIA clauses                     ----> -+-+-+-+-+-+-+-+--------
relationship                    ----> ++++++++++++++++--------
member type(s)                  ----> ++++++++------------++++
keep results in KEPT-DATA list  ----> --++--++--++--++-+-+-+-+

 Use this panel to interrogate a member to find out which other members
 refer to it directly from their member definitions.


 ----------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ----------------------------------------------------------



F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

process existing KEPT-DATA list

To interrogate only those members held in an existing KEPT-DATA list, enter the name of that KEPT-DATA list.

member name(s)

To interrogate specific members only, enter the name(s) of those members. If you enter more than one member name, each must be separated by a comma (,).

VIA clauses

If ASSOCIATED is specified then only EA relationships of the given type are examined. If RELATED is specified then only ER relationships of the given type or of a type belonging to the given class are examined. If CONNECTED is specified then only relationships of the given type or of a type belonging to the given class are examined. If no keyword is given then CONNECTED is assumed.

relationship

To specify the type of relationship you are interested in, enter the name of the relevant relationship clause (such as INPUTS, OUTPUTS or CONTAINS) in this field. Refer to the *ASG-Manager Products Quick Reference* to get a list of all valid relationship types.

member type(s)

To interrogate members of a specific member type only, enter the name(s) of those member types. If you enter more than one member type, each must be separated by a comma (,).

keep results in KEPT-DATA list

To keep the results of your interrogation in a new named KEPT-DATA list, enter a name for the KEPT-DATA list.

Caution! If you enter the name of an existing KEPT-DATA list, its contents will be overwritten.

If you do not enter a name in this field, the results of your interrogation are displayed on the screen.

Uses Indirect References Upwards (Option 1.4.2)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TD42000                        METHODMANAGER                    ToolSet SERVIC

===> |

              Uses           Indirect References Upwards

process existing KEPT-DATA list ----> ----++++----++++--++--++
member name(s)                  ----> ++++----++++----++--++--
VIA clauses                     ----> -+-+-+-+-+-+-+-+--------
relationship                    ----> ++++++++++++++++--------
member type(s)                  ----> ++++++++-------------++++
keep results in KEPT-DATA list  ----> --++--++--++--++-+-+-+-+

 Use this panel to interrogate a member to find out which members
 refer to it either directly or indirectly in their member definitions.
 Several levels of references can be output, the default is 5 levels.

 ----------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ----------------------------------------------------------



F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

process existing KEPT-DATA list

To interrogate only those members held in an existing KEPT-DATA list, enter the name of that KEPT-DATA list.

member name(s)

To interrogate specific members only, enter the name(s) of those members. If you enter more than one member name, each must be separated by a comma (,).

VIA clauses

If ASSOCIATED is specified then only EA relationships of the given type are examined. If RELATED is specified then only ER relationships of the given type or of a type belonging to the given class are examined. If CONNECTED is specified then only relationships of the given type or of a type belonging to the given class are examined. If no keyword is given then CONNECTED is assumed.

relationship

To specify the type of relationship you are interested in, enter the name of the relevant relationship clause (such as INPUTS, OUTPUTS or CONTAINS) in this field. Refer to the *ASG-Manager Products Quick Reference* to get a list of all valid relationship types.

member type(s)

To interrogate members of a specific member type only, enter the name(s) of those member types. If you enter more than one member type, each must be separated by a comma (,).

keep results in KEPT-DATA list

To keep the results of your interrogation in a new named KEPT-DATA list, enter a name for the KEPT-DATA list.

Caution! If you enter the name of an existing KEPT-DATA list, its contents will be overwritten.

If you do not enter a name in this field, the results of your interrogation are displayed on the screen.

Dir-Const Direct References Downwards (Option 1.4.3)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TD43000                          METHODMANAGER                  ToolSet SERVIC

===>

             Dir-Const      Direct References Downwards

process existing KEPT-DATA list ----> ----++++----++++--++--++
member name(s)                  ----> ++++----++++----++--++--
VIA clauses                     ----> -+-+-+-+-+-+-+-+--------
relationship                    ----> ++++++++++++++++--------
member type(s)                  ----> ++++++++------------++++
keep results in KEPT-DATA list  ----> --++--++--++--++-+-+-+-+

 Use this panel to interrogate a member to find out which other members
 it refers to directly from its member definition.


 ----------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ----------------------------------------------------------



F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

process existing KEPT-DATA list

To interrogate only those members held in an existing KEPT-DATA list, enter the name of that KEPT-DATA list.

member name(s)

To interrogate specific members only, enter the name(s) of those members. If you enter more than one member name, each must be separated by a comma (,).

VIA clauses

If ASSOCIATED is specified then only EA relationships of the given type are examined. If RELATED is specified then only ER relationships of the given type or of a type belonging to the given class are examined. If CONNECTED is specified then only relationships of the given type or of a type belonging to the given class are examined. If no keyword is given then CONNECTED is assumed.

relationship

To specify the type of relationship you are interested in enter the name of the relevant relationship clause (such as INPUTS, OUTPUTS or CONTAINS) in this field. Refer to the *ASG-Manager Products Quick Reference* to get a list of all valid relationship types.

member type(s)
To interrogate members of a specific member type only, enter the name(s) of those member types. If you enter more than one member type, each must be separated by a comma (,).

keep results in KEPT-DATA list
To keep the results of your interrogation in a new named KEPT-DATA list, enter a name for the KEPT-DATA list.

Caution! If you enter the name of an existing KEPT-DATA list, its contents will be overwritten.

If you do not enter a name in this field, the results of your interrogation are displayed on the screen.

Constitutes Indirect References Downwards (Option 1.4.4)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TD44000                          METHODMANAGER                 ToolSet SERVIC

===> |

              Constitutes    Indirect References Downwards

process existing KEPT-DATA list ----> ----++++----++++--++--++
member name(s)                  ----> ++++----++++----++--++--
VIA clauses                     ----> -+-+-+-+-+-+-+-+--------
relationship                    ----> ++++++++++++++++--------
member-type(s)                  ----> ++++++++-------------++++
keep results in KEPT-DATA list  ----> --++--++--++--++-+-+-+-+

 Use this panel to interrogate a member to find out which other
 members it refers to either directly or indirectly from its member
 definition. Several levels of references can be output, the default
 is 5 levels.
 ------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ------------------------------------------------------------



F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

process existing KEPT-DATA list
To interrogate only those members held in an existing KEPT-DATA list, enter the name of that KEPT-DATA list.

member name(s)
To interrogate specific members only, enter the name(s) of those members. If you enter more than one member name, each must be separated by a comma (,).

VIA clauses
If ASSOCIATED is specified then only EA relationships of the given type are examined. If RELATED is specified then only ER relationships of the given type or of a type belonging to the given class are examined. If CONNECTED is specified then only relationships of the given type or of a type belonging to the given class are examined. If no keyword is given then CONNECTED is assumed.

relationship

To specify the type of relationship you are interested in, enter the name of the relevant relationship clause (such as INPUTS, OUTPUTS or CONTAINS) in this field. Refer to the *ASG-Manager Products Quick Reference* to get a list of all valid relationship types.

member type(s)

To interrogate members of a specific member type only, enter the name(s) of those member types. If you enter more than one member type, each must be separated by a comma (,).

keep results in KEPT-DATA list

To keep the results of your interrogation in a new named KEPT-DATA list, enter a name for the KEPT-DATA list.

Caution! If you enter the name of an existing KEPT-DATA list, its contents will be overwritten.

If you do not enter a name in this field, the results of your interrogation are displayed on the screen.

Matrix Display Relationships as Matrix (Option 1.4.5)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TD45000                        METHODMANAGER                  ToolSet SERVIC

===> 

              Matrix         Display Relationships as Matrix

KEPT-DATA list (horizontal axis)    ----> +
KEPT-DATA list (vertical axis)      ----> +
maximum name length (vertical AXIS) ----> *
relationship and symbol             ----> +
further relationships               ----> *

 Use this function to display, in matrix form, the relationships between
 the members in two KEPT-DATA lists. You can restrict your interrogation
 to specific relationship clauses only, such as OUTPUTS, CONTAINS or
 CALLS.
 ------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ------------------------------------------------------------
Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

Use this function to display the relationships between the members held in two already existing KEPT-DATA lists, in matrix form. The format of a matrix display is as in the following example:

```
                          MEMBER1
                          |   MEMBER2
                          |   |   MEMBER3
                          .   .   .........
                          |   |   |   MEMBER-m
MEMBERa                   .   I0  .   .
MEMBERb                   I   .   .   I
MEMBERc                   0   0   .   .
MEMBER-n                  10  .   .   I
```

KEPT-DATA list (horizontal axis)

Enter the name of an existing KEPT-DATA list for the horizontal axis of the matrix. (The members from this KEPT-DATA list appear down the left-hand side of the matrix display.)

KEPT-DATA list (vertical axis)

Enter the name of another existing KEPT-DATA list for the vertical axis of the matrix. (The members from this KEPT-DATA list appear from left to right along the top of the matrix display.)

maximum name length (vertical axis)

To define the maximum length of a member name on the vertical axis of the matrix, enter an integer of 32 characters or less. For example, if you enter 8 in this field, the member name 'VG-AUTHOR-NAME' would be shortened to 'VG-AUTHO' on the vertical axis of the matrix.

relationship and type

To query just one type of relationship, enter the valid relationship type and its corresponding symbol in this field using the following syntax:

relationship-clause = matrix-symbol

where:

relationship-clause is the the clause which establishes the type of relationship you are interested in.

matrix-symbol is a one-character string representing the relationship clause in the matrix display.

For example:

```
INPUTS=I
```

further relationships

To query more than one type of relationship, enter the valid relationship types and their corresponding symbols in this field using the syntax defined above. If you enter more than one assignment, each must be separated by a blank space. You can enter up to five assignments.

Merge Combine Two Lists (Option 1.6.5)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TD65000                        METHODMANAGER                 ToolSet SERVICES

===>

              Merge          Combine Two Lists

1st KEPT-DATA list name        ----> ++++
2nd KEPT-DATA list name        ----> ++++
complete contents              ----> ++--
only members appearing in both ----> --++
keep results in KEPT-DATA list ----> -+-+

  Use this panel to list EITHER the complete contents of two KEPT-DATA lists
                     OR only those members that appear in both KEPT-DATA lists.

 ----------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ----------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R2
```

Note:

If you want to merge stored lists make sure that they have been fetched into the current buffer beforehand.

1st KEPT-DATA list

Enter the name of an existing KEPT-DATA list.

2nd KEPT-DATA list

Enter the name of another KEPT-DATA list which you want to merge or compare with the first KEPT-DATA list.

total

To list all members from both KEPT-DATA lists, enter any character other than a blank space.

overlap

To list any members that appear in both KEPT-DATA lists, enter any character except a blank space.

keep results in KEPT-DATA list

To keep the results of your interrogation in a new named KEPT-DATA list, enter a name for the KEPT-DATA list.

Caution! If you enter the name of an existing KEPT-DATA list, its contents will be overwritten.

If you do not enter a name in this field, the results of your interrogation are displayed on the screen.

COBOL Generation of COBOL Structures (Option 5.1.1)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TE11000                        METHODMANAGER                  ToolSet SERVIC

===> |

              COBOL          Generation of COBOL Structures

member name(s)                ----> ++++++++++++++++++++++++++++
ALIAS type                    ----> -+--+--+---+--+---+--+--+--
local name                    ----> --+--+--+---+--+---+--+--+-
prefix                        ----> ---+++----+++--------+++----
suffix                        ----> ------+++----+++-------+++-
replace string                ----> ---------+----------------+
with string                   ----> ---------+----------------+
form and version number       ----> ----------+++++++++++++++++
output-form                   ----> -----------------++++++++++
file name                     ----> ***************************
control card                  ----> ***************************

 Use this panel to create COBOL structures either with or without
 file definition information.
 ------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                V2R
```

This function is used to generate source programming language data descriptions in COBOL from the encoded member definitions held in the repository.

member name(s)

To generate source programming language data descriptions in COBOL enter the names of up to 16 members. If you enter more than one member name, each must be separated by a comma (,).

ALIAS

To include the specific alias of the specified type in the generated member's ALIAS clause, enter an alias type in this field.

local name

To base the generated member names wherever possible on local names from the members' KNOWN-AS clauses instead of on the members' names or aliases, enter any character except a blank space in this field.

prefix

To add a prefix to the generated name, specify that prefix in this field.

suffix

To add a suffix to the generated name, specify that suffix in this field.

replace string

To replace a particular string, specify the member name, or part of the name which is to be replaced, in this field. Alternatively enter two integers, *m* and *p*, which mark the area in the member name which is to be replaced. *m* specifies the character position in a member name, and *p* specifies the number of characters following *m*, which have to be replaced. If m is omitted a value of 1 is defaulted. *m* and *p* have to be separated by at least one blank, and *p* has to be enclosed in brackets ( ). The sum of *m* and *p* must not exceed 97.

with string

To replace the string, specified under the *replace string* field, enter a new string in this field. If you enter single quotes (' ')instead of a string the part of the member name which has been specified under replace string gets deleted.

form and version number

To specify the form and version number of the members which are to be used for generation, enter:

```
ENTERED-AS VERSION @ or  EN V @
HELD-AS VERSION @    or  H V @>
REPORT-AS VERSION @  or  REPO V @
DEFAULTED-AS         or  DE
```

@ represents the version number which has to be an integer in the range 1 to 15. If @ is omitted version 1 is defaulted.

output form

There is a variety of keywords to specify the output form. To get an overview see the *ASG-Manager Products Source Language Generation*.

file name

To specify the external dataset name of the file to which generated program source data descriptions are to be written, enter the name of a logical file as used in job control statements.

The keyword PARTITIONED (abbreviated to PA) or SEQUENTIAL (abbreviated to S) can be added to the file name. Under OS, the keyword PARTITIONED defines the dataset as a BPAM partitioned dataset. Under DOS, if PARTITIONED is stated, it is converted to SEQUENTIAL. SEQUENTIAL defines the dataset as a QSAM sequential dataset.

control card

If the output dataset is SEQUENTIAL, a character string of up to 44 characters enclosed in single quotes, being a library system control card image, can be inserted. Enter a single question mark (?) to indicate the point at which the generated library member name is to be inserted in the control card.

Assembler Generation of Assembler Structures (Option 5.1.2)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TE12000                        METHODMANAGER                 ToolSet SERVIC

===>

              Assembler      Generation of Assembler Structures

member name(s)                ----> ++++++++++++++++++++++++++
ALIAS type                    ----> -+--+--+---+--+--+--+--+--
local name                    ----> --+--+--+---+--+--+--+--+-
prefix                        ----> ---+++----+++------+++----
suffix                        ----> ------+++----+++------+++-
replace string                ----> ---------+---------------+
with string                   ----> ---------+---------------+
form and version number       ----> ----------+++++++++++++++
output form                   ----> ----------------++++++++++
file name                     ----> **************************
control card                  ----> **************************

 Use this panel to create Assembler data structures either with
 or without file definition information.
 ----------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                V2R
```

This function is used to generate source programming language data descriptions in Basic Assembler (BAL) from the encoded member definitions held in the repository.

member name(s)

To generate source programming language data descriptions in Assembler enter the names of up to 16 members. If you enter more than one member name, each must be separated by a comma (,).

ALIAS

To include the specific alias of the specified type in the generated member's ALIAS clause, enter an alias type in this field.

local name

To base the generated data names wherever possible on local names from the members' KNOWN-AS clauses instead of on the members' names or aliases, enter any character except a blank space in this field.

prefix

To add a prefix to the generated name, specify that prefix in this field.

suffix

To add a suffix to the generated name, specify that suffix in this field.

replace string

To replace a particular string, specify the member name, or part of the name which is to be replaced, in this field. Alternatively enter two integers, *m* and *p*, which mark the area in the member name which is to be replaced. *m* specifies the character position in a member name, and *p* specifies the number of characters following *m*, which have to be replaced. If *m* is omitted a value of 1 is defaulted. *m* and *p* have to be separated by at least one blank, and *p* has to be enclosed in brackets ( ). The sum of *m* and *p* must not exceed 97.

with string

To replace the string, specified under the replace string field, enter a new string in this field. If you enter single quotes (' ')instead of a string the part of the member name which has been specified under replace string gets deleted.

form and version number

To specify the form and version number of the members which are to be used for generation, enter:

```
ENTERED-AS VERSION @     or       EN V @
HELD-AS VERSION @        or       H V @>
REPORT-AS VERSION @      or       REPO V @
DEFAuLTED-AS             or       DE
```

@ represents the version number which has to be an integer in the range 1 to 15. If @ is omitted version 1 is defaulted.

output form

There is a variety of keywords to specify the output form. To get an overview see *ASG-Manager Products Source Language Generation*.

file name

To specify the external dataset name of the file to which generated program source data descriptions are to be written, enter the name of a logical file as used in job control statements.

The keyword PARTITIONED (abbreviated to PA) or SEQUENTIAL (abbreviated to S) can be added to the file name. Under OS, the keyword PARTITIONED defines the dataset as a BPAM partitioned dataset. Under DOS, if PARTITIONED is stated, it is converted to SEQUENTIAL. SEQUENTIAL defines the dataset as a QSAM sequential dataset.

control card

If the output dataset is SEQUENTIAL, a character string of up to 44 characters enclosed in single quotes, being a library system control card image, can be inserted. Enter a single question mark (?) to indicate the point at which the generated library member name is to be inserted in the control card.

PL/I Generation of PL/I-Structures (Option 5.1.3)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TE13000                        METHODMANAGER                 ToolSet SERVIC

===> |

              PL/I           Generation of PL/I Structures

member name(s)                 ----> ++++++++++++++++++++++++++
ALIAS type                     ----> -+--+--+---+--+--+--+--+--
local name                     ----> --+--+--+---+--+--+--+--+-
prefix                         ----> ---+++----+++------+++----
suffix                         ----> ------+++----+++------+++-
replace string                 ----> ---------+---------------+
with string                    ----> ---------+---------------+
form and version number        ----> ----------+++++++++++++++
output form                    ----> ----------------++++++++++
file name                      ----> ************************
control card                   ----> ************************

 Use this panel to create PL/I data structures either with
 or without file definition information.
 ----------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                V2R
```

This function is used to generate source programming language data descriptions in PL/I from the encoded member definitions held in the repository.

member name(s)

To generate source programming language data descriptions in PL/I enter the names of up to 16 members. If you enter more than one member name, each must be separated by a comma (,).

ALIAS

To include the specific alias of the specified type in the generated member's ALIAS clause, enter an alias type in this field.

local name

To base the generated data names wherever possible on local names from the members' KNOWN-AS clauses instead of on the members' names or aliases, enter any character except a blank space in this field.

prefix

To add a prefix to the generated name, specify that prefix in this field.

suffix

To add a suffix to the generated name, specify that suffix in this field.

replace string

To replace a particular string, specify the member name, or part of the name which is to be replaced, in this field. Alternatively enter two integers, *m* and *p*, which mark the area in the member name which is to be replaced. *m* specifies the character position in a member name, and *p* specifies the number of characters following *m*, which have to be replaced. If *m* is omitted a value of 1 is defaulted. *m* and *p* have to be separated by at least one blank, and *p* has to be enclosed in brackets (). The sum of *m* and *p* must not exceed 97.

with string

To replace the string, specified under the replace string field, enter a new string in this field. If you enter single quotes (' ')instead of a string the part of the member name which has been specified under replace string gets deleted.

form and version number

To specify the form and version number of the members which are to be used for generation, enter:

```
ENTERED-AS VERSION @    or    EN V @
HELD-AS VERSION @       or    H V @>
REPORT-AS VERSION @     or    REPO V @
DEFAULTED-AS            or    DE
```

@ represents the version number which has to be an integer in the range 1 to 15. If @ is omitted version 1 is defaulted.

output form

There is a variety of keywords to specify the output form. To get an overview see *ASG-Manager Products Source Language Generation*.

file name

To specify the external dataset name of the file to which generated program source data descriptions are to be written, enter the name of a logical file as used in job control statements.

The keyword PARTITIONED (abbreviated to PA) or SEQUENTIAL (abbreviated to S) can be added to the file name. Under OS, the keyword PARTITIONED defines the dataset as a BPAM partitioned dataset. Under DOS, if PARTITIONED is stated, it is converted to SEQUENTIAL. SEQUENTIAL defines the dataset as a QSAM sequential dataset.

control card

If the output dataset is SEQUENTIAL, a character string of up to 44 characters enclosed in single quotes, being a library system control card image, can be inserted. Enter a single question mark (?) to indicate the point at which the generated library member name is to be inserted in the control card.

