ASG-Manager Products™
Dictionary/Repository User's Guide

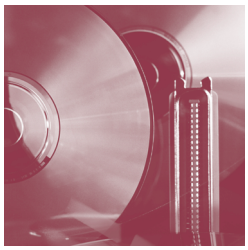
Version: 2.5

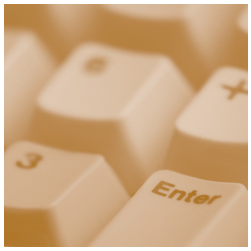
Publication Number: MPR0200-25-DRUG

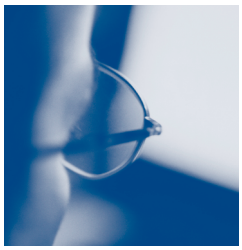
Publication Date: December 2000

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 . . . . . **xiii**

About this Publication . . . . . **xv**

Publication Conventions . . . . . **xv**

1 The Corporate Dictionary/Repository **1**

What is a Dictionary/Repository? **2**
What is a Dictionary/Repository Used for? 3

What is a Corporate Dictionary/Repository? **4**

Dictionary Management Software **5**
Basic Dictionary Management 5
Extended Dictionary Management 5
Additional Dictionary Management Facilities 5
CASE and the Corporate Dictionary/Repository 6

Basic Concepts **9**
Dictionary Member Types 9
Common Clauses 10
Dictionary Member Records 10
Physical Structure of a Dictionary/Repository 14
The Dictionary Controller 16
The Security System 17
The Automatic Recovery System 21

2 What Do You Want To Do? **23**

Accessing or Leaving a Repository **23**

Defining the Targeting Mode **23**

Updating the Repository **24**

Protecting Repository Members **24**

Interrogating the Repository **25**
Finding Out What an Index-name Is 25
Listing Index-names 25

Finding Out About the Relationships Between Members 25
Finding Out About Catalogs, Aliases, Attributes and Clauses of Members 26
Finding Out About the Ownership of Members 26
Finding Out About the Member Types Available in Your Repository. 26

Displaying Dictionary Members 27

Selecting Index-names by Name, Time, User and Status 27

Manipulating KEPT-DATA Lists 27

Performing a Series of Commands 28

Reading From, and Writing To, CMS Files and MVS Datasets 28

3 Using the Dictionary 29

Introduction to Dictionary Management Commands 29

Access Security Commands 31

Dictionary Manipulation Commands 33
Commands Which Enter Members Into a Dictionary. 33
Commands Which Alter or Remove Members of a Dictionary 35
Commands Which Manipulate KEPT-DATA Lists 36

Documentation Commands 38

Generating Output in Alternative Formats 39

Targeting Commands 40

Interrogation Commands. 41

Commands Which Activate Other Commands 43

Effects of the Status Facilities on Dictionary Management 44

Effects of User Defined Syntax on Dictionary Management. 44

Controller's Commands 45

4 Dictionary Management Commands 47

ADD (CMR-CM01). 49
How to Insert and Encode a Member 49
How to Number Lines in a Member's Source Record 50
ADD Syntax 51

ALSO 52

ALTER (CMR-CM01) 52
Introduction to the ALTER Command 52
Renumbering Lines (RENUMBER) 54
Deleting a Member's Current Source Record (WIPE-OUT) 54

Controlling Output from a Command 55
Inserting a New Line 55
Deleting a Line (DELETE) 55
Replacing a Line 56
Copying a Line (COPY) 57
Moving a Line (MOVE) 58
Replacing a String (CHANGE) 59
Filling a Line or Column with Spaces (BLANK) 59
Filling Columns of Text with a Specified String (FILL) 60
Checking Changes Before they are Implemented (DISPLAY) 61
The LOCATE Feature 62
Renumbering Lines (RENUMBER FROM, THRU, TO) 63
Terminating an ALTER Command 65
Examples of the ALTER Command 65
Messages and Error Correction 66
ALTER Syntax 67

AMEND (CMR-CM01) 69
Introduction to the AMEND Command 69
Adding a Keyword or Clause 70
Deleting a Keyword or Clause 71
Changing Text in a Clause 71
Appending to a Clause 72
Replacing a Clause 73
Amending Part of a Clause 73
Displaying an Amended Member 75
Performing a Dummy Amendment 75
Ignoring Erroneous Amendments 75
Making the Same Amendment to Many Members 76
Amending Free-Form Text Clauses 76
AMEND Command Examples 76
AMEND Syntax 77

AUTHORITY (CMR-CM01) 78
AUTHORITY Command Syntax 80

BULK ENCODE (DMR-DD1) 80
Introduction to the BULK ENCODE Command 80
Checking Syntax and Integrity 82
Encoding Locked Members 82
Encoding Members in a KEPT-DATA List 82
Encoding Particular Members 83
Encoding Unencoded Source Records 83
Encoding Dummy Members 84
Encoding Non-Dummy Members 84
Encoding Specific Member Types 84
Encoding Members in Alphanumerical Order 85
Encoding Members in a Status Environment 85

Status, Name, Time and User-Related-Selections 86
Encoding Members Based on a Specific Member Type 86
Encoding Members That Have Passed or Need Integrity Checks 86
Preserving Previous Encode Details 86
BULK ENCODE Command Examples 87
BULK ENCODE Syntax. 88