Record Generation of Language-independent Record Layouts (Option 5.1.4)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TE14000                        METHODMANAGER                 ToolSet SERVIC

===> 

              Record         Generation of Language-Independent Record Layou

member name(s)                 ----> ++++++++++++++++++++++++++
ALIAS type                     ----> -+--+--+---+--+--+--+--+--
local name                     ----> --+--+--+---+--+--+--+--+-
prefix                         ----> ---+++----+++------+++----
suffix                         ----> ------+++----+++------+++-
replace string                 ----> ---------+---------------+
with string                    ----> ---------+---------------+
form and version number        ----> ----------+++++++++++++++
output form                    ----> ----------------++++++++++
file name                      ----> **************************
control card                   ----> **************************

 Use this panel to create data structures both with
 or without file definition information.
 ----------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                      V2R
```

This function is used to generate language-independent record layouts from the encoded member definitions held in the repository.

member name(s)

To generate record layouts, enter the names of up to 16 members. If you enter more than one member name, each must be separated by a comma (,).

ALIAS

To include the specific alias of the specified type in the generated member's ALIAS clause, enter an alias type in this field.

local name

To base the generated data names wherever possible on local names from the members' KNOWN-AS clauses instead of on the members' names or aliases, enter any character except a blank space in this field.

prefix

To add a prefix to the generated name, specify that prefix in this field.

suffix

To add a suffix to the generated name, specify that suffix in this field.

replace string

To replace a particular string, specify the member name, or part of the name which is to be replaced, in this field. Alternatively enter two integers, *m* and *p*, which mark the area in the member name which is to be replaced. *m* specifies the character position in a member name, and *p* specifies the number of characters following *m*, which have to be replaced. If *m* is omitted a value of 1 is defaulted. *m* and *p* have to be separated by at least one blank, and *p* has to be enclosed in brackets ( ). The sum of *m* and *p* must not exceed 97.

with string

To replace the string, specified under the replace string field, enter a new string in this field. If you enter single quotes (' ') instead of a string the part of the member name which has been specified under replace string gets deleted.

form and version number

To specify the form and version number of the members which are to be used for generation, enter:

```
ENTERED-AS VERSION @      or      EN V @
HELD-AS VERSION @         or      H V @>
REPORT-AS VERSION @       or      REPO V @
DEFAULTED-AS              or      DE
```

@ represents the version number which has to be an integer in the range 1 to 15. If @ is omitted version 1 is defaulted.

output form

There is a variety of keywords to specify the output form. To get an overview see *ASG-Manager Products Source Language Generation*.

file name

To specify the external dataset name of the me to which generated program source data descriptions are to be written, enter the name of a logical file as used in job control statements.

The keyword PARTITIONED (abbreviated to PA) or SEQUENTIAL (abbreviated to S) can be added to the file name. Under OS, the keyword PARTITIONED defines the dataset as a BPAM partitioned dataset. Under DOS, if PARTITIONED is stated, it is converted to SEQUENTIAL. SEQUENTIAL defines the dataset as a QSAM sequential dataset.

control card

If the output dataset is SEQUENTIAL, a character string of up to 44 characters enclosed in single quotes, being a library system control card image, can be inserted. Enter a single question mark (?) to indicate the point at which the generated library member name is to be inserted in the control card.

LOADER ADABAS LOADER Definitions (Option 5.2.1.1)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TE21100                        METHODMANAGER                  ToolSet SERVIC

===> |

              LOADER         ADABAS LOADER Definitions


member name(s)          ----> +
ALIAS type              ----> *
local name              ----> *  (any non-blank character selects)
prefix                  ----> *
suffix                  ----> *
replace string          ----> *
with string             ----> *
form and version number ----> *
output form             ----> *
file name               ----> *
control card            ----> *

 Use this panel to create LOADER definitions for the
 ADABAS database.
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

This function is used to produce ADABAS LOADER Definitions from encoded member definitions held in the repository.

member name(s)

To generate ADABAS LOADER Definitions enter the names of up to 16 members. These have to be encoded SYSTEM, PROGRAM, or MODULE members which contain a PROCESS ADABAS clause. If you enter more than one member name, each must be separated by a comma (,).

ALIAS

To include the specific alias of the specified type in the generated member's ALIAS clause, enter an alias type in this field.

local name

To base the generated data names wherever possible on local names from the members' KNOWN-AS clauses instead or on the members' names or aliases, enter any character except a blank space in this field.

prefix

To add a prefix to the generated name, specify that prefix in this field.

suffix

To add a suffix to the generated name, specify that suffix in this field.

replace string

To replace a particular string, specify the member name, or part of the name which is to be replaced, in this field. Alternatively enter two integers, *m* and *p*, which mark the area in the member name which is to be replaced. *m* specifies the character position in a member name, and *p* specifies the number of characters following *m*, which have to be replaced. If *m* is omitted a value of 1 is defaulted. *m* and *p* have to be separated by at least one blank, and *p* has to be enclosed in brackets ( ). The sum of *m* and *p* must not exceed 97.

with string

To replace the string, specified under the replace string field, enter a new string in this field. If you enter single quotes (' ') instead of a string the part of the member name which has been specified under replace string gets deleted.

form and version number

To specify the form and version number of the members which are to be used for generation, enter:

```
ENTERED-AS VERSION @      or  EN V @
HELD-AS VERSION @         or  H V @>
REPORT-AS VERSION @       or  REPO V @
DEFAULTED-AS              or  DE
```

@ represents the version number which has to be an integer in the range 1 to 15. If @ is omitted version 1 is defaulted.

output form

There is a variety of keywords to specify the output form. To get an overview see *ASG-Manager Products Source Language Generation*.

file name

To specify the external dataset name of the file to which generated program source data descriptions are to be written, enter the name of a logical file as used in job control statements.

The keyword PARTITIONED (abbreviated to PA) or SEQUENTIAL (abbreviated to S) can be added to the file name. Under OS, the keyword PARTITIONED defines the dataset as a BPAM partitioned dataset. Under DOS, if PARTITIONED is stated, it is converted to SEQUENTIAL. SEQUENTIAL defines the dataset as a QSAM sequential dataset.

control card

If the output dataset is SEQUENTIAL, a character string of up to 44 characters enclosed in single quotes, being a library system control card image, can be inserted. Enter a single question mark (?) to indicate the point at which the generated library member name is to be inserted in the control card.

Structure ADABAS Record Layouts (Option 5.2.1.2)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TE21200                        METHODMANAGER                  ToolSet SERVIC

===>

              Structure      ADABAS Record Layouts


member name(s)                ----> +
ALIAS type                    ----> *
local name                    ----> *  (any non-blank character selects)
prefix                        ----> *
suffix                        ----> *
replace string                ----> *
with string                   ----> *
form and version number       ----> *
output from                   ----> *
file name                     ----> *
control card                  ----> *

 Use this panel to create structure descriptions for the
 ADABAS database.
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

This function is used to produce record layouts for ADABAS files and buffers from the encoded member definitions held in the repository.

member name(s)

To generate ADABAS record layouts enter the names of up to 16 members. These have to be encoded SYSTEM, PROGRAM, or MODULE members which contain a PROCESS ADABAS clause. If you enter more than one member name, each must be separated by a comma (,).

ALIAS

To include the specific alias of the specified type in the generated member's ALIAS clause, enter an alias type in this field.

local name

To base the generated data names wherever possible on local names from the members' KNOWN-AS clauses instead of on the members' names or aliases, enter any character except a blank space in this field.

prefix

To add a prefix to the generated name, specify that prefix in this field.

suffix

To add a suffix to the generated name, specify that suffix in this field.

replace string

To replace a particular string, specify the member name, or part of the name which is to be replaced, in this field. Alternatively enter two integers, *m* and *p*, which mark the area in the member name which is to be replaced. *m* specifies the character position in a member name, and *p* specifies the number of characters following *m*, which have to be replaced. If *m* is omitted a value of 1 is defaulted. *m* and *p* have to be separated by at least one blank, and *p* has to be enclosed in brackets (). The sum of *m* and *p* must not exceed 97.

with string

To replace the string, specified under the replace string field, enter a new string in this field. If you enter single quotes (' ') instead of a string the part of the member name which has been specified under replace string gets deleted.

form and version number

To specify the form and version number of the members which are to be used for generation, enter:

```
ENTERED-AS VERSION @     or     EN V @
HELD-AS VERSION @        or     H V @>
REPORT-AS VERSION @      or     REPO V @
DEFAULTED-AS             or     DE
```

@ represents the version number which has to be an integer in the range 1 to 15. If @ is omitted version 1 is defaulted.

output form

There is a variety of keywords to specify the output form. To get an overview see *ASG-Manager Products Source Language Generation*.

file name

To specify the external dataset name of the file to which generated program source data descriptions are to be written, enter the name of a logical file as used in job control statements.

The keyword PARTITIONED (abbreviated to PA) or SEQUENTIAL (abbreviated to S) can be added to the file name. Under OS, the keyword PARTITIONED defines the dataset as a BPAM partitioned dataset. Under DOS, if PARTITIONED is stated, it is converted to SEQUENTIAL. SEQUENTIAL defines the dataset as a QSAM sequential dataset.

control card

If the output dataset is SEQUENTIAL, a character string of up to 44 characters enclosed in single quotes, being a library system control card image, can be inserted. Enter a single question mark (?) to indicate the point at which the generated library member name is to be inserted in the control card.

Buffer Record and Format Buffer (Option 5.2.1.3)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TE21300                        METHODMANAGER                  ToolSet SERVIC

===> |

              Buffer         Record and Format Buffer


member name(s)                ----> +
record layout                 ----> *  (any character selects)
data description              ----> +
language                      ----> *
ALIAS type                    ----> *
local name                    ----> *
prefix                        ----> *
suffix                        ----> *
form and version number       ----> *
output form                   ----> *
file name                     ----> *
control card                  ----> *

 Use this panel to create buffer structures for the
 ADABAS database.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

This function is used to produce description statements for ADABAS record buffers and format buffers in COBOL, PL/I, or Assembler from the encoded member definitions held in the repository.

member name(s)

To generate ADABAS record buffers enter the names of up to 16 members. These have to be encoded SYSTEM, PROGRAM, or MODULE members which contain a PROCESS ADABAS clause. If you enter more than one member name, each must be separated by a comma (,).

record layout

To create a record layout, enter any character other than a blank space.

data description statement

To specify the buffer for which data description statements are to be generated, enter:

```
RECORD-BUFFERS (or R)

FORMAT-BUFFERS (or F)

BUFFERS (or B).
```

If you have made an entry in the *record layout* field as well, then both record layouts and data description statements will be generated for the appropriate buffer.

language

To determine the programming language in which data descriptions are to be generated, enter ALC, Assembler, BAL, COBOL, PL/I, PL/l, PLI, or PL1.

ALIAS
To include the specific alias of the specified type in the generated member's ALIAS clause, enter an alias type in this field.

local name
To base the generated data names wherever possible on local names from the members' KNOWN-AS clauses instead of on the members' names or aliases, enter any character except a blank space in this field.

prefix
To add a prefix to the generated name, specify that prefix in this field.

suffix
To add a suffix to the generated name, specify that suffix in this field.

replace string
To replace a particular string, specify the member name, or part of the name which is to be replaced, in this field. Alternatively enter two integers, *m* and *p*, which mark the area in the member name which is to be replaced. *m* specifies the character position in a member name, and *p* specifies the number of characters following *m*, which have to be replaced. If *m* is omitted a value of 1 is defaulted. *m* and *p* have to be separated by at least one blank, and *p* has to be enclosed in brackets ( ). The sum of *m* and *p* must not exceed 97.

with string
To replace the string, specified under the replace string field, enter a new string in this field. If you enter single quotes (' ') instead of a string the part of the member name which has been specified under replace string gets deleted.

form and version number
To specify the form and version number of the members which are to be used for generation, enter:

```
ENTERED-AS VERSION @     or     EN V @
HELD-AS VERSION @        or     H V @>
REPORT-AS VERSION @      or     REPO V @
DEFAULTED-AS             or     DE
```

@ represents the version number which has to be an integer in the range 1 to 15. If @ is omitted version 1 is defaulted.

output form
There is a variety of keywords to specify the output form. To get an overview see *ASG-Manager Products Source Language Generation*.

file name
To specify the external dataset name of the file to which generated program source data descriptions are to be written, enter the name of a logical file as used in job control statements.

The keyword PARTITIONED (abbreviated to PA) or SEQUENTIAL (abbreviated to S) can be added to the file name. Under OS, the keyword PARTITIONED defines the dataset as a BPAM partitioned dataset. Under DOS, if PARTITIONED is stated, it is converted to SEQUENTIAL. SEQUENTIAL defines the dataset as a QSAM sequential dataset.

control card

If the output dataset is SEQUENTIAL, a character string of up to 44 characters enclosed in single quotes, being a library system control card image, can be inserted. Enter a single question mark (?) to indicate the point at which the generated library member name is to be inserted in the control card.

DBD Produce DATABASE DESCRIPTION (DBD) (Option 5.2.2.1)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TE22100                        METHODMANAGER                  ToolSet SERVIC

===>

              DBD            Produce DATABASE DESCRIPTION (DBD)


member name(s)                ----> +
type of field                 ----> +
ALIAS type                    ----> *
prefix                        ----> *
suffix                        ----> *
file name                     ----> *
IMS version                   ----> *

 Use this panel to create database definitions for IMS databases.
 ----------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ----------------------------------------------------------
Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

This function is used to produce IMS (DL/I) database description control statements from members held in the repository.

member name(s)

To generate IMS (DL/I) database description control statements enter the name(s) of the relevant member(s). These have to be encoded IMS-DATABASE or DL/I-DATABASE members. If you enter more than one member name, each must be separated by a comma (,).

type of field

To specify which types of DBD FIELD control statements you want to generate, enter one of the following:

```
SEARCH-FIELDS (or S)

DIRECT-FIELDS (or D)

GENERATES-FIELDS (or GE)

ALL-FIELDS (or ALL).
```

ALIAS

To include the specific alias of the specified type in the generated member's ALIAS clause, enter an alias type in this field.

prefix

To add a prefix to the generated name, specify that prefix in this field.

suffix

To add a suffix to the generated name, specify that suffix in this field.

file name

To specify the external dataset name of the file to which generated program source data descriptions are to be written, enter the name of a logical file as used in job control statements.

The keyword PARTITIONED (abbreviated to PA) or SEQUENTIAL (abbreviated to S) can be added to the file name. Under OS, the keyword PARTITIONED defines the dataset as a BPAM partitioned dataset. Under DOS, if PARTITIONED is stated, it is converted to SEQUENTIAL. SEQUENTIAL defines the dataset as a QSAM sequential dataset.

version

To over-ride the target IMS system release level enter a value in this field. Valid values are 1.2, 1.3, 2, 2.0, 2.1, 2.2, 3, 3.0, and 3.1.

PSB Produce PROGRAM SPECIFICATION BLOCK (PSB) (Option 5.2.2.2)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TE22200                        METHODMANAGER                  ToolSet SERVIC

===> |

              PSB            Produce PROGRAM SPECIFICATION BLOCK (PSB)


member name(s)        ----> +
ALIAS type            ----> *
prefix                ----> *
suffix                ----> *
file name             ----> *
options               ----> *
IMS version           ----> *

 Use this panel to create program views of IMS databases.
 ---------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ---------------------------------------------------------
Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

This function is used to produce IMS (DL/I) program specification block control statements from encoded members held in the repository.

member name(s)

To generate IMS (DL/I) program specification block control statements enter the name(s) of the relevant member(s). These have to be encoded members of the type SYSTEM, PROGRAM, or MODULE. If you enter more than one member name, each must be separated by a comma (,).

ALIAS

To include the specific alias of the specified type in the generated member's ALIAS clause, enter an alias type in this field.

prefix

To add a prefix to the generated name, specify that prefix in this field.

suffix

To add a suffix to the generated name, specify that suffix in this field.

file name

To specify the external dataset name of the file to which generated program source data descriptions are to be written, enter the name of a logical file as used in job control statements.

The keyword PARTITIONED (abbreviated to PA) or SEQUENTIAL (abbreviated to S) can be added to the file name. Under OS, the keyword PARTITIONED defines the dataset as a BPAM partitioned dataset. Under DOS, if PARTITIONED is stated, it is converted to SEQUENTIAL SEQUENTIAL defines the dataset as a QSAM sequential dataset.

options
To specify additional information such as IO-PCB requirement, IO-SIZE, SSA-SIZE, PCB-LABELS/NAMES, ENQUES, CONDITION-CODE/WTOR, or OLIC enter the required parameters in this field.

version
To over-ride the target IMS system release level enter a value in this field. Valid values are 1.2, 1.3, 2, 2.0, 2.1, 2.2, 3, 3.0, and 3.1.

I/O Area Produce Segment I/O Area (Option 5.2.2.3)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TE22300                        METHODMANAGER                ToolSet SERVIC

===>

              I/O Area       Produce Segment I/O Area


member name(s)      ----> +
language            ----> +
ALIAS type          ----> *
prefix              ----> *
suffix              ----> *
file name           ----> *

 Use this panel to create data structures for the IMS database.
 ------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ------------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

This function is used to generate COBOL, PL/I, or Assembler data description statements and/or record layouts for segment I/O areas from encoded members held in the repository.

member name(s)
To generate record layouts for segment I/O areas, enter the name(s) of the relevant member(s). These have to be encoded SEGMENT members. If you enter more than one member name, each must be separated by a comma (,).

language
To determine the programming language in which data structures are to be generated, enter ALC, ASSEMBLER, ASSEMBLY, BAL, COBOL, PL/I, PL/1, PLI, or PL1.

ALIAS
To include a specific alias of the specified type in the generated member's ALIAS clause, enter an alias type in this field.

prefix
To add a prefix to the generated name, specify that prefix in this field.

suffix
To add a suffix to the generated name, specify that suffix in this field.

file name
To specify the external dataset name of the file to which generated program source data descriptions are to be written, enter the name of a logical file as used in job control statements.

The keyword PARTITIONED (abbreviated to PA) or SEQUENTIAL (abbreviated to S) can be added to the file name. Under OS, the keyword PARTITIONED defines the dataset as a BPAM partitioned dataset. Under DOS, if PARTITIONED is stated, it is converted to SEQUENTIAL. SEQUENTIAL defines the dataset as a QSAM sequential dataset.

SESAM SESAM Database System (Option 5.2.3)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TE23000                        METHODMANAGER                    ToolSet SERVIC

===> |

              SESAM          SESAM Database System

member name                    ----> +
ALIAS                          ----> *
form and version number        ----> *
prefix                         ----> *
suffix                         ----> *
symbolic name                  ----> *
work disk                      ----> *
save disk                      ----> *
compound key name              ----> *
file name                      ----> *

 Use this panel to create data structures for the SESAM database.
 ----------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ----------------------------------------------------------


F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

This function is used to produce a Siemens BS2000 procedure to build up a SESAM database from encoded member definitions held in the repository.

member name
To build up a procedure that creates a SESAM database, enter the name of the relevant member. This has to be an encoded member of the type SESAM-STORAGE.

ALIAS
To include a specific alias of the specified type in the generated member's ALIAS clause, enter an alias type in this field.

form and version number

To specify the form and version number of the members which are to be used for generation, enter:

```
ENTERED-AS VERSION @    or      EN V @
HELD-AS VERSION @       or      H V @>
REPORT-AS VERSION @     or      REPO V @
DEFAULTED-AS            or      DE
```

@ represents the version number which has to be an integer in the range 1 to 15. If @ is omitted version 1 is defaulted.

prefix

To add a prefix to the generated name, specify that prefix in this field.

suffix

To add a suffix to the generated name, specify that suffix in this field.

symbolic name

To prevent a symbolic name for ITEM members being output in the attribute definition clauses for SESASB, enter `N`.

work disk

Enter the name of a private disk model (device) and the volume (volume) of the disk. If you enter more than one volume, each must be separated by a comma (,). (device) is one of the following:

```
D458        D3460        D5801
D4580       D3465        D5802
D3440       D3468        D5804
D3450       D3470        D5881
D3455       D580         D5882
```

(volume) is the volume of a private disk complying with the rules stated in the *Siemens BS2000 Command Language* manual.

save disk

To generate control statements for SESASB and create a backup of the first database on the specified device and volume, enter the name of a private disk model followed by its volume.

compound key name

Enter the name which is to be used for a SESAM compound key. Only complete this field if the SESAM-STORAGE member has an ITEM list in its ACCESS-IS clause.

file name

To specify the name of the output source library dataset (this is the logical file name used in job control statements to indicate the external dataset name of the file, to which generated program source data descriptions are to be written) enter that name in this field.

Note:

If you do not enter a name in this field, NOGENERATION is defaulted, and no output file is written.

SUDS SUDS Database (Option 5.2.4)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TE24000                        METHODMANAGER                  ToolSet SERVIC

===> 

               SUDS           SUDS datababase

specify statement              ----> +
member name                    ----> +
ALIAS                          ----> *
prefix                         ----> *
suffix                         ----> *
form and version number        ----> *
condition name                 ----> *
file name                      ----> *

 Use this panel to create data structures for the SUDS database.
 ---------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ---------------------------------------------------------------
Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

This function is used to produce Siemens Universal Database System (SUDS) statements of the following types from encoded member definitions held in the repository:

- Schema Storage Structure Language (SSL)
- Schema Data Description Language (DDL)
- Subschema Data Description Language (DDL).

specify statement

To specify the type of statement which is to be generated, enter one of the following:

```
SSL (or SS)

SCHEMA-DDL (or SC)

SUBSCHEMA-DDL (or SU).
```

member name

To generate SSL or SCHEMA-DDL statements, enter the name of an encoded member of the type SUDS-DATABASE. To generate SUBSCHEMA-DDL statements, enter the name of an encoded SUDS-SUBSCHEMA member.

ALIAS

To include the specific alias of the specified type in the generated member's ALIAS clause, enter an alias type in this field.

form and version number

To specify the form and version number of the members which are to be used for generation, enter:

```
ENTERED-AS VERSION @    or      EN V @
HELD-AS VERSION @       or      H V @>
REPORT-AS VERSION @     or      REPO V @
DEFAULTED-AS            or      DE
```

@ represents the version number which has to be an integer in the range 1 to 15. If @ is omitted version 1 is defaulted.

prefix

To add a prefix to the generated name, specify that prefix in this field.

suffix

To add a suffix to the generated name, specify that suffix in this field.

condition name

This selection is only valid when generating Subschema Data Description Language. To generate level 88 statements from the CONDITION-NAMES clauses of ITEM members, enter `Y`. To suppress this generation, enter `N`.

file name

To specify the name of the output source library dataset (this is the logical file name used in job control statements to indicate the external dataset name of the file, to which generated program source data descriptions are to be written) enter that name in this field.

Note:

If you do not enter a name in this field, NOGENERATION is defaulted, and no output file is written.

DB2 Admin DB2 Database Administration (Option 5.3.1.1)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Help
TE31100                    METHODMANAGER                ToolSet SERVICES

 ===>

              DB2 Admin      DB2 Database Administration

 member name                 ----> +++++
 SQL statement: CREATE       ----> +----
                DROP         ----> -+---
                COMMENT      ----> --+--
                LABEL        ----> ---+-
 estimate object size        ----> ----+
 expand nested data? (Y/N)   ----> *-***
 SQL ID of object            ----> *-**-
 location of object          ----> *-**-
 user exit routine name      ----> *****
 suppress ref. integrity     ----> *----
 no impact analysis report   ----> -*---
 output options              ----> *****
 ------------------------------------------------------------------------------
 +  MANDATORY INPUT,   * OPTIONAL INPUT,  - NO INPUT ALLOWED
F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                      V2R300
```

Use this function to produce SQL statements for the administration of a DB2 Database Management System. You can:

- Generate SQL CREATE, DROP, COMMENT ON and LABEL ON statements
- Produce impact analysis reports for SQL DROP statements
- Produce an estimate of the size of an index, or a table and its rows, and so estimate storage space before generating an SQL CREATE statement.

member name

Enter the name of a member belonging to one of the following types:

- DB2-TABLE (for all SQL statements)
- DB2-VIEW (for all SQL statements)
- DB2-ALIAS (for all SQL statements)
- DB2-DATABASE (for SQL CREATE and DROP statements)
- DB2-STOGROUP (for SQL CREATE and DROP statements)
- DB2-TBSPACE (for SQL CREATE and DROP statements)
- DB2-INDEX (for SQL CREATE and DROP statements)

from which the required SQL statement is to be generated.

SQL statement: CREATE/ DROP/ COMMENT/ LABEL

To generate an appropriate SQL statement for a DB2 object from its definition in a repository member, enter any character other than a blank space.

When generating an SQL DROP statement, an impact analysis report is also generated, displaying the effect on your DB2 environment.

estimate object size

To calculate the total size of either:

- A table and the size of each row contained in that table
- An index

from the definition of a DB2-TABLE or DB2-INDEX member, enter any character other than a blank space.

expand nested data

For an SQL CREATE statement, you can expand (or suppress expansion of) GROUP members (documenting columns), contained in DB2-TABLE, DB2-INDEX, or DB2-VIEW members. To expand these nested data structures, enter `y`. To suppress expansion, enter `n`.

For other SQL statements, if you previously expanded (or suppressed expansion of) nested data structures, you should enter `y` or `n` as appropriate.

SQL ID of object

To generate an SQL statement for a table with a specific owner, enter the authorization ID of a specified user.

location of object

To generate an SQL statement for a table at a specific location, enter the location name.

routine name

To take a user exit, enter the name of your user exit routine.

suppress referential integrity

To suppress generation of any referential integrity clauses (foreign keys) in a DB2-TABLE member, enter any character other than a blank space.

no impact analysis report

To prevent an impact analysis report being given when generating an SQL DROP statement, enter any character other than a blank space. There is no impact when generating an SQL DROP statement for a DB2-INDEX, so no impact analysis report is given.

output options

To send your generated output to a specific destination, or to suppress all output except messages and impact analysis reports, enter any non-blank character in this field.

When the ENTER key is pressed, an additional panel (TE31101) will be displayed on which you can specify these options.

DB2 Admin DB2 Database Administration Output Options

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Help

TE31101                        METHODMANAGER          ToolSet SERVICES

 ===>

               DB2 Admin      DB2 Database Administration

                              Output Options

 output to: private user       ----> +----
            public user        ----> -+---
            PDS member         ----> --+--
            sequential file    ----> ---+-
 new/append/replace user       ----> ++---
 new/replace PDS member        ----> --+--
 suppress print                ----> *****
 ---------------------------------------------------------------------------
 +  MANDATORY INPUT,   * OPTIONAL INPUT,  - NO INPUT ALLOWED
 ---------------------------------------------------------------------------
Each column of the above symbol table describes a valid combination of inputs.  To
read the screen, look down each column, matching up each symbol with its
corresponding input
 F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R300
```

Use this panel to specify output options for panel TE31100.

output to: private user/public user/PDS member/sequential file

To send your generated output to a specific destination, enter the destination in one of these input fields.

To send output to a private or public USER-MEMBER on the MP-AID, enter the USER-MEMBER name in the *private user* or *public user* field.

To send output directly to an (existing) partitioned dataset (PDS) enter the name of a PDS member in the *PDS member field*.

To send output directly to a sequential file, enter any character other than a blank space in the *sequential file field*.

new/append/replace member

To file the generated output in a new USER-MEMBER, enter `n` or `new`. To append the generated output in an existing USER-MEMBER, enter `a` or `append`. To replace the contents of an existing USER-MEMBER, enter *`r`* or *`replace`*.

new/replace PDS member

To file the generated output in a new PDS member, enter *`n`* or *`new`*. To replace the contents of an existing PDS member, enter *`r`* or *`replace`*.

suppress print

To output only messages and impact analysis reports, enter any character other than a blank space.

Alter Generate SQL ALTER TABLE Statements (Option 5.3.1.2)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Help
TE31200                    METHODMANAGER          ToolSet SERVICES

 ===>

              Alter          Generate SQL ALTER TABLE Statements

 member name                ----> +++
 add to table               ----> +--
 change auditing options    ----> **+
 drop from table            ----> -+-
 expand nested data? (Y/N)  ----> ***
 SQL ID of object           ----> ***
 location of table          ----> ***
 specified columns          ----> *--
 primary key                ----> **-
 referential constraint     ----> **-
 validation procedure       ----> **-
 user exit routine name     ----> ***
 output options             ----> ***
 ------------------------------------------------------------------------------
  +  MANDATORY INPUT,   * OPTIONAL INPUT,  - NO INPUT ALLOWED
 F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                      V2R300
```

You can generate SQL ALTER TABLE statements from DB2-TABLE repository members, to alter specified DB2 tables. By accurately documenting the DB2 tables with repository member definitions, you can maintain these tables with SQL statements generated from the definitions.

member name
Enter the name of an encoded DB2-TABLE member.

add to table
To generate an SQL ALTER TABLE statement to add certain objects (to be selected further down in this panel) to a table, enter any character other than a blank space.

change auditing options
To generate an SQL ALTER TABLE statement to add or modify the auditing option on a table, enter any character other than a blank space.

drop from table
To generate an SQL ALTER TABLE statement to drop certain options (to be selected further down in this panel) from a table, enter any character other than a blank space.

expand nested data
If nested data structures for the table were expanded when the table was created, enter `y`. If expansion was suppressed, enter `n`. EXPAND clauses used in the DB2-TABLE member are used as the default.

SQL ID of object
To generate an SQL ALTER TABLE statement for a table with a specific owner, enter the authorization ID of a specified user.

location of object
To generate an SQL ALTER TABLE statement for a table at a specific location, enter the location name.

specified columns
To specify the columns to be added to the tables, enter any integer in the range 1 to 299. The columns to be added are the last 'n' columns derived from the COLUMNS clause of the DB2-TABLE member from which the SQL ALTER TABLE statement is being generated. 'n' must be less than the total number of columns derived from the COLUMNS clause.

primary key
To add or drop (depending on your previous selection) a primary key on a table, enter any character other than a blank space. Primary key columns are defined with the PRIMARY-KEY clause of the DB2-TABLE member. All columns in the DB2-TABLE that have an associated PRIMARY-KEY keyword are generated as part of the primary key.

referential constraint
To add or drop (depending on your previous selection) a referential constraint on a table, enter either of the following:

- The name of the constraint as specified in the NAMED clause of the DB2-TABLE member
- A number identifying the constraint by its sequence among other referential constraints in the DB2-TABLE member.

validation procedure
To add or drop (depending on your previous selection) a validation routine on a table, enter any character other than a blank space. An SQL statement to add a validation routine, when submitted to DB2, modifies any existing validation routine on that table.

user exit routine name
To take a user exit, enter the name of your user exit routine.

output options
To send your generated output to a specific destination, or to suppress all output except messages and impact analysis reports, enter any non-blank character in this field.

When the ENTER key is pressed, an additional panel (TE31201) will be displayed on which you can specify these options.

DB2 Admin DB2 Database Administration Output Options

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Help
TE31201                    METHODMANAGER                ToolSet SERVICES

 ===>

              Alter          Generate SQL ALTER TABLE Statements

                             Output Options

 output to: private user       ----> +----
            public user        ----> -+---
            PDS member         ----> --+--
            sequential file    ----> ---+-
 new/append/replace user       ----> ++---
 new/replace PDS member        ----> --+--
 suppress print                ----> *****
 ---------------------------------------------------------------------------
 +  MANDATORY INPUT,   * OPTIONAL INPUT,  - NO INPUT ALLOWED
 ---------------------------------------------------------------------------
Each column of the above symbol table describes a valid combination of inputs.  To
read the screen, look down each column, matching up each symbol with its
corresponding input

 F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                      V2R300
```

Use this panel to specify output options for panel TE31200.

output to: private user/public user/PDS member/sequential file

To send your generated output to a specific destination, enter the destination in one of these input fields.

To send output to a private or public USER-MEMBER on the MP-AID, enter the USER-MEMBER name in the *private user* or *public user* field.

To send output directly to an (existing) partitioned dataset (PDS) enter the name of a PDS member in the *PDS member* field.

To send output directly to a sequential file, enter any character other than a blank space in the *sequential file* field.

new/append/replace member

To file the generated output in a new USER-MEMBER, enter `n` or `new`.

To append the generated output in an existing USER-MEMBER, enter `a` or `append`.

To replace the contents of an existing USER-MEMBER, enter `r` or `replace`.

new/replace PDS member

To file the generated output in a new PDS member, enter `n` or `new`.

To replace the contents of an existing PDS member, enter `r` or `replace`.

suppress print

To print out only messages and impact analysis reports, enter any character other than a blank space.

Access DB2 Security Administration (Option 5.3.1.3)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Help
TE31300                    METHODMANAGER                ToolSet SERVICES

 ===>

              Access         DB2 Security Administration

 member name                  ----> ++++++++++++++++++++
 SQL statement: GRANT         ----> +++++----------------
                REVOKE        ----> -----+++++----------
                CREATE SYNONYM ----> ----------+++++-----
                DROP SYNONYM  ----> ----------------+++++
 expand nested data? (Y/N)    ----> **********----------
 SQL ID of object             ----> ********************
 user exit routine name       ----> ********************
 output to: private user      ----> -+----+----+----+---
            public user       ----> --+----+----+----+--
            PDS member        ----> ---+----+----+----+-
            sequential file   ----> ----+----+----+----+
 new/append/replace user      ----> -**---**---**---**--
 new/replace PDS member       ----> ---*----*----*----*-
 suppress print               ----> ********************

+  MANDATORY INPUT,   * OPTIONAL INPUT,  - NO INPUT ALLOWED

 F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                V2R300
```

This function is used to produce statements important for the security administration of a DB2 Database Management System.

member name

Enter the name of a member from which the required SQL statement is to be generated. The selected member must be one of the following member types:

- DB2-PRIVILEGE (for SQL GRANT and REVOKE statements)
- DB2-USER (for SQL CREATE SYNONYM and SQL DROP SYNONYM statements)

SQL statement: GRANT/REVOKE/CREATE SYNONYM/DROP SYNONYM

To generate SQL GRANT, REVOKE, CREATE SYNONYM or DROP SYNONYM statements from the definition of an appropriate repository member, enter any character other than a blank space in one of these input fields.

After generating SQL REVOKE statements, you should consider whether or not to remove the member from the repository, as you may need to grant the privilege again.

When generating SQL CREATE SYNONYM (or SQL DROP SYNONYM) statements, a separate statement is generated for every DB2-TABLE or DB2-VIEW member specified in the SYNONYM clause of the DB2-USER member.

expand nested data

If you previously expanded nested data structures, enter `y`. If you suppressed expansion of nested data structures, enter `n`.

SQL ID of object

To generate an SQL statement for an object with a specific owner, enter the authorization ID of a specified user.

user exit routine name
To take a user exit, enter the name of your user exit routine.

output to: private user/public user/PDS member/sequential file
To send your generated output to a specific destination, enter the destination in one of these input fields.

To send output to a private or public USER-MEMBER on the MP-AID, enter the USER-MEMBER name in the *private user* or *public user field*.

To send output directly to an (existing) partitioned dataset (PDS) enter the name of a PDS member in the *PDS member* field.

To send output directly to a sequential file, enter any character other than a blank space in the *sequential file* field.

new/append/replace
To file the generated output in a new USER-MEMBER, enter `n` or `new`.

To append the generated output in an existing USER-MEMBER, enter `a` or `append`.

To replace the contents of an existing USER-MEMBER, enter `r` or `replace`.

new/replace PDS member
To add the generated output to a new PDS member, enter `n` or `new`.

To replace the contents of an existing PDS member, enter `r` or `replace`.

suppress print
To print out only messages and impact analysis reports, enter any character other than a blank space.

DB2 Program Host Language Data Structures (Option 5.3.1.4)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Help
TE31400                    METHODMANAGER                ToolSet SERVICES

 ===>

              DB2 Program    Host Language Data Structures

 member name                ----> ++++++++++
 data structure/layout      ----> +++++-----
 SQL DECLARE statement      ----> *****+++++
 host language/table-layout ----> ++++++++++
 expand nested data? (Y/N)  ----> **********
 SQL ID of object           ----> **********
 location of object         ----> **********
 for sql/working-storage    ----> *****-----
 add clause to output       ----> *****-----
 user exit routine name     ----> **********
 output to: private user    ----> -+----+---
            public user     ----> --+----+--
            PDS member      ----> ---+----+-
            sequential file ----> ----+----+
 new/append/replace user    ----> -++---++--

 F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                    V2R300
```

Use this function to generate:

- Assembler, COBOL or PL1 host language data structures, to use (with embedded SQL statements) within application programs
- Reports describing the layout of repository members documenting DB2 tables or views
- SQL DECLARE TABLE statements.

member name

Enter the name of an encoded member of one of the following types:

- DB2-TABLE (for all options)
- DB2-VIEW (for all options)

data structure/layout

To generate a host language data structure or table layout, enter any character other than a blank space.

SQL DECLARE statement

To generate an SQL DECLARE TABLE statement in COBOL, PL/I, or Assembler, enter any character other than a blank space.

host language/table-layout

To generate a COBOL, PL/I, or Assembler host language data structure, enter the language name. To display a table layout, enter:

```
TABLE-LAYOUT
```

expand nested data
If you previously expanded nested data structures, enter `y`. If you suppressed expansion of nested data structures, enter `n`.

SQL ID of object
To generate an SQL statement for an object with a specific owner, enter the authorization ID of a specified user.

location of object
To generate an SQL statement for an object at a specific location, enter the location name.

for SQL/working-storage
To generate flat (two-level) data structures, enter `SQL`. To generate nested data structures, enter either `WS` or `working-storage`.

add clause to output
To add the contents of any text clause in the specified member to the REMARKS column of the table layout, enter the name of the clause.

user exit routine name
To take a user exit, enter the name of your user exit routine.

output to: private user/public user/PDS member/sequential file
To send your generated output to a specific destination, enter the destination in one of these input fields.

To send output to a private or public USER-MEMBER on the MP-AID, enter the USER-MEMBER name in the *private user* or *public user* field.

To send output directly to an (existing) partitioned dataset (PDS), enter the name of a PDS member in the *PDS member* field.

To send output directly to a sequential file, enter any character other than a blank space in the *sequential file* field.

new/append/replace user
To file the generated output in a new USER-MEMBER, enter `n` or `new`.

To append the generated output in an existing USER-MEMBER, enter `a` or `append`.

To replace the contents of an existing USER-MEMBER, enter `r` or `replace`.

new/replace PDS member
To add the generated output to a new PDS member, enter `n` or `new`.

To replace the contents of an existing PDS member, enter `r` or `replace`.

IDMS IDMS Directory (Option 5.3.2)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TE32000                        METHODMANAGER                  ToolSet SERVIC

===> |

               IDMS           IDMS Directory

process existing KEPT-DATA list ----> *
member name(s)                  ----> *
output file name                ----> *
IDMS object type                ----> *

 Use this panel to transfer information from METHODMANAGER
 format into IDMS format.


 ------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ------------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

This function provides rules, which control the export of information from MethodManager format to IDMS.

process existing KEPT-DATA list

To process only those member held in an existing KEPT-DATA list, enter the name of that KEPT-DATA list.

member name(s)

To process only particular members, enter the names of those members in this field. If you enter more than one member name, each must be separated by a comma (,).

output file name

To specify the name for the transferred file, enter a name in this field. The transferred file is used as input for the directory.

IDMS object type

To specify the type of IDMS object you want to transfer your MethodManager formatted member into enter one of the following:

- DB (Database)
- AR (Area)
- SE (Set)
- RE (Record)
- SU (Subschema)
- VI (View).

PREDICT ADABAS Directory (Option 5.3.3)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TE33000                        METHODMANAGER                  ToolSet SERVIC

===>

               PREDICT        ADABAS Directory

process existing KEPT-DATA list ----> *
member name(s)                  ----> *
output file name                ----> *
ADABAS object type              ----> *

Use this panel to transfer information from the METHODMANAGER
format into ADABAS format.


 ------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ------------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

This function provides rules, which control the export of information from MethodManager format to ADABAS.

process existing KEPT-DATA list

To process only those member held in an existing KEPT-DATA list, enter the name of that KEPT-DATA list.

member name(s)

To process only particular members, enter the names of those members in this field. If you enter more than one member name, each must be separated by a comma (,).

output file name

To specify the name for the transferred file, enter a name in this field. The transferred file is used as input for the directory.

ADABAS object type

To specify the type of ADABAS object you want to transfer your MethodManager-formatted member into, enter one of the following:

- DB (Database)
- FI (File)
- VI (View)
- DD (DDM).

SQL/DS-Objects SQL/DS Database Administration (Option 5.3.4.1)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TE34100                        METHODMANAGER                  ToolSet SERVIC

===> |

              SQL/DS-Objects   SQL/DS Database Administration

member name                ----> ++++++
ACQUIRE                    ----> +-----
CREATE                     ----> -+----
DROP                       ----> --+---
COMMENT                    ----> ---+--
LABEL                      ----> ----+-
SIZE                       ----> -----+
no impact analysis report  ----> --*---
private user name          ----> *****-
public user name           ----> *****-
new/append/replace         ----> *****-
print/noprint              ----> *****-
 ------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ------------------------------------------------------------


F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

This function is used to produce statements important for the administration of an SQL/DS Database Management System.

member name

Enter the name of a member from which the required statement is to be generated, belonging to one of the following types:

- SQL-TABLE (for all commands except ACQUIRE)
- SQL-VIEW (for all commands except ACQUIRE and SIZE)
- SQL-DBSPACE (for all commands ACQUIRE and DROP)
- SQL-INDEX (for all commands CREATE and DROP)

ACQUIRE

To generate an SQL ACQUIRE DBSPACE statement from the definition of an SQL-DBSPACE member, enter any character other than a blank space.

CREATE

To generate an SQL CREATE statement for an SQL/DS object from its definition in a repository member, enter any character other than a blank space. The SQL/DS data type of a column in a table is derived from the definition of the ITEMs and GROUPs specified in the COLUMNS clause of the SQL-TABLE member.

DROP

To generate an SQL DROP statement for a SQL/DS object from its definition in a repository member, enter any character other than a blank space. An impact analysis report is generated displaying the impact the SQL DROP statement will have in your SQL/DS environment.

COMMENT
To generate an SQL COMMENT ON statement from the definition of a SQL-TABLE member or SQL-VIEW member, enter any character other than a blank space. SQL COMMENT ON statements can be generated for a table, and for columns within a table. An SQL COMMENT ON statement is generated from every SQL-COMMENT clause in the definition of the specified SQL-TABLE or SQL-VIEW member. If the member does not have an SQL-COMMENT clause then no statements are generated.

LABEL
To generate an SQL LABEL ON statement from the definition of an SQL-TABLE or SQL-VIEW member, enter any character other than a blank space. SQL LABEL ON statements can be generated for a view, and for columns within a view. An SQL LABEL ON statement is generated from every SQL-LABEL clause in the definition of the specified SQL-TABLE or SQL-VIEW member. If the member does not have a SQL-LABEL clause then no statements are generated.

SIZE
To calculate the total size of a table and the maximum size of each row contained in that table from the definition of an SQL-TABLE member, enter any character other than a blank space. The size of a table is estimated by calculating the maximum size of each row in the table and the number of rows the table contains. The number of rows in the table is taken from the value specified in the CARDINALITY clause of the SQL-TABLE member. If the SQL-TABLE member has no CARDINALITY clause then only the maximum row size is calculated. The size of a row is determined by the SQL/DS data type of the columns in the table. These are in turn derived from the definition of the ITEMs and GROUPs specified in the COLUMNS clause of the SQL-TABLE member. The output from the SQL SIZE command cannot be automatically filed in a USER-MEMBER on the MP-AID.

no impact analysis report
To prevent an impact analysis report being generated by the SQL DROP command, enter any character other than a blank space. Furthermore an impact analysis report is not generated when a SQL DROP command is applied to an SQL-INDEX member.

private user name
Enter the name of a private USER member on the MP-AID from which generated data can be transferred into an external file which can be used as input to your DB2 environment.

public user name
Enter the name of a public USER member on the MP-AID from which generated data can be transferred into an external file which can be used as input to your DB2 environment.

new/append/replace
To file the generated output in a new USER member, enter `N`. To append the generated output in an existing USER member, enter `A`. To replace the contents of an existing USER member, enter `R`.

print/noprint
To print out all generated SQL statements, enter `P`. To print out only messages and impact analysis reports, enter `N`.

ALTER SQL Alter Command (Option 5.3.4.2)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TE34200                        METHODMANAGER                 ToolSet SERVIC

===>

              ALTER          SQL ALTER Command

member name         ----> ++++
ADD                 ----> +---
DROP                ----> -+--
ACTIVATE            ----> --+-
DEACTIVE            ----> ---+
columns             ----> *---
primary key         ----> ****
all                 ----> --**
constraint          ----> ****
private user name   ----> ****
public user name    ----> ****
new/append/replace ----> ****
print/noprint       ----> ****
 ------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ------------------------------------------------------------

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                    V2R
```

This function is used to generate SQL ALTER TABLE statements from the definition of an SQL-TABLE member, which if applied to your SQL/DS environment makes the following alterations to a table:

- Add one or more columns
- Add or drop a referential constraint
- Add or drop a primary key
- Add, modify or drop a validation routine
- Add or modify the auditing option
- A combination of the above.

The different SQL ALTER TABLE statements are generated from particular clauses in the definition of the SQL-TABLE member specified in the SQL ALTER command.

member name

Enter the name of an encoded SQL-TABLE member.

ADD

To generate an SQL ALTER TABLE statement from the definition of an SQL-TABLE member, which when applied to your SQL/DS environment adds certain objects (to be selected further down in this panel) to a table, enter any character other than a blank space.

DROP

To generate an SQL ALTER TABLE statement from the definition of a SQL-TABLE member, which when applied to your SQL/DS environment drops certain objects (to be selected further down in this panel) from a table, enter any character other than a blank space.

ACTIVATE
To generate an SQL ALTER TABLE statement which when applied to your SQL/DS environment activates certain objects (to be selected further down in this panel) on a table, enter any character other than a blank space.

DEACTIVATE
To generate an SQL ALTER TABLE statement which when applied to your SQL/DS environment deactivates certain objects (to be selected further down in this panel) on a table, enter any character other than a blank space.

columns
To identify the columns to be added to the table, enter an integer in the range 1 to 299. The columns to be added are the last 'n' columns derived from the COLUMNS clause of the SQL-TABLE member from which the SQL ALTER TABLE statement is being generated. 'n' must be less than the total number of columns derived from the COLUMNS clause.

primary key
To add or drop/activate or deactivate (depending on your previous selection) a primary key on a table, enter any character other than a blank space. Columns which allow a null value cannot be part of the primary key. The SQL statement will be rejected when submitted to SQL/DS if you attempt to add a primary key to a table that has one. The existing primary key will not be modified. If you wish to replace it you should first use the SQL ALTER command to drop the existing primary key and then to add the new one.

all
To activate or deactivate (depending on your previous selection) both the primary and foreign keys on a parent table and all the foreign keys on its dependent tables, enter any character other than a blank space. Foreign keys are defined in the CONSTRAINT clause of the SQL-TABLE member.

constraint
Enter a name or an integer to add or drop/activate or deactivate (depending on your previous selection) a referential constraint on a table. The constraint name is specified in the NAMED clause of the SQL-TABLE member which identifies the referential constraint. The integer identifies which referential constraint is to be added or dropped/activated or deactivated by its sequence among other referential constraints in the SQL-TABLE member definition.

private user name
Enter the name of a private USER member on the MP-AID from which generated data can be transferred into an external file which can be used as input to your DB2 environment.

public user name
Enter the name of a public USER member on the MP-AID from which generated data can be transferred into an external file which can be used as input to your DB2 environment.

new/append/replace
To file the generated output in a new USER member, enter `N`. To append the generated output in an existing USER member, enter `A`. To replace the contents of an existing USER member, enter `R`.

print/noprint
To print out all generated SQL statements, enter `P`. To print out only messages and impact analysis reports, enter `N`.

Access SQL/DS Security Administration (Option 5.3.4.3)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TE34300                        METHODMANAGER                  ToolSet SERVIC

===> |

              Access         SQL/DS Security Administration

member name         ----> ++++
GRANT               ----> +---
REVOKE              ----> -+--
SYNONYM             ----> --++
CREATE              ----> --+-
DROP                ----> -*-+
private user name   ----> ****
public user name    ----> ****
new/append/replace  ----> ****
print/noprint       ----> ****
 ------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ------------------------------------------------------------
Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                V2R
```

This function is used to produce statements important for the security administration of an SQL/DS Database Management System. For example, the GRANT and REVOKE commands generate SQL statements specifying the grantors and recipients of privileges, whereas SYNONYM is used to generate statements which inform you whether synonyms for tables and views for a particular user have been created or dropped.

member name

Enter the name of a member from which the required statement is to be generated. The selected member must have one of the following member types:

- SQL-PRIVILEGE (for GRANT and REVOKE),
- SQL-USER (for SYNONYM).

GRANT

To generate SQL GRANT statements from the definition of an SQL-PRIVILEGE member, enter any character other than a blank space.

REVOKE

To generate SQL REVOKE statements from the definition of a SQL-PRIVILEGE member, enter any character other than a blank space.

SYNONYM

To generate SQL CREATE SYNONYM or DROP SYNONYM statements from the definition of an SQL-USER member, enter any character other than a blank space. These statements can be generated for a synonym on a table or a view. An SQL CREATE SYNONYM or DROP SYNONYM statement is generated for every synonym specified on an SQL-TABLE or SQL-VIEW member in the SYNONYM clause of the SQL-USER member.

CREATE
To generate an SQL CREATE SYNONYM statement, enter any character other than a blank space. In this panel CREATE can only be used in combination with SYNONYM.

DROP
To generate an SQL DROP SYNONYM statement, enter any character other than a blank space. In this panel DROP can only be used in combination with SYNONYM.

private user name
Enter the name of a private USER member on the MP-AID from which generated data can be transferred into an external file which can be used as input to your SQL/DS environment.

public user name
Enter the name of a public USER member on the MP-AID from which generated data can be transferred into an external file which can be used as input to your SQL/DS environment.

new/append/replace
To file the generated output in a new USER member, enter `N`. To append the generated output in an existing USER member, enter `A`. To replace the contents of an existing USER member, enter `R`.

print/noprint
To print out all generated SQL statements, enter `P`. To print out only messages and impact analysis reports, enter `N`.

SQL/DS Program Host Language Data Structures (Option 5.3.4.4)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TE34400                        METHODMANAGER                 ToolSet SERVIC

===>

               SQL/DS Program   Host Language Data Structures

member name         ----> +++
Assembler           ----> +--
COBOL               ----> -+-
PL/I                ----> --+
private user name   ----> ***
public user name    ----> ***
new/append/replace  ----> ***
print/noprint       ----> ***

 ----------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ----------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

This function is used to generate a COBOL, PL/I, or Assembler host language data structure from the definition of an encoded member in the repository.

member name
To generate a statement or data structure, enter the name of an encoded SQL-TABLE or SQL-VIEW member.

Assembler
To generate a host language data structure in Assembler, enter any character other than a blank space.

COBOL
To generate a host language data structure in COBOL, enter any character other than a blank space.

PL/I
To generate a host language data structure in PL/I, enter any character other than a blank space.

private user name
Enter the name of a private USER member on the MP-AID from which generated data can be transferred into an external file which can be used as input to your SQL/DS environment.

public user name
Enter the name of a public USER member on the MP-AID from which generated data can be transferred into an external file which can be used as input to your SQL/DS environment.

new/append/replace
To file the generated output in a new USER member, enter `N`. To append the generated output in an existing USER member, enter `A`. To replace the contents of an existing USER member, enter `R`.

print/noprint
To print out all generated SQL statements, enter `P`. To print out only messages and impact analysis reports, enter `N`.

Other Directories of Other Systems (Option 5.3.5)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TE35000                        METHODMANAGER                  ToolSet SERVIC

===> |

              Other          Directories of Other Systems


process existing KEPT-DATA list     ----> *
member name(s)                      ----> *
external file name                  ----> *
translation rule                    ----> *

 Use this panel to transfer information from the
 METHODMANAGER format to the format required by
 another system.
 ------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ------------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

Use this function to transfer information from the MethodManager format to the format required by another system.

process existing KEPT-DATA list

To process only those member held in an existing KEPT-DATA list, enter the name of that KEPT-DATA list.

member name(s)

To process only particular members, enter the names of those members in this field. If you enter more than one member name, each must be separated by a comma (,).

external file name

To transfer data to an external file, enter the name of that file in this field. This file can be used as input to other directories.

translation rule

To specify the translation rules you want to apply to the Manager Products repository members you want to transfer to another system, enter the name of either an Executive-Routine or a TRANSLATION-RULE member.

Enter `1` in the Command Area of this panel for details of creating Executive-Routines for export purposes.

Diagram Generation of managerView diagram-members (Option 5.5)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TE50000                        METHODMANAGER                  ToolSet SERVIC

===>

              Diagram        Generation of managerVIEW diagram-members

Specify one of the following:
  Seed member name         --> +--
  Un-named KEPT-DATA list  --> -+-   (Enter Y)
  KEPT-DATA list name      --> --+

Diagram type mnemonic      --> +++
Number of tiers            --> ***
User-defined parameter 1   --> ***
User-defined parameter 2   --> ***
User-defined parameter 3   --> ***
User-defined parameter 4   --> ***
User-defined parameter 5   --> ***
User-defined parameter 6   --> ***

 Use this panel to generate a managerVIEW diagram-member. After the
 generation it can be downloaded to your local dictionary

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

name of the seed-member

To specify the seed or starting point for the diagram-member you wish to generate, enter the name of an encoded repository member.

process existing KEPT-DATA list

To interrogate only those members held in an existing KEPT-DATA list, enter the name of that KEPT-DATA list. Typically the members in the KEPT-DATA list are those members which will appear as objects on the diagram.

code for scope and type

To identify the scope and type of diagram-members to be generated, enter one of the following codes:

- BAS for Repository-Basic
- DLL for IMS (DL/I) Logical Database
- DLP for IMS (DL/I) Physical Database
- D2P for DB2 Physical
- D2R for DB2 Referential Structure
- EDM for ER Data Model
- ERD for Entity Relationship Diagram
- FDC for Functional Decomposition
- GDC for Goal Decomposition
- IMP for Impact Analysis
- ODC for Organizational Decomposition
- RDC for Requirements Decomposition
- SDC for System Decomposition
- XDM for ER Extended Data Model

Each code represents a type of diagram and all diagram-types are in the default local dictionary system supplied by ASG.

Your Systems Administrator has available an additional set of capabilities to tailor and augment the standard capabilities. Check with your Systems Administrator to determine whether additional codes are available.

number of tiers

To indicate the number of tiers of decomposition required in the diagram-member to be generated, enter an integer. For example, 2 indicates a two-tiered diagram-member, that is, the seed-member and all its direct relationships will form the diagram-member.

user-defined parameter

This allows users to specify extra syntax on the command for their own use in tailoring RDG for such things as naming diagram members or changing the way the diagram is constructed for particular instances.

For further details refer to *ASG-MethodManager User's Guide Volume 2*.

Export ADW/IEW Export (Option 5.6.2)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
E60000                        METHODMANAGER                    Export

===> |

                    Export         ADW/IEW Export

select one of the following


       Export to ADW/IEW Workstation Export Files

           1  Display        Show Generation Options
           2  Export         Perform Export function
           3  Name           Name-Related Interrogation
           4  Value          Catalog, Alias, or Attribute Interrogation









F1=Help F3=Return F4=Exit F12=Recall                                  V2R
```

Use this function to populate ADW or IEW with objects generated from exported repository members.

Name of source KEPT-DATA list

The exported members must be held in a KEPT-DATA list. The ASG-supplied default is for the KEPT-DATA list to be named EXPORT. The systems administrator can define an alternative default name. To override the ASG-supplied or user defined default specify the name of another KEPT- DATA list.

Lock exported members?

Specify `Y` to lock, and `N` to not lock, the exported members. The ASG default is to not lock the members. Locking a member prevents other users updating it for a predefined period of time. Specifying that exported members are to be locked will also RESERVE the entire repository for updates until the export is complete.

Include print output?

Specify `Y` to print, or `N` to not print, the generated data transfer file records. The ASG default is to print them.

Generate export messages?

Specify `Y` to display, or `N` to not display, DM05900I messages generated during export. The ASG default is to display them.

Write export data to files?

Specify `Y` to generate, or `N` to not generate, external files on your mainframe containing the data transfer file record. The ASG default is to create files with the following logical file names:

- IEWXOI (the OI.EXP object instance file)
- IEWXAI (the AI.EXP associations file)
- IEWXTI (the TI.EXP long properties file)
- IEWXPI (the PI.EXP short properties file).

Use aliases for object names?

Specify `Y` to generate object names from the ALIAS clauses of the exported members, or `N` to generate object names from the member's names. The ASG default is to generate names from the IEW alias type.

Process CREATED Timestamp?

Specify `Y` to export, or `N` to not export, the information held in the CREATION-DATE clauses of the exported members. The ASG default is to export the information.

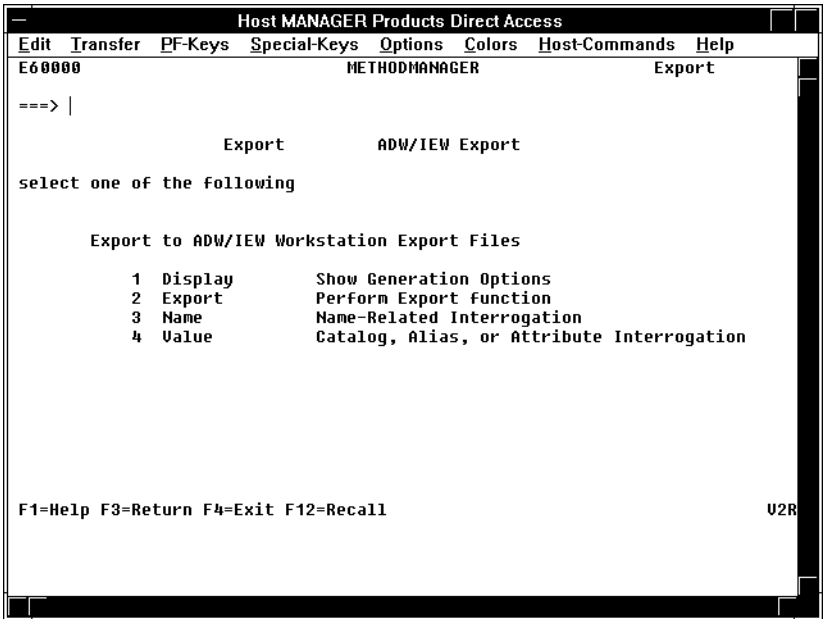
Process UPDATED Timestamp?

Specify `Y` to export, or `N` to not export, the information held in the LAST-UPDATE clauses of the exported members. The ASG default is to export the information.

Process LAST-SAVED Timestamp?

Specify `Y` to export, or `N` to not export, the information held in the LAST-SAVE-DATE clauses of the exported members. The ASG default is to export the information.

COBOL Generate repository members from COBOL (Option 4.1.1)



This function is used to generate repository definitions from COBOL source data descriptions and insert them into the repository and to update the index dataset accordingly.

source-name

Enter the name of a source member or a source copybook from which COBOL source is to be read.

file-name

To specify the dataset from which COBOL source is to be read, enter a (logical) file name.

from line-number

To specify the first line of the COBOL source record that you want to be read, enter an integer of up to eight digits.

to line-number

To specify the last line of the COBOL source record that you want to be read, enter an integer of up to eight digits.

overwrite old record

Enter one of the following:

- `GR`
- `IT`
- `IM`

If you enter `GR` and a repository definition is generated for a GROUP member type with the same name as an existing GROUP member, the new definition overwrites the old one.

If the newly generated member has the same name as an existing non-GROUP member, the new member is rejected and the existing one remains unchanged in the repository.

If no members exist in the repository under the newly generated name, the new GROUP source record is inserted into the repository.

If you enter `IT` a repository definition is generated for an ITEM member type. The same insertion rules apply for an ITEM member type as detailed above for a GROUP member type.

If you enter `IM` and a new ITEM member type is generated with the same name as an ITEM already existing in the repository, an attempt is made to merge the two data definitions.

If the newly generated member has the same name as an existing non-GROUP member, the new member is rejected and the existing one remains unchanged in the repository.

If no members exist in the repository under the newly generated name, the new GROUP source record is inserted into the repository.

form and version number

To specify the form and version number of the data description to be included in the generate definition, enter any of the following:

```
ENTERED-AS VERSION @     or     EN V @
HELD-AS VERSION @        or     H V @>
REPORT-AS VERSION @      or     REPO V @
DEFAULTED-AS             or     DE
```

@ represents the version number which has to be an integer in the range 1 to 15. If @ is omitted version 1 is defaulted.

If you do not complete this field, DEFAULTED-AS is the default.

select string from/to

To select a particular string, enter two integers and a string in this field as follows:

m (*p*) *string*

The two integers: *m* and (*p*) mark the area in the COBOL data name which is to be read. *m* specifies the character position in a data name, and *p* specifies the number of characters following *m*, which have to be read.

If *m* is omitted, a value of 1 is defaulted. If you enter *m* and *p* they must be separated by at least one blank space, and *p* has to be enclosed in brackets ( ). The sum of *m* and *p* must not exceed 97.

The string specifies the string that is to be compared with the COBOL data names in the source dataset.

For example, if you enter:

```
1 (2) MI
```

to execute the comparison between the string 'MI' and the first two characters of all member names, then all member names starting with 'MI' (e.g., MILE, MISC, MICRO) are selected for generation as a result.

replace string from/to

To specify the part of the COBOL data name to be replaced, enter two integers and a string in this field as follows:

m (*p*) *string*

The two integers: *m* and (*p*) mark the area in the COBOL data name which is to be read. *m* specifies the character position in a data name, and p specifies the number of characters following m, which have to be read.

If *m* is omitted, a value of 1 is defaulted. If you enter *m* and *p* they must be separated by at least one blank space, and *p* has to be enclosed in brackets ( ). The sum of *m* and *p* must not exceed 97.

with string

To replace the string specified under the replace string from/to field, enter a new string in this field. If you do not complete this field, the part of the COBOL data name specified in the replace string from/to field gets deleted.

delete string from/to

To specify the part of the COBOL data name to be deleted, enter two integers and a string in this field as follows:

```
m (p) string
```

The two integers: *m* and (*p*) mark the area in the COBOL data name which is to be deleted. *m* specifies the character position in the data name, and *p* specifies the number of characters following *m*, which you want to delete.

If *m* is omitted, a value of 1 is defaulted. If you enter *m* and *p* they must be separated by at least one blank space, and *p* has to be enclosed in brackets ( ). The sum of *m* and *p* must not exceed 97.

insert string

Enter a string which is to be inserted into source data names in the way described in the in front of/behind operand string field.

in front of/behind operand string

The keywords explained below allow the selection of source data names and the action which is to be executed on them. Enter:

```
P nnn OP
```

where:

P determines whether the new string is to be inserted before or after the operand string, and must be either `B` (for before) or `A` (for after).

nnn specifies the number of occurrences of the operand string that you want to adjust, and must be either a specific integer or AL if all occurrences the new string is to be added to all operand strings.

OP must be one of the following:

```
WHEN LINE    =   line-number
WHEN ANY     =   string
WHEN (p)     =   string
WHEN m (p)   =   string
```

where:

line-number is a five digit integer.

(*p*) is an integer in the range 1 to 97 enclosed in brackets.

m is an integer greater than 0.

string is the operand string.

The sum of both integers *m* and (*p*) must not exceed 97, because this is the maximum number of characters of a data name.

PL/I Generate Repository Members from PL/I (Option 4.1.2)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TI12000                          METHODMANAGER                    ToolSet SERVIC

===>

              PL/I           Generate repository members from PL/I

source-name                        ---> ++++++
file-name                          ---> ******
from line-number                   ---> ++----
to line-number                     ---> -+----
overwrite old record               ---> ******
form and version number            ---> --+---
select string from/to              ---> ------
replace string from/to             ---> ++-+--
with string                        ---> ++-+--
delete string from/to              ---> ----+-
insert string                      ---> -----+
in front of/behind operand string ---> -----+

 Use this panel to generate member definitions in a METHODMANAGER
 repository from data descriptions currently written in PLI.
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

This function is used to generate repository definitions from PL/I source data descriptions and insert them into the repository and to update the index dataset accordingly.

source-name
Enter the name of a source member or a source copybook from which PL/I source is to be read.

file-name
To specify the dataset from which PL/I source is to be read, enter a (logical) file name.

from line-number
To specify the first line of the PL/I source record that you want to be read, enter an integer of up to eight digits.

to line-number
To specify the last line of the PL/I source record that you want to be read, enter an integer of up to eight digits.

overwrite old record
Enter one of the following:

- DG
- DE
- DM.

If you enter DG and a repository definition is generated for a GROUP member type with the same name as an existing GROUP member, the new definition overwrites the old one.

If the newly generated member has the same name as an existing non-GROUP member, the new member is rejected and the existing one remains unchanged in the repository.

If no members exist in the repository under the newly generated name, the new GROUP source record is inserted into the repository.

If you enter DE a repository definition is generated for an ITEM member type. The same insertion rules apply for an ITEM member type as detailed above for a GROUP member type.

If you enter DM and a new ITEM member type is generated with the same name as an ITEM already existing in the repository, an attempt is made to merge the two data definitions.

If the newly generated member has the same name as an existing non-GROUP member, the new member is rejected and the existing one remains unchanged in the repository.

If no members exist in the repository under the newly generated name, the new GROUP source record is inserted into the repository.

form and version number

To specify the form and version number of the data description to be included in the generated definition, enter any of the following:

```
ENTERED-AS VERSION @     or       EN V @
HELD-AS VERSION @        or       H V @>
REPORT-AS VERSION @      or       REPO V @
DEFAULTED-AS             or       DE
```

@ represents the version number which has to be an integer in the range 1 to 15. If @ is omitted version 1 is defaulted.

If you do not complete this field, DEFAULTED-AS is the default. Enter `1` for further details of form descriptions.

select string from/to

To select a particular string, enter two integers and a string in this field as follows:

m (*p*) *string*

The two integers: *m* and (*p*) mark the area in the COBOL data name which is to be read. *m* specifies the character position in a data name, and *p* specifies the number of characters following in, which have to be read.

If *m* is omitted, a value of 1 is defaulted. If you enter *m* and *p* they must be separated by at least one blank space, and *p* has to be enclosed in brackets ( ). The sum of *m* and *p* must not exceed 97.

The string specifies the string that is to be compared with the PL/I data names in the source dataset.

For example, if you enter:

```
1 (2) MI
```

to execute the comparison between the string 'MI' and the first two characters of all member names, then all member names starting with 'MI' (e.g., MILE, MISC, MICRO) are selected for generation as a result.

replace string from/to

To specify the part of the PL/I data name to be replaced, enter two integers and a string in this field as follows:

```
m (p) string
```

The two integers: *m* and (*p*) mark the area in the PL/I data name which is to be read. *m* specifies the character position in a data name, and *p* specifies the number of characters following *m*, which have to be read.

If *m* is omitted, a value of 1 is defaulted. If you enter *m* and *p* they must be separated by at least one blank space, and *p* has to be enclosed in brackets ( ). The sum of *m* and *p* must not exceed 97.

with string

To replace the string specified under the replace string from/to field, enter a new string in this field. If you do not complete this field, the part of the PL/I data name specified in the replace string from/to field gets deleted.

delete string from/to

To specify the part of the PL/I data name to be deleted, enter two integers and a string in this field as follows:

```
m (p) string
```

The two integers: *m* and (*p*) mark the area in the PL/I data name which is to be deleted. *m* specifies the character position in the data name, and *p* specifies the number of characters following *m*, which you want to delete.

If *m* is omitted, a value of 1 is defaulted. If you enter *m* and *p* they must be separated by at least one blank space, and *p* has to be enclosed in brackets ( ). The sum of *m* and *p* must not exceed 97.

insert string

Enter a string which is to be inserted into source data names in the way described in the in front of/behind operand string field.

in front of/behind operand string

The keywords explained below allow the selection of source data names and the action which is to be executed on them. Enter:

```
P nnn OP
```

where:

P defines whether the new string is to be inserted before or after the operand string, and must be either B (for before) or A (for after).

nnn specifies the number of occurrences of the operand string that you want to adjust, and must be either a specific integer or AL if all occurrences the new string is to be added to all operand strings.

OP must be one of the following:

```
WHEN LINE     =   line-number
WHEN ANY      =   string
WHEN (p)      =   string
WHEN m (p)    =   string
```

where:

line-number is a five digit integer.

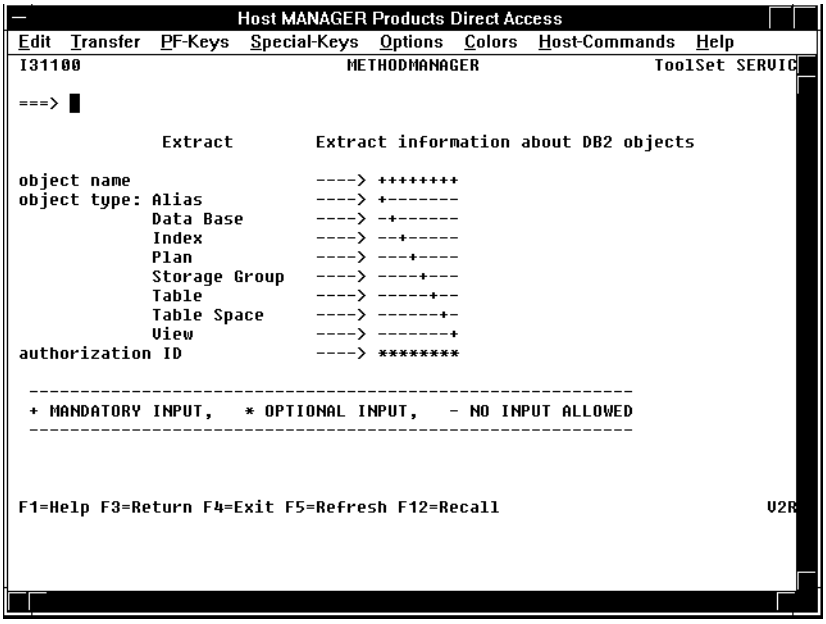
(p) is an integer in the range 1 to 97 enclosed in brackets.

m is an integer greater than 0.

string is the operand string.

The sum of both integers *m* and *(p)* must not exceed 97, because this is the maximum number of characters of a data name.

Extract Extract Information About DB2 Objects (Option 4.3.1.1)



Use this function to import information about specified DB2 objects, of specified types, from your DB2 environment.

object name selection

Enter one of the following:

- The name of a single object in your DB2 environment
- A combination of characters and * symbols matching a selection of object names, where each * symbol represents a string of zero or more characters of any type.

object type: alias/database/index/plan/storage group/table/table space/view

To import information about a DB2 alias, database, index, plan, storage group, table, table space or view, enter any character other than a blank space in one of these input fields.

owner ID selection

Enter one of the following:

- The authorization ID of the owner of the object being imported
- A combination of characters and * symbols matching a selection of authorization IDs, where each * symbol represents a string of zero or more characters of any type.

Reconcile Produce a Report of the Extracted Objects (Option 4.3.1.2 and 4.3.42)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
I31200                          METHODMANAGER                ToolSet SERVIC

===> |

              Reconcile      Produce a report of the extracted objects

initialize                  ----> -+-
translation rule name       ----> -*-
no common clauses           ----> -**
ignore:    member type      ----> -**
           all/duplicates   ----> -**
           member name      ----> -**
           member ID        ----> -**
replace:   member type      ----> -**
           all/duplicates   ----> -**
           member name      ----> -**
           member ID        ----> -**
add:       member type      ----> -**
           all/duplicates   ----> -**
           member name      ----> -**
           member ID        ----> -**
show summary/detail/both    ----> -**

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

Use this function to generate proposed members from extracted information on the WBTA, and reconcile these proposed members with the current contents of the repository.

initialize

To regenerate the proposed members according to the default translation rules, abandoning any changes made previously, enter any character other than a blank space.

translation rule name
To tailor the reconcile stage to generate proposed members which meet your repository standards, enter the name of a user-written executive routine to perform this tailoring.

no common clauses
To prevent the common clauses of existing repository members being incorporated in proposed members which are replacing them, enter any character other than a blank space.

ignore
To ignore a selection of proposed members, so that these members are not subsequently entered in the repository, enter one or more of the following:

- Member type: enter a specific member type
- All/duplicates: enter `all` (or `a`) to ignore all members, or `duplicates` (or `d`) to ignore proposed members that have the same name as an existing member
- Member name: enter the name of a specific member
- Member ID: enter the ID number of a member in the reconciliation report

in the appropriate input fields.

replace
To replace a selection of existing members on the repository, enter one or more of the following:

- Member type: enter a specific member type
- All/duplicates: enter `all` (or `a`) to replace all members, or `duplicates` (or d) to replace proposed members that have the same name as an existing member
- Member name: enter the name of a specific member
- Member ID: enter the ID number of a member in the reconciliation report

in the appropriate input fields.

add
To add a selection of proposed members to the repository, enter one or more of the following:

- Member type: enter a specific member type
- All/duplicates: enter `all` (or `a`) to add all members, or `duplicates` (or `d`) to add proposed members that have the same name as an existing member
- Member name: enter the name of a specific member
- Member ID: enter the ID number of a member in the reconciliation report

in the appropriate input fields.

show summary/detail/both
To display a summarized reconciliation report, enter `summary` (the default). To display a detailed reconciliation report, enter `detail`. To display both a summarized and a detailed reconciliation report, enter both.

Preview Generate Repository Definitions (Option 4.3.1.3 and 4.3.4.3)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
I31300                          METHODMANAGER                 ToolSet SERVIC

===>

              Preview        Generate repository definitions

layout executive name    ----> ***
output to: private user  ----> +--
           public user   ----> -+-
new/append/replace       ----> **-
suppress print           ----> **-



 ----------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ----------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

Use this function to generate a member definition statement from information obtained in the reconcile stage of the Import Procedure.

layout executive name

To tailor member definition statements to your own layout requirements, enter the name of a user-written executive routine.

output to: private user/public user

Enter the name of a private or public USER-MEMBER on the MP-AID to which generated data should be sent.

new/append/replace

To file the generated output in a new USER-MEMBER, enter `n` or `new`. To append the generated output in an existing USER-MEMBER, enter `a` or `append`. To replace the contents of an existing USER-MEMBER, enter `r` or `replace`.

suppress print

To print out only messages and impact analysis reports, enter any character other than a blank space.

Populate Insert Definitions Into the Repository (Option 4.3.1.4 and 4.3.4.4)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TI31400                        METHODMANAGER                 ToolSet SERVIC

===> |

              Populate       Insert generated definitions into a repository

from USER-MEMBER name ----> --++
roll statements back  ----> -+-+
rollback level (e/w)  ----> -*-*



 ------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ------------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

Use this function to insert member definitions into your repository. This is done by executing the member definition statements previously generated during the preview stage.

from USER-MEMBER name

To execute member definition statements contained in a USER-MEMBER on the MP-AID, enter the name of the USER-MEMBER.

roll statements back

To specify that member definition statements are to be treated as a single logical unit of work, enter any character other than a blank space. If the populate stage is not totally successful, all of the statements will be rolled back, leaving the repository unchanged.

rollback error level

To specify that statements should be rolled back if error messages occur, enter `e`. To specify that statements should be rolled back if warning messages occur, enter `w`.

Extract Extract Information about SQL/DS Objects (Option 4.3.2.1)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
I32100                        METHODMANAGER                ToolSet SERVIC

===> 

              Extract        Extract information about SQL/DS objects

object name                  ----> ++++
object type: Index           ----> +---
             Dbspace         ----> -+--
             Table           ----> --+-
             View            ----> ---+
authorization ID             ----> ****


 ----------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ----------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

Use this function to import information about specified SQL/DS objects, of specified types, from your SQL/DS environment.

object name selection

Enter one of the following:

- The name of a single object in your SQL/DS environment
- A combination of characters and * symbols matching a selection of object names, where each * symbol represents a string of zero or more characters of any type.

object type: index/dbspace/table/view

To import information about a SQL/DS index, dbspace, table or view, enter any character other than a blank space in one of these input fields.

authorization ID

Enter one of the following:

- The authorization ID of the owner of the object being imported
- A combination of characters and * symbols matching a selection of authorization IDs, where each * symbol represents a string of zero or more characters of any type.

Import ADW/IEW Import (Option 4.5.2)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TI52000              METHODMANAGER                                   Impor

===>

             Import         ADW/IEW Import

extract   ----> +----++++
reconcile ----> -+---++++
  sub-clauses ----> -*---****
preview   ----> --+---+++
  sub-clauses ----> --*---***
populate  ----> ---+---++
  sub-clauses ----> ---*---**
RESET     ----> ----+---+

 ------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ------------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                    V2R
```

Use this function to populate your repository with members generated from imported ADW or IEW objects.

extract

To extract information from the ADW or IEW export files resident on your mainframe and populate Procedures Language global variables on the WBTA, enter any character other than a blank space.

The ASG-supplied default is that the export files have the following logical file names:

- ASSOC (the AI.EXP associations file)
- OBJECT (the OI.EXP object instances file)
- PROP (the TI.EXP long properties file)
- TEXT (the PI.EXP short properties file).

A message specifying the number of objects on which information has been extracted is output. Any previously extracted information held on the WBTA is automatically replaced.

reconcile

To generate:

- Proposed members from the information placed on the WBTA by EXTRACT
- A reconciliation report that reports both the proposed members and any existing members in the current or next visible status that have the same name

enter any character other than a blank space.

Depending on whether ASG-supplied or user defined defaults are in place in your environment, RECONCILE will determine whether proposed members:

- Are subsequently entered into the repository with an ADD or a REPLACE command. The ASG-supplied default is to ADD proposed members.
- Include in their definition the following common clauses of the existing members they are to replace:
  - ADMINISTRATIVE-DATA
  - CATALOG
  - COMMENT
  - DESCRIPTION
  - NOTE.

The ASG-supplied default is to exclude the common clauses of existing members.

You can override the above defaults by specifying the following keywords in the panel:

- ADDING to specify that you want the proposed members to be entered in the repository by ADD commands
- REPLACING to specify that you want the proposed members to be entered in the repository by REPLACE commands
- COMMON-CLAUSES or NO-COMMON-CLAUSES to include or exclude the common clauses of existing members in the definition of the proposed members replacing them.

You can also use the following commands in combination with the reconciliation report:

- RADD to specify that a proposed member will be entered in the repository by an ADD command
- RREP to specify that a proposed member will be entered in the repository by a REPLACE command
- RIGN to specify that a proposed member will not be entered in the repository
- RREN to rename a proposed member
- RUPD to update an existing member with the same name as a proposed member.

For example, if you did not want a proposed member to replace an existing member you could use the RREN command to rename the proposed member or the RIGN command to specify that it is not to be entered in the repository.

Only proposed IEW-DATAFLOW members with names prefixed by IF1- (or a user defined alternative) are reconciled with existing repository members.

Objects that are not supported by Manager Products are listed on the reconciliation report but member definition are not generated for them. The unsupported objects are indicated by comments in the subsequent PREVIEW output.

preview

To generate ADD or REPLACE command and member definition statements for the proposed members generated by the preceding RECONCILE, enter any character other than a blank space.

Depending on the ASG-supplied or user defined defaults in place in your environment, PREVIEW will:

- File the generated statements in either a new or an existing public USER-MEMBER. If the USER-MEMBER already exists then the statements either replace its contents or are appended to it. The ASG-supplied default is for the statements to replace the contents of a USER-MEMBER named IEWRUN. The filed statements can be both:
  - Held across Manager Products sessions
  - Edited in the edit buffer
  - Either print or not print the statements. The ASG-supplied default is not to print statements.

You can override the above defaults by specifying the following in the panel:

- The name of the USER-MEMBER in which you want to file the generated output
- NEW to file the output in a new member
- APPEND to append the output to the contents of an existing member
- REPLACE (the default) to replace the contents of an existing member
- NOPRINT or NO-PRINT to not print the generated statements
- PRINT to print the generated statements.

If you specify NEW, and the member already exists, the output will not be generated. If you specify APPEND or REPLACE, and the member does not already exist, a new member is created.

Definitions are generated for members in the same order as they are listed on the reconciliation report.

Objects that are not supported by Manager Products and therefore cannot have a member definition generated for them are indicated by comments. These comments help you to relate the output with the previous reconciliation report. Generated IEW-TBSPACE and IEW-INDEX member definitions are also indicated by comments.

populate

To populate the repository with the definitions generated by the preceding PREVIEW, enter any character other than a blank space.

Depending on whether ASG-supplied or user defined defaults are in place in your environment, POPULATE will:

- Only execute statements filed in a public USER-MEMBER or alternatively displayed in the current buffer. The ASG-supplied default is for PREVIEW to file statements in a USER-MEMBER named IEWRUN and for POPULATE to execute them.
- Print or not print the output generated by the executed statements. The ASG-supplied default is to print output.
- Execute all the statements separately or as one Logical Unit of Work (LUW). The ASG-supplied default is for all the statements to form a LUW that will either update the repository or be rolled back in its entirety, leaving the repository unchanged, if for any reason any of the statements are unsuccessful.

You can override the above defaults by specifying the following keywords in the panel:

- USER user-member-name to execute statements filed in a specified USER-MEMBER
- BUFFER to execute statements displayed in the current buffer
- ROLLBACK to specify that all the executed statements will form one LUW
- PRINT to print any output generated in response to the executed statements
- NOPRINT or NO-PRINT to not print any output generated in response to the executed statements.

You can separately execute each of the statements filed in a USER-MEMBER by entering the member's name in the command area or by specifying it in a batch job. You may want to do this if by default all statements in your environment are executed as an LUW but space limitations on the recovery dataset are restrictive in a particular instance. An estimate would be that on average each imported object requires 6K of space on the recovery dataset, though the actual amount of space required by an individual object will vary according to its size.

RESET

To reset the WBTA, enter any character other than a blank space.

If you have completed or abandoned the import process you can reset the WBTA, thereby clearing it of any extracted information. Resetting the WBTA increases the amount of virtual storage available in your environment.

If you do not reset the WBTA then the information held upon it is replaced by a subsequent EXTRACT.

Print Output Documents (Option 3.3)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TK30000                        METHODMANAGER                 ToolSet SERVIC

===> 

              Print          Output documents

name of document       ----> +-+-
name of KEPT-DATA list ----> -+-+
produce hardcopy       ----> --++



 Generate a document, or several documents from the named KEPT-DATA list.
 If there is no entry in the produce hardcopy field, the generated
 output is displayed on the screen. If any character except a space
 is entered in this field, the output is sent to the printer.
 ------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

name of document

Enter the name of an existing DOCUMENT member that is to be displayed or output.

name of KEPT-DATA list

Enter the name of an existing KEPT-DATA list containing DOCUMENT members to be displayed or printed. This list should only contain DOCUMENT members.

produce hard copy

Enter any character except a space. This initiates the output of hard copies of the selected documents. If no character is entered the documents will be displayed on screen.

Structure Display Dependencies of Documents (Option 3.4)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TK40000                        METHODMANAGER                 ToolSet SERVIC

===> |

              Structure      Display dependencies of a document

name of document             ----> ++++++++-
sub-documents                ----> +---+----
higher documents             ----> -+---+---
subordinate members          ----> --+---+--
higher members               ----> ---+---+-
KEPT-DATA list to be created ----> ----+++++



 This panel enables you to interrogate all subordinate or higher level
 documents/members. If you enter a name in the KEPT-DATA list field, the
 results of your interrogation are saved in the named KEPT-DATA list.
 ----------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ----------------------------------------------------------

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

name of document
Enter the name of an existing DOCUMENT member you want to interrogate.

sub-documents
To display a list of all DOCUMENT members that are used by the named member, enter any character except a space.

higher documents
To display a list of all DOCUMENT members that use the named member, enter any character except a space.

subordinate members
To display a list of all members (of any type) that are used by the named member, enter any character except a space.

higher members
To display a list of all members (of any type) that use the named member, enter any character except a space

KEPT-DATA list to be created
To keep the displayed list, enter the name of a KEPT-DATA list. If you choose the name of an existing KEPT-DATA list, its contents will be overwritten.

Load Load from the Repository (Option 2.2.B.1)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TW2B100                        METHODMANAGER                 ToolSet SERVIC

===> 

              Merge          Load from Repository

without checking ITEMs or GROUPs ----> +--+
with expansion of the GROUPs     ----> ++--
name of dictionary member        ----> ++++



 ----------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ----------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                V2R
```

Use this function to load additional repository entries into the current Workbench Design Area (WBDA). Entries can be data-views (USERVIEWs and ENTITIEs) or VIEWSETs.

without checking ITEMs or GROUPs

To suppress checking of the data elements (ITEMs or GROUPs) referred to directly or indirectly in the selected repository members to see if they are verified members of the repository, enter any character except a blank space.

with expansion of the GROUPs

To expand any referenced GROUP data elements into their constituent ITEM data elements when loaded, enter any character except a blank space. This does not apply to instances of such GROUPs already present in the WBDA.

name of repository member

Enter the name of a repository member you want to be loaded into the WBDA. This must be the name of a USERVIEW, ENTITY or VIEWSET member.

Normal Form Normalize 1NF/2NF/3NF (Option 2.2.B.2)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TW2B200                        METHODMANAGER                 ToolSet SERVIC

===> |

              Normal Form    Normalize 1NF/2NF/3NF

1NF           ----> ++----
2NF           ----> --++--
3NF           ----> ----++
long report   ----> +-+-+-
short report  ----> -+-+-+




 ------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ------------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                V2R
```

Use this function to normalize a re-modeled workbench and to create reports that describe the contents of the Workbench Design Area (WBDA).

1NF
To initiate the generation of a 1NF relational or network schema from the dependency data in the WBDA, enter any character except blank. Enter `1` for further details.

2NF
To initiate the generation of a 2NF relational or network schema from the dependency data in the WBDA, enter any character except blank. Enter `2` for further details.

3NF
To initiate the generation of a 3NF relational or network schema from the dependency data in the WBDA, enter any character except blank. Enter `3` for further details.

long report
To initiate the production of a detailed audit report of any inconsistent or extraneous contents in the WBDA, enter any character except blank.

short report
To initiate the production of a brief audit report of any inconsistent or extraneous contents in the WBDA, enter any character except blank.

Load Factors Calculate and Evaluate Load Factors (Option 2.3.A)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TW3A000                        METHODMANAGER                 ToolSet SERVIC

===> |

              Load Factors   Calculate and Evaluate Load Factors

calculate        ----> +--
evaluate         ----> -++
selection        ----> -++
summary report   ----> --+
detail report    ----> -+-

 Use this panel to report all or some of the load factors that
 are present in the WBDA

 ------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ------------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

Use this function to calculate and evaluate load factors. A load factor is the physical access time of a data element of the logical database model with respect to the userviews in which the data element appears.

calculate

To calculate the load factors for all data elements that are referenced by the userviews in the current WBDA, enter any character other than a blank space.

evaluate

To output a load factor analysis report, enter any character except a blank space.

selection

To output a load factor analysis report for all data elements referenced by userviews held in the current WBDA, enter:

```
all
```

To select certain objects, either enter the name or the internal number of a data element. If you enter more than one name or number, each must be separated by a comma.

summary report

To generate a summary report, enter any character except a blank space.

detail report

To generate a detailed report, enter any character except a blank space.

Objects List Workbench Objects (Option 2.3.1)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TW31000                        METHODMANAGER                 ToolSet SERVIC

===> 

              Objects        List Workbench Objects

entities      ----> +-------
userviews     ----> -+------
relations     ----> --+-----
records       ----> ---+----
data elements ----> ----+---
tables        ----> -----++-
roots         ----> ------*-
parents       ----> ------*-
leaves        ----> ------*-
dependents    ----> ------*-
independents  ----> ------*-
cycles        ----> -------+

Use this panel to list any object types in the current workbench
 ------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

Use this function to display lists of selected types of objects from the Workbench Design Area (WBDA).

Enter any non-blank character to select the member types in which you are interested.

entities
Selects all ENTITIES.

userviews
Selects all USERVIEWs.

relations
Selects all relations that have been generated during the normalization process.

records
Selects all records that have been generated during the normalization process.

data elements
Selects all data elements that are referenced by ENTITIES or USERVIEWs in the current WBDA.

tables
Selects all DB2 and/or SQL/DS TABLE members in the current WBDA: If the DB2 or SQL/DS functional units are not installed at your site, selecting this field will have no effect (the same applies to the following fields).

roots
Selects only the root parent tables.

parents
Selects all parent tables, including tables that are root parents, and tables that are both parents and dependents.

leaves
Selects only leaf dependent tables.

dependents
Selects all dependent tables, including tables that are leaf dependents, and tables that are both dependents and parents, enter any character except blank.

independents
Selects only those tables that are neither parent nor dependent tables (that is tables which do not participate in any foreign key relationship).

cycles
Selects the cycles found in the DB2 or SQL/DS design in the WBDA.

Dependencies List Dependencies of Workbench Objects (Option 2.3.2)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TW32000                        METHODMANAGER                  ToolSet SERVIC

===> |

              Dependencies   List Dependencies of Workbench Objects

intersecting data elements      ----> +------
all dependency types            ----> -+-----
functional dependencies         ----> --+----
multivalued dependencies        ----> ---+---
implied functional dependencies ----> ----+--
domain dependencies             ----> -----+-
root records                    ----> ------+

Use this panel to produce reports on the relationships between
different object types in the current workbench.
 ------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ------------------------------------------------------------
Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

Use this function to produce reports on the different types of relationships between object types in the Workbench Design Area (WBDA).

Enter any non-blank character to select the type of relationship you want to report.

intersecting data elements

Selects all the intersecting data elements in the WBDA. An intersecting data element is one that appears on the right hand side of two or more functional dependencies (FDs) and/or domain dependencies (DDs), but is not contained in the left-hand side of any FD. This means that the data element is defined uniquely by more than one left-hand side, so the report is useful in identifying possible occurrences of homonyms or synonyms. (For an example, see the *ASG-DesignManager User's Guide*.)

all dependencies

Selects all the dependencies.

functional dependencies

Selects all the functional dependencies (FDs). A functional dependency holds from A to B if each value of A determines exactly one value of B. B is said to be functionally dependent on A. For example:

```
EMPLOYEE-NUMBER ----> EMPLOYEE-NAME
```

multivalued dependencies

Selects all the multivalued dependents (MVDs). A multivalued dependency holds from A to B if, for each value of A, a variable number of values of B may be determined. B is said to be multiply dependent upon A. For example:

```
EMPLOYEE-NAME ---->> EMPLOYEE-CHILD-NAME.
```

The double-headed arrow (> >) indicates that an employee may have more than one child.

implied functional dependencies

Selects all implied functional dependencies. If a data element that appears in a dependency described in the WBDA, is a GROUP member of the repository, and another data element in the WBDA is one of the CONTAINed members, then the system automatically generates an 'implied' FD from the GROUP data element to the CONTAINed data element, and unless it already exists there, adds the implied FD to the WBDA.

For example: if the data element, DATE is a GROUP member containing the ITEMs: YEAR, MONTH, and DAY, and if the data element, MONTH appears in any dependency in the WBDA, then, unless already present, the implied FD DATE ---> MONTH is added to the WBDA.

domain dependencies

Selects all domain dependencies (DDs). A domain dependency holds from A to B if A is a subcategory of B, that is, every value of A is also a value of B. It implies that any data element which is functionally dependent on B is also functionally dependent on A.

For example: if you have the two data elements: EMPLOYEE-NUMBER and MANAGER-NUMBER and you want to express the fact that the set of MANAGER-NUMBERs is itself a subset of the set of EMPLOYEE NUMBERs, then you can do this by defining the DD:

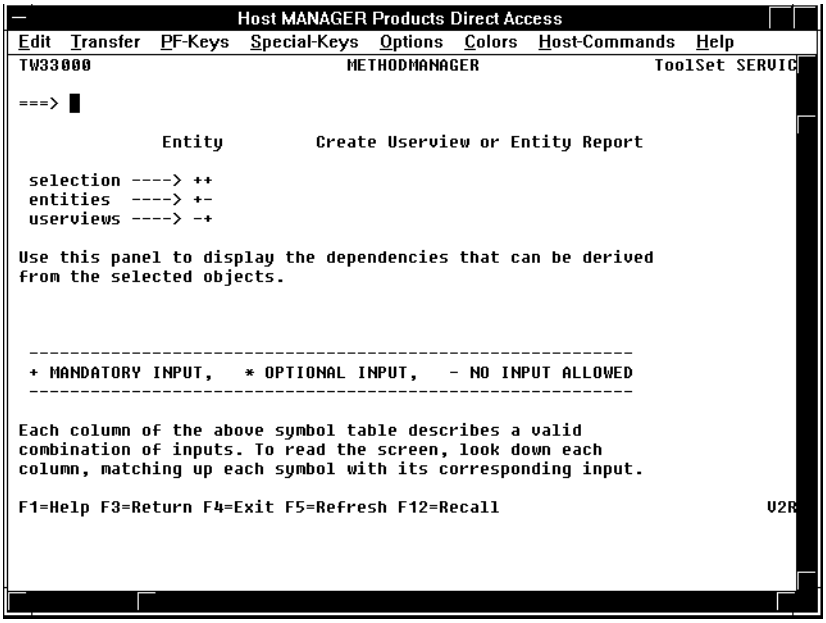
```
MANAGER-NUMBER ====> EMPLOYEE-NUMBER.
```

root records

Selects all root records. A root record is a record with no parents. That means a root record is not the target record in any multivalued association, and it does not contain the key of any other record within its key (in a hierarchical association).

The purpose of identifying root records is to help you to select good seeds for the Consolidated Network Plot, that is, seeds which permit the display of the network in a hierarchically top-down direction.

Entity Create Userview or Entity Report (Option 2.3.3)



Use this function to report all the data in the Workbench Design Area (WBDA) relevant to selected data-views (either userviews or entities).

A detailed report is produced about the number of data-views in the current workbench, the dependencies (FDs, MVDs and/or DDs) of each individual data-view plus relevant information about these dependencies (including the effects of normalization, if applicable, and the resulting relations and records that represent the dependencies).

selection

To get a report about all userview or entities, enter `all`.

To select certain objects, enter either the name or the internal number of a member. If you enter more than one name or number each must be separated by a comma.

For an overview of valid data-view names and their internal numbers in the current WBDA, use panel TW31000, selecting either *entities* or *userviews*.

All

Is the default value for this field and is applied if you make no input.

entities
To select all entities in the current workbench, enter any character other than a blank space.

userviews
To select all userviews in the current workbench, enter any character other than a blank space.

Usage Create Data Element Usage Report (Option 2.3.4)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TW34000                        METHODMANAGER                 ToolSet SERVIC

===> |

              Usage          Create Data Element Usage Report

selection ----> +

Use this panel to display all dependencies from userviews
entities and any relations or records that are referrenced
by the selected data elements.



 ------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ------------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                V2R
```

Use this function to get details of the usage of all, or selected, data elements held in the Workbench Design Area (WBDA).

A detailed report is produced about the number of data elements in the current workbench, plus details about the usage of each data element by individual data-views, by individual dependencies, by individual relations and by individual records.

selection
To report all data elements held in the WBDA, enter:

```
all
```

To select certain objects, either enter the name of the internal number of a data element. If you enter more than one name or number, each must be separated by a comma.

For an overview of valid data-elements names and their internal numbers in the current WBDA, use panel TW31000, selecting 'data elements'.

Relational Create Relational Schema Report (Option 2.3.5)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TW35000                        METHODMANAGER                ToolSet SERVIC

===>

              Relational     Create Relational Schema Report

selection      ----> +++
logical schema ----> ++-
summary        ----> -+-
tables         ----> --+
format name    ----> --*

Use this panel to display all the dependencies which have lead
to the creation of the selected relations, or give information
on the origin of the relations.
 ------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ------------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                V2R
```

Use this function to report the relations/tables held in the Workbench Design Area (WBDA).

selection

To report all relations/tables held in the WBDA, enter:

```
all
```

To select certain objects, enter either the name or the internal number of a relation. If you enter more than one name or number, each must be separated by a comma.

Note:

If you want to select individual relations by name, ensure that their names have been assigned previously. Use panel TW28000 to assign names to relations.

For an overview of valid relation names and their internal numbers in the current WBDA, use panel TW31000, selecting 'relations'.

logical schema

To report associated dependencies, pointers to other relations and generated syntax, enter any character except a blank space. The syntax is intended for use in storing the relations in the repository.

summary

To report the logical/relational schema, showing the number of each selected relation, its name, the data elements forming the primary key of the relation and its non-key elements, enter any character except a blank space.

The default is a detailed report of the logical/relational schema.

tables
To report for each DB2 or SQL/DS table in the WBDA, which depend are represented by the table and which other tables it is related to, enter character except a blank space.

If the DB2 or SQL/DS functional units are not installed at your site, selecting this field will have no effect.

format name
To generate reports in a particular layout, enter the name of an existing FORMAT member in which the required layout is pre-defined.

Network Create Network Schema Report (Option 2.3.6)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TW36000                        METHODMANAGER                  ToolSet SERVIC

===> |

              Network        Create Network Schema Report

selection    ----> ++
summary      ----> -+
format name  ----> **

Use this panel to select records that can be transferred into
a network schema, which can be used as a help to generate the
segments, records or files required, respectively, for a
hierarchical, network or inverted file type of DBMS.

 ----------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ----------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

Use this function to display a network schema. A network schema is a set of records, in a Normalized Form (1NF, 2NF or 3NF) that can be used as a help to generate the segments, records, or files required, respectively, for a hierarchical, network, or inverted file type of DBMS.

selection
To select all existing records in the current WBDA that are to be used for the creation of the network schema, enter:

`all`

(default).

To select certain objects, either enter the name or the internal number of a record. If you enter more than one name or number, each must be separated by a comma.

summary
To generate a summary network schema report, enter any character except blank. The default is the display of a detailed network schema report.

format name

To generate reports in a particular layout, enter the name of an existing FORMAT member in which the required layout is pre-defined.

Objects-M List Workbench Objects - Added by Latest Merger (Option 2.3.7)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TW37000                        METHODMANAGER                  ToolSet SERVIC

===> |

              Objects-M      List Workbench Objects -
                             Added by Latest Merger

entities                        ----> +----------
userviews                       ----> -+---------
relations                       ----> --+--------
records                         ----> ---+-------
data elements                   ----> ----+------
all dependency types            ----> -----+-----
functional dependencies         ----> ------+----
multivalued dependencies        ----> -------+---
implied functional dependencies ----> --------+--
domain dependencies             ----> ---------+-
root records                    ----> ----------+

Use this panel to display all the dependencies that can be
derived from the selected objects after the latest merger.
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

The function of this panel is fundamentally the same as that provided by panels TW31000 'List Workbench Objects' and TW32000 'List Dependencies of Workbench Objects' together. This panel, however, only lists those workbench objects that were loaded from the repository, and merged in the Workbench Design Area (WBDA) during the latest merger.

The function is especially useful for incremental design techniques, when new data from the repository is added to the contents of an already existing WBDA, because it allows you to examine the incremental data only.

Enter any non-blank character to select the workbench contents that you want to list.

entities

Selects all entities that have been loaded into the WBDA most recently.

userviews

Selects all userviews that have been loaded into the WBDA most recently.

relations

Selects all relations.

records

Selects all records.

data elements
Selects all data elements, which were added to the WBDA by the most recent loading of data-views into the WBDA, and that are referenced by entities or userviews in the current WBDA.

all dependency types
Selects all those dependencies most recently merged into the WBDA: functional dependencies (FDs), multivalued dependencies (MVDs) and domain dependencies (DDs).

functional dependencies
Selects all functional dependencies (FDs) most recently merged into the WBDA.

A functional dependency holds from A to B if each value of A determines exactly one value of B. B is said to be functionally dependent on A. For example:

```
EMPLOYEE-NUMBER ----> EMPLOYEE-NAME.
```

multivalued dependencies
Selects all multivalued dependencies (MVDs) most recently merged in the WBDA.

A multivalued dependency A----> > B holds from A to B if, for each value of A, a variable number of values of B may be determined. B is said to be multiply dependent upon A. For example:

```
EMPLOYEE-NAME ---->> EMPLOYEE-CHILD-NAME
```

The double-headed arrow (---->>) indicates that the employee has more than one child.

implied functional dependencies
Selects all implied functional dependencies most recently merged in the WBDA. If any data element, which appears in a dependency described in the WBDA, is a GROUP, and another data element in the WBDA is one of the CONTAINed members, an 'implied' FD is automatically generated from the GROUP data element to the CONTAINed data element, and unless it already exists there, the implied FD is added to the WBDA.

For example, if the data element DATE is a GROUP member containing the ITEMs: YEAR, MONTH and DAY, and if the data element, MONTH appears in any dependency in the WBDA, then, unless already present, the implied FD DATE----> MONTH is added to the WBDA.

domain dependencies
Selects all domain dependencies (DDs) most recently added to the WBDA.

A domain dependency holds from A to B if A is a subcategory of B, that is, every value of A is also a value of B. It implies that any data element which is functionally dependent on B is also functionally dependent on A.

For example, if you have the two data elements: EMPLOYEE-NUMBER and MANAGER-NUMBER and you want to express the fact that the set of MANAGER-NUMBERs is itself a subset of the set of EMPLOYEE-NUMBERs, then you can do this by defining the DD:

```
MANAGER-NUMBER ====> EMPLOYEE-NUMBER.
```

root records

Selects all root records most recently added to the WBDA. A root record is a record with no parents. That means a root record is not the target record in any multivalued association, and it does not contain the key of any other record within its key (in a hierarchical association).

The purpose of identifying root records is to help you to select good seed for the Consolidated Network Plot, that is, seeds which permit the display of the network in a hierarchically top-down direction.

Entity-M Create Userview/Entity Report - Added by Latest Merger (Option 2.3.8)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TW38000                        METHODMANAGER                    ToolSet SERVIC

===>

              Entity-M       Create Userview/Entity Report -
                             Added by Latest Merger

entities  ----> +-
userviews ----> -+

Use this panel to display all dependencies relevant to
the seleceted objects. Only those objects merged during
the most recent load procedure are considered.


 ------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ------------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

The function of this panel is fundamentally the same as that provided by panel TW33000. This panel, however, only lists those workbench objects that were loaded from the repository, and merged in the Workbench Design Area (WBDA) during the latest merger.

The function is especially useful for incremental design techniques, when new data from the repository is added to the contents of an already existing WBDA, because it allows you to examine the incremental data only.

A detailed report is produced about the number of data-views (either entities or userviews) in the current workbench, the dependencies (FDs, MVDs and/or DDs) of each individual data-view plus relevant information about these dependencies (including the effects of normalization, if applicable, and the resulting relations and records that represent the dependencies).

Enter any non-blank character to select the workbench contents that you want to list.

entities

Selects all entities which have been loaded into the WBDA most recently.

userview

Selects all userviews which have been loaded into the WBDA most recently.

Usage-M Create Data Element Usage Report - Added by Latest Merger (Option 2.3.9)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TW39000                        METHODMANAGER                ToolSet SERVIC

===> |

              Usage-M        Create Data Element Usage Report -
                             Added by Latest Merger

dependency                      ----> +-
data element without dependency ----> -+

Use this panel to display the dependencies, entities and
userviews that reference the data elements added to the
workbench during the most recent merger.


 ----------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ----------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

The function of this panel is fundamentally the same as that provided by panel TW34000. This panel, however, only lists those workbench objects that were loaded from the repository, and added to the Workbench Design Area (WBDA) during the latest merger.

Enter any non-blank character to select the workbench objects you want.

dependency

Selects detailed information on data elements that are used in dependencies resulting from the latest merger.

data element without dependency

Selects detailed information on data elements that are not used in dependencies resulting from the latest merger.

RECORD Repository Members of the Type RECORD (Option 2.4.1)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TW41000                        METHODMANAGER                  ToolSet SERVIC

===>

              RECORD         Repository members of the type RECORD

select from current workbench  ----> ++++
select stored workbench        ----> --++
automatic repository update    ----> -+-+

 Use this panel to generate IRDS members of the type RECORD based on
 the named records in the active or a stored workbench. If the name
 of a stored workbench is given, the same name is used for the
 generation.

 ----------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ----------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

Use this function to generate repository members of the type RECORD based on named records in the current or a stored Workbench.

select from current workbench

To select all records in the current WBDA for use in the generation of members of the type RECORD, enter:

```
all
```

(default).

To select certain objects, either enter the name or the internal number of a record. If you enter more than one name or number, each must be separated by a comma (,).

select stored workbench

To load a stored workbench into the current WBDA and generate members of the type RECORD, enter the name of a workbench member that is stored on the MP-AID.

If you do not want to generate RECORDS from everything in the reloaded workbench, you can use the previous selection criteria to limit the operation of this function.

automatic repository update

To file the generated members of the type RECORD in the repository, enter any character except a blank space. When the member is filed, a report of the member is output, indicating if there have been problems with filing the member.

RELATION Repository Members of the Type RELATION (Option 2.4.2)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TW42000                        METHODMANAGER                 ToolSet SERVIC

===> |

              RELATION       Repository members of the type RELATION

select from current workbench  ----> ++++
select stored workbench        ----> --++
automatic repository update    ----> -+-+

 Use this panel to generate repository members of the type RELATION
 based on named records in the current or a stored workbench. If the
 name of a stored workbench is given, the same name is used for the
 generation.

 ----------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ----------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

Use this function to generate repository members of the type RELATION based on named relations in the current or a stored Workbench.

select from current workbench

To select all relations in the current WBDA for use in the generation of members of the type RELATION, enter:

```
all
```

(default).

To select certain objects only, either enter the name or the internal number of a relation. If you enter more than one name or number, each must be separated by a comma.

select stored workbench

To load a stored workbench into the current WBDA and generate members of the type RELATION, enter the name of a workbench member that is stored on the MP-AID.

If you do not want to generate RELATIONs from everything in the reloaded workbench, you can use the previous selection criteria to limit the operation of this function.

automatic repository update

To file the generated members of the type RELATION in the repository, enter any character except a blank space. When the member is filed, a report of the member is output, indicating if there have been problems with filing the member.

DB2-Docu Repository Members for DB2 Member Types (Option 2.4.3)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TW43000                        METHODMANAGER                  ToolSet SERVIC

===> 

              DB2-Docu       Repository members for DB2 member types

select relations                        ----> ***
generate TABLE                          ----> +--
generate INDEX                          ----> ***
generate VIEW                           ----> ***+
without referential integrity clause    ----> *--
for DB2-TABLESPACE                      ----> *--
CREATOR-OWNER                           ----> ***
automatic repository update             ----> ***

 ------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ------------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

Use this function to generate DB2 member types from the relations held in the Workbench Design Area (WBDA).

select relations

To select all relations in the current WBDA for use in the generation of DB2 member types in the repository, enter:

`all`

(default).

To select certain relations only, either enter the name or the internal number of a relation. If you enter more than one name or number, each must be separated by a comma (,).

For an overview of valid relation names and their internal numbers in the current WBDA, use panel TW31000, selecting 'relations'.

generate TABLE

To generate DB2-TABLE members from the WBDA, enter: `x`. The names of the DB2-TABLE members are derived automatically from the names of the relations in the WBDA and will additionally conform to the naming conventions.

generate INDEX

To generate DB2-INDEX members from the WBDA, enter: `x`. The names of the DB2-INDEX members are derived automatically from the names of the relations in the WBDA and will additionally conform to the naming conventions.

generate VIEW

To generate DB2-VIEW members from the WBDA, enter: `x`. The names of the DB2-VIEW members are derived automatically from the names of the relations in the WBDA and will additionally conform to the naming conventions.

without referential integrity clause

If you do not want to generate a referential integrity clause, enter:

```
x
```

Explanation: DB2 provides data consistency among tables through referential constraints. The enforcement of referential constraints, called referential integrity, ensures that all references from one table to another are valid. DB2 uses the primary keys and foreign keys in DB2 tables to enforce referential integrity.

for DB2-TABLESPACE

Enter the DB2-TABLESPACE name that you want inserted into the IN clause of the generated definitions.

CREATOR-OWNER

Enter the name of the creator/owner that you want inserted into the CREATOR-OWNER clause of the generated definitions.

automatic repository update

To file the generated DB2 members in the repository, enter any character except blank. When a member is filed, a report of the member is output, indicating if there have been problems with filing the member.

SQL/DS-Docu Repository Members for SQL/DS Member Types (Option 2.4.4)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TW44000                        METHODMANAGER                 ToolSet SERVIC

===> |

              SQL/DS-Docu    Repository members for SQL/DS member types

select relations                     ----> ***
generate TABLE                       ----> +--
generate INDEX                       ----> ***
generate VIEW                        ----> ***
without referential integrity clause ----> *--
for SQL/DS-DBSPACE                   ----> *--
CREATOR-OWNER                        ----> ***
automatic repository update          ----> ***

 ------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ------------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

Use this function to generate SQL/DS member types from the relations held in the Workbench Design Area (WBDA).

select relations

To select all relations in the current WBDA for use in the generation of SQL/DS member types in the repository, enter:

```
all
```

(default).

To select certain relations only, either enter the name or the internal number of a relation. If you enter more than one name or number, each must be separated by a comma (,).

For an overview of valid relation names and their internal numbers in the current WBDA, use panel TW31000, selecting 'relations.'

generate TABLE

To generate SQL-TABLE members from the WBDA, enter `x`. The names of the SQL-TABLE members are derived automatically from the names of the relations in the WBDA and will additionally conform to the naming conventions.

generate INDEX

To generate SQL-INDEX members from the WBDA, enter `x`. The names of the SQL-INDEX members are derived automatically from the names of the relations in the WBDA and will additionally conform to the naming conventions.

generate VIEW

To generate SQL-VIEW members from the WBDA, enter `x`. The names of the SQL-VIEW members are derived automatically from the names of the relations in the WBDA and will additionally conform to the naming conventions.

without referential integrity clause

If you do not want to generate a referential integrity clause, enter `x`.

Explanation: SQL/DS provides data consistency among tables through referential constraint. The enforcement of referential constraints, called referential integrity, ensures that all references from one table to another are valid. SQL/DS uses the primary keys and foreign keys in SQL tables to enforce referential integrity.

for SQL/DS-DBSPACE

Enter the SQL/DS-DBSPACE name that you want inserted into the IN clause of the generated definitions.

CREATOR-OWNER

Enter the name of the creator/owner that you want inserted into the CREATOR-OWNER clause of the generated definitions.

automatic repository update

To file the generated SQL members in the repository, enter any character except blank. When a member is filed, a report of the member is output, indicating if there have been problems with filing the member.

DB2-Table SQL CREATE TABLE Statements for DB2 (Option 2.4.5)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TW45000                          METHODMANAGER                  ToolSet SERVIC

===>

              DB2-Table      SQL CREATE TABLE Statements for DB2

creator            ----> *
database-name      ----> *
tablespace-name    ----> *
edit-proc-name     ----> *
valid-proc-name    ----> *
output-file-name   ----> *
select relations   ----> *
alphabetic         ----> *
format-name        ----> *

 ----------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ----------------------------------------------------------
Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

Use this function to generate CREATE TABLE statements for DB2 tables from the relations held in the Workbench Design Area (WBDA).

creator

To identify the creator of the tables, enter the appropriate MethodManager user-id.

database-name

To specify the database in which the tables are to be created, enter the name of a DB2 database.

tablespace-name

To specify the space in which the tables are to be created, enter the name of a corresponding tablespace.

edit-proc-name

To specify the edit routine that will be invoked whenever a row is retrieved, updated, or inserted, enter the name of that edit routine.

valid-proc-name

To specify the validation routine that will be invoked whenever a row is retrieved, updated, or inserted, enter the name of that validation routine.

output-file-name

To specify the logical output file in which the generated CREATE TABLE statements are to be stored, enter the name of that logical output file.

select relations

To select all relations in the WBDA for use in the generation of CREATE TABLE statements, enter:

```
all
```

(default).

To select certain objects only, either enter the name or the internal number of a relation. If you enter more than one name or number, each must be separated by a comma (,).

For an overview of valid relation names and their internal numbers in the current WBDA, use panel TW31000, selecting 'relations'.

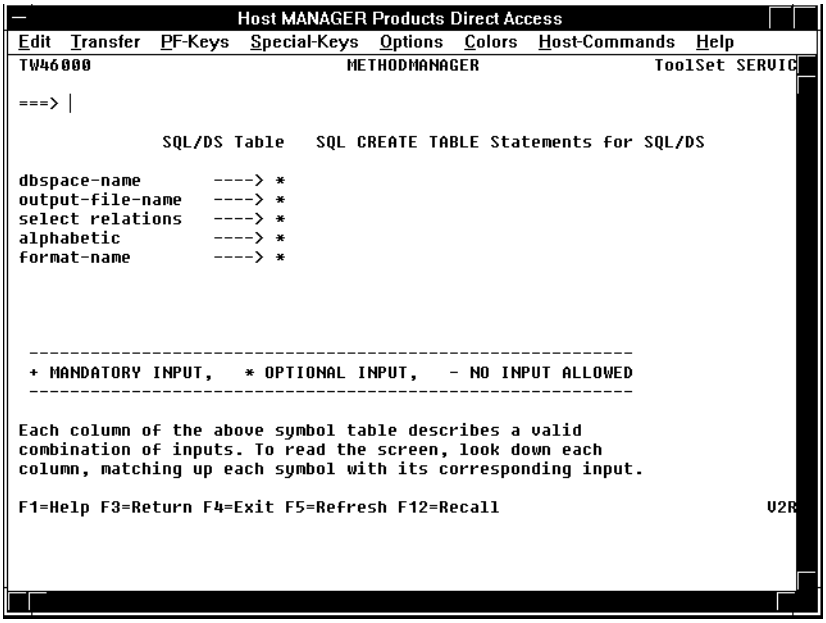
alphabetic

To report any CREATE TABLE statements that were produced from named relations, in alphanumeric order, enter `x`. This will be followed by any commands that were produced from unnamed relations.

format-name

To tailor the way the CREATE TABLE statements are displayed, enter the name of an existing FORMAT member.

SQL/DS Table SQL CREATE TABLE Statements for SQL/DS (Option 2.4.6)



This function is used to generate CREATE TABLE statements for SQL/DS tables from the relations held in the Workbench Design Area (WBDA).

dbspace-name

Enter the name of a dbspace you want to process.

output-file-name
To specify where you want the generated CREATE TABLE commands to be stored, enter the name of a logical output file.

select relations
To use every relation in the WBDA to generate a CREATE TABLE command, enter:

```
all
```

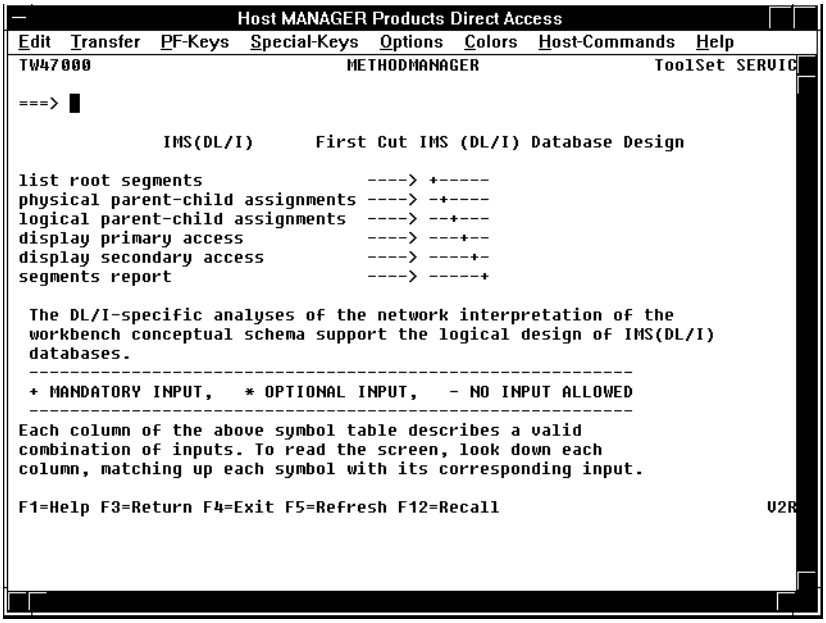
(default).

To select certain objects for generation, either enter the name or the internal number of a relation. If you enter more than one name or number, each must be separated by a comma (,).

alphabetic
To report any CREATE TABLE statements that were produced from *named* relations, in alphanumeric order, enter `x`. Any commands that were produced from *unnamed* relations will then be reported.

format-name
To tailor the way the CREATE TABLE statements are displayed, enter the name of an existing FORMAT member.

IMS(DL/I) First Cut IMS (DL/I) Database Design (Option 2.4.7)



Use this function to produce a fast cut IMS (DL/I) database design, based on the network interpretation of the conceptual schema in the Workbench Design Area (WBDA).

list root segments
To list the DL/I root segments including, for each, its name (if one has been assigned) and number, enter any character except a blank space.

Explanation: the root segments for use as the starting segments of physical hierarchies are automatically identified at the same time as the network records are converted into segments. A root segment is defined as a segment that has no physical parent.

physical parent-child assignments

To display one or more parent-child graphs, each showing the segments of a physical hierarchy, enter any character except a blank space. A separate hierarchy is produced for each root segment whether or not it has any physical dependent segments. Each new hierarchy begins with a root segment appearing in the left-hand corner of a new page.

A root segment which has no physical dependents is displayed as a separate single-segment hierarchy and is a candidate for a root-only physical database.

logical parent-child assignments

To display one or more graphs displaying the unidirectional logical relationships identified by the system, enter any character except blank. Each of these relationships consists of a logical child segment and a logical parent segment, where the former contains the concatenated key of the latter. The output shows each logical child segment pointing to its respective logical parent segment.

display primary access

To display one or more graphs displaying the access paths derived from multivalued dependencies defined in USERVIEW members of the repository and from MULTI-ATTRIBUTES and MULTI-ASSOCIATION clauses specified in ENTITY members, enter any character except a blank space. Each access path displayed indicates that access is required from a key set of data elements in a source segment to a key set of data elements in a destination segment. Such a path is also referred to as a 'primary key' access path to distinguish it from a secondary key access path.

display secondary access

To output a report describing the secondary key access paths derived from multivalued dependencies defined in USERVIEW members of the repository and from MULTI-ATTRIBUTES and MULTI-ASSOCIATION clauses specified in ENTITY members, enter any character except a blank space. Each secondary key access path indicates that access is required from a non- key set of data elements in a source segment to a key of data elements in a target segment.

segments report

To output a report of all the segments generated, enter any character except blank. For each segment reported, the output includes:

- The segment name (if one has been assigned)
- Its number
- Its concatenated key
- Its non-key fields.

The sequence key of each non-root segment can be derived by removing, from its concatenated key, the concatenated key of its physical parent segment. For a root segment, its sequence key is the reported concatenated key.

Relational Relational Cluster Plot (Option 2.6.1)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TW61000                        METHODMANAGER                ToolSet SERVIC

===> |

              Relational     Relational Cluster Plot

selection   ----> +++
relations   ----> ++-
summary     ----> -+-
tables      ----> --+
format name ----> *-*

Use this panel to display a diagram of the relational view of the
conceptual schema. The direct environment is displayed for each
relation.
 ------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ------------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

Use this function to provide, for each selected relation/table in the generated relational schema, a graphical display in cluster form of all its associations with the other relations/tables in the schema, including:

- Direct foreign key associations
- Direct hierarchical associations
- Domain associations.

In domain associations a role relation is generated from a domain dependency (DD). DDs are used to avoid the occurrence of naming problems which can lead to anomalies. DDs have been designed to handle situations where the left-hand side of a DD is perceived as indicating a role played by the right- hand side. For example:

```
BUYER-FIRM ====> MEMBER-FIRM
SELLER-FIRM ====> MEMBER-FIRM
```

The type of association is indicated by the type of arrow used in the plot. In addition, for the subject relation of each cluster, the output indicates all of its data elements, including the key.

selection

To get a description of all relations/tables held in the WBDA, enter:

```
all
```

(default).

To select certain objects, enter either the name or the internal number of a relation. If you enter more than one name or number, each must be separated by a comma.

Note:

If you want to select individual relations by name, make sure that names have been assigned to them previously. Use panel TW28000 to assign names to relations.

For an overview of valid relation names and their internal numbers in the current WBDA, use panel TW31000, selecting 'relations'.

relations

To display a detailed relational cluster plot of the current WBDA, enter any character except a blank space.

summary

To display a summarized relational cluster plot of the current WBDA, enter any character except a blank space.

tables

To get a detailed graphical display for each selected DB2 or SQL/DS table in the DB2 or SQL/DS design in the WBDA, enter any character except blank. If the functional units that provide DB2 or SQL/DS are not installed at your site, this selection has no effect.

format-name

To tailor the way the cluster plot is displayed, enter the name of an existing FORMAT member.

Structure Referential Structure Plot (Option 2.6.2)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TW62000                          METHODMANAGER                    ToolSet SERVIC

===> |

              Structure      Referential Structure Plot

selection                       ----> +
parents                         ----> *  (any non-blank character selects)
dependents                      ----> *           "

Use this panel to output a graphical display of the network view
of the conceptual schema, grouped in sections. The direct
environment is displayed for each record.


 ---------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ---------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                          V2R
```

Use this function to produce a consolidated, overview plot of the referential structures in the DB2 or SQL/DS design in the Workbench Design Area (WBDA).

A referential structure is a set of tables and relationships such that each table in the set is a parent or dependent of itself or some other table in the set. Every table that is a parent or dependent in the set is part of exactly one referential structure.

It depends on your own configuration, if you get the referential structure plot for DB2 or SQL/DS. If the functional units that provide DB2 or SQL/DS are not installed at your site, this function of this panel has no effect.

selection

To get a description of all relations held in the WBDA, enter:

```
all
```

(default).

To select certain objects, enter either the name or the internal number of a relation. Multiple selection is not allowed.

Note:

If you want to select individual relations by name, make sure that names have been assigned to them previously. Use panel TW28000 to assign names to relations.

For an overview of valid relation names and their internal numbers in the current WBDA, use panel TW31000, selecting 'relations'.

The plot is displayed with both, parent and dependent tables and relationships.

parents

To display only parent tables and relationships, enter any character except blank.

dependents

To display only dependent tables and relationships, enter any character except blank.

Network Network Cluster Plot (Option 2.6.3)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TW63000                        METHODMANAGER                  ToolSet SERVIC

===>

              Network        Network Cluster Plot

selection                    ----> ***
summary                      ----> -+-
format name                  ----> --+



 ------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ------------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                V2R
```

Use this function to provide, for each record in the generated network schema, a graphical display in cluster form of all its associations with the other records in the schema, including:

- Direct foreign key associations
- Direct hierarchical associations
- Domain associations
- Multivalued associations
- Secondary key associations.

The type of association is indicated by the type of arrow used in the plot. In addition for the subject record of each cluster, the output indicates all of its data elements, including the key, plus details of all the secondary key associations displayed in the cluster.

selection

To get a description of all records held in the WBDA, enter:

```
all
```

(default).

To select certain objects, enter either the name or the internal number of a record. If you enter more than one name or number, each must be separated by a comma.

Note:

If you want to select individual records by name, make sure that names have been assigned to them previously. Use panel TW28000 to assign names to records.

For an overview of valid record names and their internal numbers in the current WBDA, use panel TW31000, selecting 'records'.

summary
To display a summary network cluster plot, enter any character except a blank space. The default is a detailed network plot.

format-name
To tailor the way the cluster plot is displayed, enter the name of an existing FORMAT member.

Consolidated Consolidated Network Plot (Option 2.6.4)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TW64000                        METHODMANAGER                    ToolSet SERVIC

===> |

              Consolidated   Consolidated Network Plot

select root record            ----> *
select link types             ----> *
select label                  ----> *



 ----------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ----------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

Use this function to display the hierarchical structure of network interpretation of the conceptual schema.

The records and their connecting direct foreign key associations, direct hierarchical associations, role associations and multivalued associations are listed. In addition, a numeric and alphabetic directory, that enables you to locate a record either by logical line number or alphabetically, is output with the plot. Once located you can determine its box and pointer numbers.

select root record
Enter the name or the number of a root record held in the WBDA.

Explanation: a root record is a record that is not the target record in any multivalued association and is not the source record in any foreign key association, nor does its key contain the key of any other record.

In terms of link types it is defined as being a record which can be represented by a box with no double-headed arrows entering it and no single-headed arrows leaving it.

A root record is more likely to be a good seed, that is, one which permits the display of the network in a hierarchical top-down direction. The selection of 'ordinary' records would generally result in a fragmented diagram which could be more difficult to interpret.

select link types

Enter a link code number which represents a corresponding link type to indicate that all occurrences of a particular type of association are to be depicted in the plot. If you enter more than one number, each must be separated by a comma (,).

Valid link codes are:

Link Code	Link Type	Association
1	----->	one
2	---->>	many
3	<-----	one
4	<---->	one-one
5	<--->>	one-many
6	<-H->>	hierarchical
7	<<----	many
8	<<--->	many-one
9	<<-H->	hierarchical
10	<<-->>	many-many
11	=====>	role
12	<=====	role

select label

To control the text displayed within boxes and pointers of the Consolidated Network Plot, enter either:

`name`

or:

`key`

If you enter `name` then the text displayed is the name of the record represented (provided that it had been named).

If you enter `key` then the text displayed is the key of the record represented.

Selection Select another Repository/Status (Option 9.1)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
T210000                    METHODMANAGER                    ToolSet SERVIC

---> 

          Selection      Select another Repository/Status

repository            ----> -++++
password              ---->
status                ----> +--++
processing mode       ----> --+-+

 Use this panel to open a different repository or to change the current
 status within the repository.

 ---------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ---------------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1-Help F3-Return F4-Exit F5-Refresh F12-Recall                    V2R
```

This function is used to open another repository, and to change the status.

repository

To change the repository you are currently working in, enter the name of the repository you want to open. Ask your repository Controller for valid names of other existing repositories.

password

To open the selected repository, enter a valid password.

status

To select the status that you want to work in, enter the name of an existing status. For a list of all valid status names enter STATUS LIST in the Command Area.

processing mode

Enter the required processing mode. The valid modes are:

- UPDATE or U (default)
- READ or R
- SYSNAME *name* or SY *name*
- IBUF N or IB *n* (I/O buffer for index records)
- SBUF *n* or SB *n* (I/O buffer for source records)
- DBUF N or DB *n* (I/O buffer for data entries records).

where:

name is the ddname in OS job control statements referring to an external dataset of the repository
n is an integer between 1 and 255.

Password Change Logon Password (Option 9.2)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TZ20000                        METHODMANAGER                  ToolSet SERVIC

===> |

              Password       Change Logon Password

old password         ---->
new password         ---->

 Use this panel to change your logon password.




 ----------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ----------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

This function is used to change your logon password. To activate the new password, either logoff and logon again, or enter RESTART in the Command Area.

old password

Enter the password currently being used.

new password

Enter the new password which is going to replace your old one. The maximum length of the password is 8 characters.

PF Keys Change Function Key Settings (Option 9.3)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TZ30000                        METHODMANAGER                  ToolSet SERVIC

===>

                       Change Function Key Settings
      PF Key     Text          Description
      F1         Help          call extended help for current panel
      F3         Return        return to previous panel
      F4         Exit          return to previous application point
      F5         Refresh       refresh current panel
      F7         Backward      scroll backward
      F8         Forward       scroll forward
      F12        Recall        recall
      F2         BTM           user 1
      F6         Cancel        user 2
      F9         Cancel        user 3
      F10        Cancel        user 4
      F11        Cancel        user 5

      F4         Run           start command in expert mode
      F3         File          file and return to previous panel
      F4         File          file and return to previous application point

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                          V2R
```

This function displays the current settings of all function (PF) keys.

To change a PF key setting, overwrite the current setting with the new one.

You can specify another function key in the first column (PF Key) and new text in the second column (Text).

For example, you might replace:

```
Fl   Help  call extended help for current panel
```

with:

```
F2  Mayday  call extended help for current panel
```

Note:

The text in the third column (Description) cannot be changed.

A changed setting becomes active for the entire environment once you press ENTER.

Whenever the PF Key setting is changed for the first time, an individual user profile (MDU-MMRPF) is automatically created for the user who changes the setting.

The individual user profile is saved across sessions and executed whenever the user logs on again.

Editor Switch Operating System Editor On/Off (Option 9.6)

```
Host MANAGER Products Direct Access
Edit  Transfer  PF-Keys  Special-Keys  Options  Colors  Host-Commands  Help
TZ60000                        METHODMANAGER                 ToolSet SERVIC

===> |

              Editor         Switch Operating System Editor On/Off

switch editor off  ----> +-
work file          ----> -+
switch editor on   ----> -+

Use this panel to switch the operating system editor on or off.

Workfile = DSN name for IBM OS
         = EDT area for Siemens BS2000

 ------------------------------------------------------------
 + MANDATORY INPUT,   * OPTIONAL INPUT,   - NO INPUT ALLOWED
 ------------------------------------------------------------

Each column of the above symbol table describes a valid
combination of inputs. To read the screen, look down each
column, matching up each symbol with its corresponding input.

F1=Help F3=Return F4=Exit F5=Refresh F12=Recall                        V2R
```

This function is used to switch from the MethodManager editor to the operating system editor and vice versa.

switch editor off

To switch the operating system editor off, enter any character other than a blank space.

work file

To switch the operating system editor on, enter a dataset name (for IBM OS), or an EDT area (for Siemens BS2000).

switch editor on

To switch the operating system editor on, enter any character other than a blank space.

8 Navigation Charts

How To Use the Charts

This chapter provides a list of the panels that you access using the menus, to help you navigate more quickly and efficiently.

For each panel the following information is provided:

- The option number that you enter on the command line to access the panel
- The long title of the panel
- The panel name
- The panel type.

For example:

Option Number	Panel Long Title	Panel Name	Panel Type
1.3	Select a different project	TV50000	selection

Panel type is one of the following:

- Menu for menu panels
- Input for input panels
- Selection for selection panels
- Display for panels that display information
- Action for panels that perform an action when you select them
- Expert for expert mode panels
- Edit panels for panels that you have to edit
- Help for tutorial help panels
- -------- for facilities that are not yet available
- Command for options that end the panel interface session and return you to the command interface.

Refer to Chapter 4, "Using the Panel Interface," on page 21 for details of panel types.

The panels are arranged according to the menu structure. Therefore everything under option `1` of a menu is listed before everything under option `2`. To simplify the option numbers, the panels are split into the following sections.

LifeCycle SERVICES: the option numbers assume that you are working from the Information Engineering menu, V00000.

ToolSet SERVICES: the option numbers assume that you are working from ToolSet SERVICES menu, X00000. If you are working from V00000 all the options should be prefixed by 4.

Tutorial: the option numbers assume that you have selected option `T` from menu V00000 or X00000 and that you are working from the Information Engineering Tutorial menu XT00000.

Strategic Information Planning Project: the option numbers assume that you have selected the *Work on an active project* option (LCS 1.1) and that project ECCAM (see page 244) is the active project.

Note:

The navigation charts include all the panels that you access by selecting menu options. Some panels selected in this way take you into a dialog. For example, if you select TSS 2.5.3, DB2 member types, you can select a DB2 member type. A subsequent list of the existing members of the selected type is displayed. You can select one of these members and update it. The navigation charts do not include panels that are nested in this way.

Lifecycle Services

Information Engineering		**V00000**	**menu**

1	**Work on a project**	**V80000**	**menu**
1.1	Work on active project	project	menu
1.2	Work on tasks	TV30000	input
1.3	Select a different project	TV50000	selection
1.4	Display environment information	TV40000	display
1.U	User defined option	---------	--------

2	**Project management**	**V60000**	**menu**
2.1	Create project	TV61000	input
2.2	Assign life cycle model to project	TV12000	selection
2.3	Assign existing user to project	TV63000	input
2.4	Add and assign new user	TV62000	input
2.5	Exclude user from project	TV64000	selection
2.6	Delete user from repository	TV68000	selection
2.7	Delete project from repository	TV67000	selection
2.8	List all users	TV66000	display
2.9	List projects visible from the current status	TV65000	display
2.A	List all projects	TV69000	display
2.B	**Task management**	**M00000**	**menu**
2.B.1	List tasks	TM10000	input
2.B.2	Monitor project development	TM20000	selection
2.B.3	Create and update TASK members	TM30000	selection
2.B.U	User defined option	--------	--------
2.C	**Review of project members**	**C00000**	**menu**
2.C.1	Deactivate dummies in project-view	TC10000	input
2.C.2	Activate project related members in project	TC20000	input
2.C.U	User defined option	--------	--------
2.U	User defined option	--------	--------

3	**LifeCycle management**	**V70000**	**menu**
3.1	List Life Cycle Models	TV81000	selection
3.2	Print Life Cycle Model documentation	V13000	input
3.3	**Test the consistency of a Life Cycle Model**	**V16000**	**menu**
3.3.1	Phase test list	TV16100	selection
3.3.2	Phase test matrix	TV16300	input
3.3.U	User defined option	--------	--------
3.4	Copy a Life Cycle Model	TV14000	input
3.5	Create or update Life Cycle Model members	TV15000	selection
3.6	Generate, simulate or check a Life Cycle Model	TV76000	selection
3.7	Delete generated MP-AID members	TV77000	selection
3.U	User defined option	--------	--------
3.T	Tutorial	VT20000	menu

4	**ToolSet SERVICES**	**X00000**	**menu**

T	**Tutorial**	**XT0000**	**menu**

X	Exit	--------	command

Toolset Services

Toolset services		**X00000**	**menu**

1	**Repository functions**	**D00000**	**menu**
1.1	**List members by member type**	**TD10000**	**menu**
1.1.1	Strategic Information Planning	TD11000	selection
1.1.2	Application Development	TD11000	selection
1.1.3	DB2 member types	TD11000	selection
1.1.4	SQL/DS member types	TD11000	selection
1.1.5	IMS/DL1 member types	TD11000	selection
1.1.6	IEW/ADW planning workstation members	TD11000	selection
1.1.7	IEW/ADW analysis workstation members	TD11000	selection
1.1.8	IEW/ADW design workstation members	TD11000	selection
1.2	**List members, aliases and catalogs by name**	**D20000**	**menu**
1.2.1	Frequency of use	TD21000	input
1.2.2	Name-related interrogation	TD22000	input
1.2.3	Interrogate time, status and user	TD23000	input
1.2.4	Display all CATALOG classifications	TD24000	input
1.2.5	Display all aliases	TD25000	input
1.2.E	Expert mode	D2E000	expert
1.2.U	User defined option	--------	--------
1.3	**Query attributes**	**D30000**	**menu**
1.3.1	Search for catalog classifications	TD31000	input
1.3.2	Search for aliases	TD32000	input
1.3.3	Search for attributes containing values	TD33000	input
1.3.4	Search for attributes containing text	TD34000	input
1.3.E	Expert mode	D4E000	expert
1.3.U	User defined option	--------	--------

1.4	**Display connections between members**	**D40000**	**menu**
1.4.1	Direct references upwards	TD41000	input
1.4.2	Indirect references upwards	TD42000	input
1.4.3	Direct references downwards	TD43000	input
1.4.4	Indirect references downwards	TD44000	input
1.4.5	Display relationships as matrix	TD45000	input
1.4.E	Expert mode	D4E000	expert
1.4.U	User defined option	--------	--------
1.5	**Create and update members**	**TD50000**	**menu**
1.5.1	Strategic Information Planning	TD51000	selection
1.5.2	Application Development	TD51000	selection
1.5.3	DB2 member types	TD51000	selection
1.5.4	SQL/DS member types	TD51000	selection
1.5.5	IMS/DL1 member types	TD51000	selection
1.5.6	IEW/ADW planning workstation members	TD51000	selection
1.5.7	IEW/ADW analysis workstation members	TD51000	selection
1.5.8	IEW/ADW design workstation members	TD51000	selection
1.6	**Process existing KEPT-DATA lists**	**D60000**	**menu**
1.6.1	Retrieve stored lists	TD61000	selection
1.6.2	Store named lists	TD62000	selection
1.6.3	Delete stored lists	TD63000	selection
1.6.4	Delete named lists	TD64000	selection
1.6.5	Combine two lists	TD65000	input
1.6.U	User defined option	--------	--------

1.U	User defined option	--------	--------
1.T	Repository tutorial	--------	--------

2	**Data modeling and design functions**	**W00000**	**menu**
2.1	**List member by member type**	**TW10000**	**menu**
2.1.1	Strategic Information Planning	TD11000	selection
2.1.2	Application Development	TD11000	selection
2.1.3	DB2 member types	TD11000	selection

2.1.4	SQL/DS member types	TD11000	selection
2.1.5	IMS/DL1 member types	TD11000	selection
2.1.6	IEW/ADW planning workstation members	TD11000	selection
2.1.7	IEW/ADW analysis workstation members	TD11000	selection
2.1.8	IEW/ADW design workstation members	TD11000	selection
2.2	**Workbench manipulation**	**W20000**	**menu**
2.2.1	Display condition of current workbench	TW21000	action
2.2.2	Delete current workbench	TW22000	action
2.2.3	Store workbench on MP-AID	TW23000	selection
2.2.4	Retrieve workbench file from MP-AID	TW24000	selection
2.2.5	Normalization 1NF	TW25000	action
2.2.6	Normalization 2NF	TW26000	action
2.2.7	Normalization 3NF	TW27000	action
2.2.8	Name records/relations	TW28000	action
2.2.9	Load from repository	TW29000	selection
2.2.A	Remove stored workbench file	TW2A000	selection
2.2.B	**Re-model inconsistent workbench**	**W2B000**	**menu**
2.2.B.1	Load from repository	TW2B100	input
2.2.B.2	Normalize 1NF/2NF/3NF	TW2B200	input
2.2.B.U	User defined option	--------	--------
2.2.U	User defined option	--------	--------
2.3	**Analyze workbench**	**W30000**	**menu**
2.3.1	List workbench objects	TW31000	input
2.3.2	List dependencies of workbench objects	TW32000	input
2.3.3	Create userview or entity report	TW33000	input
2.3.4	Create data element usage report	TW34000	input
2.3.5	Create relational schema report	TW35000	input
2.3.6	Create network schema report	TW36000	input
2.3.7	List objects added by latest merger	TW37000	input
2.3.8	Create userview/entity report added by latest merger	TW38000	input
2.3.9	Create data element added by latest merger	TW39000	input
2.3.A	Calculate and evaluate load factors	TW3A000	input
2.3.U	User defined option	--------	--------

2.4	**Generate from workbench**	**W40000**	**menu**
2.4.1	Repository member of the type RECORD	TW41000	input
2.4.2	Repository member of the type RELATION	TW42000	input
2.4.3	Repository members for DB2 member types	TW43000	input
2.4.4	Repository members for SQL member types	TW44000	input
2.4.5	SQL CREATE TABLE statements for DB2	TW45000	input
2.4.6	SQL CREATE TABLE statements for SQL	TW46000	input
2.4.7	First cut IMS(DL/1) database design	TW47000	input
2.4.U	User defined option	--------	--------
2.5	**Create and Update members**	**TW50000**	**menu**
2.5.1	Strategic Information Planning	TD51000	selection
2.5.2	Application Development	TD51000	selection
2.5.3	DB2 member types	TD51000	selection
2.5.4	SQL/DS member types	TD51000	selection
2.5.5	IMS/DL1 member types	TD51000	selection
2.5.6	IEW/ADW planning workstation members	TD51000	selection
2.5.7	IEW/ADW analysis workstation members	TD51000	selection
2.5.8	IEW/ADW design workstation members	TD51000	selection
2.6	**Graphical display of conceptual schema**	**W60000**	**menu**
2.6.1	Relational cluster plot	TW61000	input
2.6.2	Referential structure plot	TW62000	input
2.6.3	Network cluster plot	TW63000	input
2.6.4	Consolidated network plot	TW64000	input
2.6.U	User defined option	--------	--------

2.U	User defined option	--------	--------
2.T	Tutorial	--------	--------

3	**Documentation functions**	**K00000**	**menu**
3.1	List DOCUMENT members	TK10000	selection
3.2	Selection project related DOCUMENT members	TK20000	selection
3.3	Output documents	TK30000	input
3.4	Display dependencies of documents	TK40000	input

3.5	Create and update DOCUMENT members	TK50000	selection
3.U	User defined option	--------	--------
3.T	**Documentation tutorial**	**--------**	**--------**

4	**Import to the repository**	**I00000**	**menu**
4.1	**Import program/parts of program**	**I10000**	**menu**
4.1.1	Generate repository members from COBOL	TI11000	input
4.1.2	Generate repository members from PL/1	TI12000	input
4.1.3	Document record layouts	--------	--------
4.1.U	User defined option	--------	--------
4.2	**Import database systems**	**I20000**	**menu**
4.2.1	Generate repository members from IMS	--------	--------
4.2.U	User defined option	--------	--------
4.3	**Import catalogs of other systems**	**I30000**	**menu**
4.3.1	**DB2 catalog**	**I31000**	**menu**
4.3.1.1	Extract information about DB2 objects	I31100	input
4.3.1.2	Produce a report of the extracted objects	I31200	input
4.3.1.3	Generate repository definitions	I31300	input
4.3.1.4	Insert generated definitions into a repository	I31400	input
4.3.1.U	User defined option	--------	--------
4.3.2	**SQL/DS catalog**	**I32000**	**menu**
4.3.2.1	Extract information about SQL/DS objects	I32100	input
43.2.2	Produce a report of the extracted objects	I32200	input
4.3.2.3	Generate repository definitions	I32300	input
4.3.2.4	Insert generated definitions into a repository	I32400	input
4.3.2.U	User defined option	--------	--------
4.4	Import from other repositories	--------	--------
4.5	**Import from ADW/IEW export files**	**I50000**	**menu**
4.5.1	Show generation options	TI51000	display
4.5.2	Perform import functions	TI52000	input

4.U	User defined option	-------	-------
4.T	Import tutorial	-------	-------

5	**Export from the repository**	**E00000**	**menu**
5.1	**Generation for Programming**	**E10000**	**menu**
5.1.1	Generation of COBOL structures	TE11000	input
5.1.2	Generation of Assembler structures	TE12000	input
5.1.3	Generation of PL/1 structures	TE13000	input
5.1.4	Generate record layout	TE14000	input
5.1.5	DB2 host language data structure	TE31400	input
5.1.6	SQL/DS host language data structure	TE34400	input
5.1.E	Expert mode	E1E3000	expert
5.1.U	User defined option	-------	-------
5.2	**Generation for database systems**	**E20000**	**menu**
5.2.1	ADABAS database	E21000	menu
5.2.1.1	ADABAS LOADER definitions	TE21100	input
5.2.1.2	ADABAS Record Layouts	TE21200	input
5.2.1.3	Record and Format Buffer	TE21300	input
5.2.1.U	User defined option	-------	-------
5.2.2	**IMS database**	**E22000**	**menu**
5.2.2.1	Produce database description (DBD)	TE22100	input
5.2.2.2	Produce program specification block (PSB)	TE22200	input
5.2.2.3	Produce Segment I/O area	TE22300	
5.2.2.U	User defined option	-------	-------
5.2.3	SESAM database	TE23000	input
5.2.4	SUDS database	TE24000	input
5.2.U	User defined option	-------	-------
5.3	**Data transfer to catalog of other systems**	**E30000**	**menu**
5.3.1	**DB2 catalog**	**E31000**	**menu**
5.3.1.1	DB2 database administration	TE31100	input
5.3.1.2	Generate SQL ALTER TABLE statements	TE31200	input
5.3.1.3	DB2 security administration	TE31300	input
5.3.1.4	Host language data structures	TE31400	input
5.3.1.U	User defined option	-------	-------
5.3.2	IDMS directory	TE32000	input

5.3.3	PREDICT directory	TE33000	input
5.3.4	**SQL/DS catalog**	**E34000**	**menu**
5.3.4.1	SQL/DS database administration	TE34100	input
5.3.4.2	SQL ALTER command	TE31200	input
5.3.4.3	SQL/DS security administration	TE31300	input
5.3.4.4	Host language data structures	TE31400	input
5.3.4.U	User defined option	-------	-------
5.3.N	Directories of other Systems	TE35000	input
5.3.U	User defined option	-------	-------
5.4	Data transfer to other MMR repositories	-------	-------
5.5	Generation of MVW diagram members	TE50000	input
5.6	**ADW/IEW export**	**E60000**	**menu**
5.6.1	Show generation options	TE61000	display
5.6.2	Perform export functions	TE62000	input
5.6.3	Name-related interrogation	TD22000	input
5.6.4	**Catalog, alias or attribute interrogation**	**D30000**	**menu**
5.6.4.1	Search for catalog classifications	TD31000	input
5.6.4.2	Search for aliases	TD32000	input
5.6.4.3	Search for attributes containing values	TD33000	input
5.6.4.4	Search for attributes containing text	TD34000	input
5.6.4.E	Expert mode	D4E000	expert
5.6.4.U	User defined option	-------	-------
5.U	User defined option	-------	-------
5.T	Tutorial	-------	-------

6	Screen generation functions	-------	-------
7	Code generation functions	-------	-------

8	**System administration**	**A00000**	**menu**
8.1	**Selection members by member type**	**TD10000**	**menu**
8.1.1	Group of RIM member types	TD11000	selection
8.1.2	Functional member types	TD11000	selection
8.1.3	Other member types	TD11000	selection

8.2	**Create and update members**	**TD50000**	**menu**
8.2.1	Group of RIM member types	TD51000	selection
8.2.2	Functional member types	TD51000	selection
8.2.O	Other member types	TD51000	selection
8.3	**Enable ToolSet SERVICES**	**A70000**	**menu**
8.3.1	Selection hierarchy member	TA71000	selection
8.3.2	Enable ToolSet SERVICES	TA72000	selection
8.3.3	Re-enable ToolSet SERVICES	TA73000	selection
8.3.4	Tailor menus, panels and screens	TA74000	selection
8.3.5	Remove execs from the MP-AID	TA75000	selection
8.3.6	Enable tutorial index	TA76000	selection
8.3.7	Enable ToolSet SERVICES in batch	A77000	selection
8.3.8	Enable HDS Tables	A78000	selection
8.3.U	User defined option	-------	-------
8.U	User defined option	-------	-------
8.T	Administration tutorial	-------	-------

9	**Set and query MMR environment**	**Z00000**	**menu**
9.1	Selection another repository/status	TZ10000	input
9.2	Change logon password	TZ20000	input
9.3	Change function key settings	TZ30000	input
9.4	**Change your MMR profile**	**Z40000**	**menu**
9.4.1	**Display or set MMR general profile**	**Z41000**	**menu**
9.4.1.1	Customizing the command interface	Z41100	edit
9.4.2	**TSS - Display or set MMR profile variables**	**Z42000**	**menu**
9.4.2.1	Customizing the retain options	Z42100	edit
9.4.2.2	Customizing assisted update	Z42200	edit
9.4.2.3	Customizing naming conventions for members	Z42300	edit
9.4.2.4	Activating user exits for TSS	Z42400	edit
9.4.2.5	Customizing the panel interface	Z42500	edit
9.4.2.6	Customizing the workbench design area	Z42600	edit
9.4.3	**LCS - Display or set MMR profile variables**	**Z43000**	**menu**

9.4.3.1	Customizing the panels used under LCS	Z43100	edit
9.4.3.2	Customizing member types relevant for LCS	Z43200	edit
9.4.3.3	Customizing clauses of LCS member types	Z43300	edit
9.4.3.4	Customizing the duration of a Task/Project	Z43400	edit
9.4.3.5	Customizing member type relationships	Z43500	edit
9.4.3.6	Customizing project management functions	Z43600	
9.4.4	User defined option	-------	-------
9.5	Display current user information	TZ50000	display
9.6	Switch operating system editor on/off	TZ60000	input
9.7	**Create batch calls**	**Z70000**	**menu**
9.7.1	**Set up a print job**	**Z71000**	**menu**
9.7.1.1	Edit a print job	-------	edit
9.7.1.2	Initialize a print job	-------	action
9.7.1.U	User defined option	-------	-------
9.7.2	Create and execute commands in batch	Z72000	menu
9.7.2.1	Edit batch job	-------	edit
9.7.2.2	Execute batch job	-------	action
9.7.2.U	User defined option	-------	-------
9.7.U	User defined option	-------	-------
9.8	Environment	TZ80000	display
9.U	User defined option	-------	-------
9.T	Settings tutorial	-------	-------

U	User defined option	-------	-------
T	Tutorial	XT00000	menu
E	Exit	-------	command

Tutorial

Information Engineering		**XT00000**	**menu**

1	**Introduction**	**XT01000**	**menu**
1.1	What is MethodManager	XT01100	help
1.2	Features	XT01200	help
1.3	Benefits	XT01300	help
11	What is MethodManager	XT01100	help
12	**Automating information engineering**	**XT02000**	**menu**
12.1	Competitive solutions for your business	XT02010	help
12.2	Enabling IBM's AD/Cycle - today	XT02020	help
12.3	A tailorable, repository-driven environment	XT02030	help
12.4	An independent and active repository	XT02040	help
12.5	Cooperative processing with multiple platforms	XT02050	help
12.6	The benefits you get from MethodManager	XT02060	help
13	**Repository Tutorial**	**MPRBC0003**	**menu**
13.1	**Introduction to Repositories**	**MFRBC1100**	**menu**
13.1.1	Introduction	MPRBC1110	help
13.1.2	What is a repository?	MPRBC1120	help
13.1.3	What is a repository used for?	MPRBC1130	help
13.1.4	How does a repository fit in your systems	MPRBC1140	help
13.2	Repository functions	MPRBC12100	help
13.3	Repository SERVICES	MPRBC1220	help
13.4	MethodManager repository: basic concepts	MPRBC1400	menu

2	**LifeCycle SERVICES**	**VT00000**	**menu**
2.1	**Introduction to LifeCycle SERVICES**	**VT10000**	**menu**
2.1.1	Concepts	VT10030	help
2.1.2	Illustration	VT10010	help
2.1.3	Benefits	VT100220	help

2.2	**Life Cycle management**	**VT22000**	**menu**
2.2.1	Member Types Defining a Life Cycle Model	VT21000	help
2.2.2	Enabling LifeCycle SERVICES	VT22000	help
2.2.3	**Instructions which Produce Deliverables**	**VT23000**	**menu**
2.2.3.1	Introduction	VT23010	help
2.2.3.2	Macros	VT23020	menu
2.2.3.3	Commands	XT70000	menu
2.2.4	Example of a Life Cycle Model	VT24000	menu
2.2.5	Producing documentation	VT25000	help
2.2.6	Procedures for creating and maintaining Life Cycle Models	VT26000	help
2.3	Project Management	-------	-------

3	ToolSet SERVICES	XT30000	help
4	Repository SERVICES	MPRBC1220	help

5	**Manager Methods Tutorial**	**VM0000**	**menu**
5.1	**Strategic Information Planning**	**VM2000**	**menu**
5.1.0	General Description of the Methods	L50	help
5.1.1	Entity Analysis	L51	menu
5.1.2	Functional Analysis and Design	L52	menu
5.1.8	Affinity Analysis	L58	menu
5.1.9	Economic Appraisal of Information System Projects	--------	--------
5.2	Application Development	--------	--------
5.U	User Defined methods	--------	--------

6	How to move between panels	XT60000	help
7	Types of panel	XT40000	help
8	Line Commands: Summary	XT70080	help
9	PF-key settings	XT50000	help

Strategic Information Planning

ECCAM International Suits Corporation - SIP study		**MMRSIP**	**menu**

1	**Gaining Commitment**	**M21**	**menu**
1.1	**Establish scope of study roles**	**M211**	**menu**
1.1.1	Establish business areas	M2111	selection
1.1.2	Set goals for study	M2112	selection
1.1.3	Develop business reasons	M2113	selection
1.1.4	Perform risk analysis	M2114	selection
1.2	**Define the study roles**	**M212**	**selection**
1.3	Develop schedule for study	M213	selection
1.4	**Make a presentation to management**	**M214**	**menu**
1.4.1	Prepare presentation	M2141	selection
1.4.2	Make presentation	M2142	selection
1.5	Review study proposals	M215	selection
1.6	Confirm commitment	M216	selection

2	**Short strategy study**	**M22**	**menu**
2.1	**Prepare for the study**	**M221**	**selection**
2.1.1	Orient the team	M2211	selection
2.1.2	Educate the team	M2212	selection
2.1.3	Review study goals	M2213	selection
2.1.4	Outline final report	M2214	selection
2.1.5	Determine facts	M2215	selection
2.1.6	Selection and orient interviewees	M2216	selection
2.1.7	Set project management controls	M2217	selection
2.2	**Construct the organization model**	**M222**	**menu**
2.2.1	Identify and define organizational units	M2221	selection
2.2.2	Identify and define goals	M2222	selection
2.2.3	Relate organizational units to goals	M2223	selection
2.2.Q	Validate organizational model	M222Q	selection

2.3	**Construct the functional model**	**M223**	**menu**
2.3.1	Identify and define functions	M2231	selection
2.3.2	Relate functions to organizational model	M2232	selection
2.3.3	Publish results	M2233	selection
2.3.Q	Validate functional model	M223Q	selection
2.4	**Construct the entity model**	**M224**	**menu**
2.4.1	Identify and define entities	M2241	selection
2.4.2	Define associations of entities	M2242	selection
2.4.3	Derive entity subject areas	M2243	selection
2.4.4	Refine entity model	M2244	selection
2.4.Q	Validate entity model	M224Q	selection
2.5	**Define information architecture**	**M225**	**menu**
2.5.1	Relate functions to entities	M2251	selection
2.5.2	Refine entity subject areas	M2252	selection
2.5.3	Refine function clusters	M2253	selection
2.5.4	Refine information architecture	M2254	selection
2.5.Q	Validate information architecture	M225Q	selection
2.6	**Develop applications portfolio**	**M226**	**menu**
2.6.1	Review current systems support	M2261	selection
2.6.2	Define current use of data	M2262	selection
2.6.3	Identify proposed new systems	M2263	selection
2.6.Q	Validate applications portfolio	M226Q	selection
2.7	**Interview executives**	**M227**	**menu**
2.7.1	Make general preparations	M2271	selection
2.7.2	Prepare each interview	M2272	selection
2.7.3	Conduct each interview	M2273	selection
2.7.4	Analyze each interview	M2274	selection
2.7.Q	Validate executive interviews	M227Q	selection

2.8	**Determine architecture priorities**	**M228**	**menu**
2.8.1	Review the fact-gathering for completeness	M2281	selection
2.8.2	Analyze functions, requirements, entities and systems	M2282	selection
2.8.3	Determine findings and conclusions	M2283	selection
2.8.4	Perform a high level economic appraisal	M2284	selection
2.8.5	Review prioritization criteria	M2285	selection
2.8.6	Review system and data priorities	M2286	selection
2.8.Q	Validate architecture priorities	M228Q	selection
2.9	**Report the results of the study**	**M229**	**menu**
2.9.1	Review report outline and categories	M2291	selection
2.9.2	Develop recommendations	M2292	selection
2.9.3	Prepare report	M2293	selection
2.9.4	Present the report to executives	M2294	selection
2.9.5	Review and refine the report	M2295	selection
2.9.6	Publish the study report	M2296	selection
2.Q	Validate short strategy study	M22Q	selection

3	Full Strategy Study	M23	-------
4	Tactical Implementation Planning	M24	-------
5	Ongoing Update	M25	-------

Index

Symbols

&LPA level variable 91
&NPA level variable 91
&PAG level variable 91
&RLPA level variable 91
&RNPA level variable 91
&RPAG level variable 91
&T level variable 91
* line command 99
/ line command 99
? line command 102
??ARRAY command 68
??BODY command 53
??CBPATH command 62
??CFPATH command 61
??CUPATH command 63
??ESCAPE command 71
??EXTRACT command 49
??FIXED command 78
??FLOAT command 78
??FUNCTION command 69
??INCLUDE command 43
??JUSTIFY command 77
??LINCOUNT command 76
??LINE command 73
??LMAR command 74
??LMASK command 73
??LPAGE command 82
??MASK command 73
??PAGE command 76
??PARA command 81
??PARAMETER command 67
??PATH command 63
??PBOT command 75
??PCTL command 80
??PRINT command 79
??PROCEND command 74
??PSET command 91
??PTOP command 75
??RMAR command 75
??RMASK command 73
??SHIFT command 82
??SHOW command 70, 93
??SKIP command 75
??SPACE command 77
??TITLE command 83
??TITLE DEFINE command 85
??TLINCOUNT command 76, 89
??TLMAR command 88
??TLMASK 88
??TLPAGE command 89
??TPAGE command 89
??TPBOT command 88
??TPRINT command 89
??TPSET command 89
??TPTOP command 88
??TRMAR command 88
??TRMASK 88
??TSHIFT command 89
??TSKIP command 88
??TVSET command 89, 92
??UPPER command 81
??VARIABLE command 93
??VSET command 92

A

ACCEPT command 34
accessing
    LifeCycle SERVICES 14
    ToolSet SERVICES 15
ACT line command 100
ACTIVATE command 102
add new line 100
ADIS line command 100
ADISPLAY command 102
AGET command 102
AGETC command 102
analysis of systems 6
ASM command 102
assisted display
    command 100
    panel 33
assisted update
    command 100

panel 34
attributes 7
AUPD line command 34
AUPDATE command 103
automatic recovery system 8

B

B line command 99
backup 8
build contents of document 38, 104

C

C line command 99
CANCEL command 34
clauses 8
COB line command 100
COBOL 103
command summary 99
construction of a system 6
conventions page vi
copy
line 99
Scratchpad 100
create document 39, 104

D

D line command 100
DACT line command 100
DCUPD line command 39
DEACTIVATE command 103
define document
first text line 41
last text line 41
delete line 100
design of system 6
DIS line command 100
display
command 103
panel 33
DISPLAY command 103
DOC command 94
DOCUQUERY command 97
dummy member 8
duplicate document
line 100

E

editing command 99
encoded record 8
entity 7
entity-association (EA) modeling 7
entity-relationship (ER) modeling 7
exit 16
expert input panels 30
extract clauses for document 49

F

FDOC command 96
field help 102
function keys 26

G

generate assembler
command 102
line command 100
generate COBOL
command 103
line command 100
generate PL 1
command 103
line command 101

H

HARDCOPY command 38
help commands 34

I

I line command 100
include member in project view
command 102
line command 100
includes references between documents 43
in-context help panel 29
INFOBANK command 102
InfoBank help 102
Information Model 3
Input panel 28
insert
blank line in document 73
empty line 100
insert Scratchpad
after line 99–100
before line 99
following line 99
interfaces to the repository 11
interrogate
alias
command 104
line command 101
catalogs
command 104
line command 101
direct and indirect references
from member
command 103
line command 101

to member
command 103
line command 101

K

K line command 100
KEPT-DATA lists 9

L

LCS command 14
leaveing panel interface 16
life cycle
analysis 6
construction 6
design 6
LifeCycle SERVICES 1
line command summary 100
line commands 100
list members
by alias
command 104
line command 101
by catalog
command 104
line command 101
list references
direct and indirect
from member
command 103
to member
command 103
logging 8
LOOKASIDE buffer 33

M

M line command 100
mainframe backup facilities 8
Manager Products Repository 3
MDRIM 3
member protection 9
member type 7
help 102
members 5
Menu panel 27
MER line command 101
MERE
command 103
line command 101
MERGE command 103
MERU
command 103
line command 101
MERUE
command 103
line command 101
meta-data 5
MethodManager Dictionary/Repository
Information Model 3
MMRVIEW command 103
modeling techniques 7
models
changing 7
entering 8
interrogating 9
properties 9
protecting 9
storing 7
types 7
Move lines 100
MP-AID 9
MPR DOCUQUERY command 97
MTHELP command 102

O

owning members 9

P

PANEL command 102
panel interface
accessing 14
function key settings 26
leaving 16
panels
display 33
expert input 30
in-context help 29
input 28
menu 27
selection 32
update 34
PLI
command 103
line command 101
position document frame 73
primary commands 102
print new page of document 79
process DOCUMENT 94
producing document
numbered headings 83
page numbers 91
table of contents 88
proposed members 100
protecting members 9
PRP
command 103
line command 101

PRPD
command 103
line command 101
PUR line command 101
PURGE command 103

Q

Q line command 100

R

R line command 100
REF line command 101
REFA
command 103
line command 101
REFERENCES command 103
RELABEL command 103
Relationship 7
remove member from project view
command 103
line command 100
rename proposed member
command 103
line command 101
replacing with proposed member
command 104
line command 101
repository
introduction 1
purpose of 5
SERVICES 3
user interfaces 11
RETRACE command 102
RIGN
command 103
line command 101
routing commands 102
RREN
command 104
line command 101
RREP
command 104
line command 101

S

Scratchpad 99
SDOC command 95
security levels
general 9
specific 9
SELECT command 102
selecting InfoBank panel 102
selection panel 32
source record 8
statuses 7
storing models 7
structure of
LifeCycle SERVICES 17
Strategic Information Planning 18
ToolSet SERVICES 19
Tutorial 20
symbols 33
system development cycle
overview 5

T

TOAUPD command 104
topic help 102
TSS command 14

U

update panel 34
US
line command 101
USA line command 101
USAGE command 104
USR line command 101

V

valid input combinations 102

W

WAL
command 104
line command 101
WF
command 104
line command 101

X

X line command 100