BULK PRINT (DMR-DD1) 91
Introduction to the BULK PRINT Command. 91
Printing Members in a Status Environment 93
Printing a Member's Source Record as it was at the Time it was Last Successfully Encoded 93
Printing Locked Members. 94
Printing Members in a KEPT-DATA List 94
Printing Particular Members 94
Printing Source Members 95
Printing Dummy Members 95
Printing Non-Dummy Members 96
Printing Particular Member Types Only 96
Processing Members Alphabetically. 97
Format of the Output. 97
Status, Name, Time and User-Related-Selection 98
Printing Members Based on a Specific Member Type 98
Selecting Members That Have Passed or Need Integrity Checks. 98
Bulk Print Examples 98
Bulk Print Syntax 99

BULK REPORT (DMR-DD1) 100
Introduction to the BULK REPORT Command. 100
Reporting a Member and the Members That it Uses Via an EA Relationship Type CONTAINS. 102
Reporting a Member in a Status Environment 102
Reporting Locked Members 103
Reporting Members in a KEPT-DATA List. 103
Reporting Particular Members 104
Reporting Dummy Members. 104
Reporting Non-Dummy Members 105
Reporting Specific Member Types 105
Processing Members Alphabetically. 106
Inserting a Blank Line or Page Break Between Reports. 106
Reporting a Member and Listing the Members That it Uses Via Any Relationship Type. 107
Output for an Encoded Member 107
Output for a Dummy Member. 108
Name, Status, Time and User-Related Selections. 109
Selecting Members That Have Passed or Need Integrity Checks. 109
Reporting Members Based on a Specific Member Type 109

BULK REPORT Examples 109
BULK REPORT Syntax 110

CHECK (MMR-PF61) 111
Introduction to the CHECK Command 111
Encoding Members 112
Removing Members 112
CHECK Syntax 112

CLOSE Command 113
Closing a Dictionary 113
Closing an External Dataset 114
CLOSE Syntax 115

CONTRAST. 115
Output 116
CONTRAST Syntax 116
CONTRAST Examples 117

COPY (CMR-CM01) 117
Copy Syntax 119

DICTIONARY (CMR-CM01) 119
How To Open a Dictionary 119
Dictionary Opening Modes 121
Effects on Opening Modes of the Logging of All Commands and/or Messages 122
Opening a Dictionary Using the SYSNAME Clause 122
How to Specify Dictionary Buffer Pool Sizes 123
Monitoring Buffer Steals. 124
How a DICTIONARY Command is Validated 124
DICTIONARY Syntax 125

DMS 125

DOES (DMR-DD1) 126
DOES Syntax 126

DROP (DMR-DD1) 127
DROP Syntax 128

ENCODE (CMR-CM01) 129
How to Encode a Member. 129
Encoding Members in a Status Environment 130
ENCODE Syntax 131

ERASE KEPT-DATA (DMR-DD1). 131
ERASE KEPT-DATA Syntax. 132

FETCH KEPT-DATA (DMR-DD1) 132
FETCH KEPT-DATA Syntax. 133

FILE (CMR-FE01) 133
Error and Warning Messages 134

FILE Syntax 135

FORMAT AS (CMR-CM01) 135
FORMAT AS Syntax 136

GLOSSARY (DMR-DD1) 136
Introduction to the GLOSSARY Command 136
Outputting Common Clauses 138
Outputting Clauses Specific to Given Member Types 139
Selecting or Excluding Specific Member Types. 140
Selecting Aliases 141
Selecting a List of Member Names 141
Selecting Dummy Members 142
Selecting Non-Dummy Members 142
Selecting Locked Members. 143
Selecting KEPT-DATA Lists 143
Outputting Members that Use or are Used By Selected Members 144
Outputting Only Those Members With a Given Clause 145
Name, Status, Time and User-Related-Selections 145
Displaying Output Alphabetically. 145
Suppressing Headings and Total-Lines. 146
Suppressing Member-Type and Top-Level Clause Names 146
Selecting Members and Aliases from All Statuses 146
Selecting Members That Have Passed or Need Integrity Checks. 147
GLOSSARY Command Examples 147
GLOSSARY Syntax 148

INSERT (CMR-CM01). 152
How to Insert a Member 152
How to Number Lines in a Member's Source Record 154
Inserting and Displaying a Source Record 155
INSERT Syntax. 155

KEEP and ALSO (DMR-DD1) 156
KEEP Syntax. 158

LIST (CMR-CM01) 158
Introduction to the LIST Command 158
Listing Members in a Status Environment 160
Listing Index-names With Details of When They Were Inserted and Encoded. 161
Listing Members With Details of When and By Whom They Were Locked. 162
Listing Currently Locked Members 162
Listing Named Members. 163
Listing Index-Names in a KEPT-DATA List 163
Listing Index-Names Alphabetically 164
Listing Source Members 164
Listing Dummy Members 164
Listing Real, that is, Non-dummy Members. 165
Listing or Excluding Index-names and Member Types 165

Name, Status, Time, and User-Related-Selections 167
Example Output 167
Description of the Output 167
Modifying the Output 169
Selecting Members That Have Passed or Need Integrity Checks. 170
Listing Members Based on a Specific Member Type. 170
LIST Command Examples 171
LIST Syntax 172

LOCK (CMR-WS01) 173
How to Create a New Member and Lock it 173
How to Lock an Existing Member 174
How to Renew a Lock. 175
How to Release a Lock on a Member. 175
LOCK Syntax 176

MODE, QUERY MODE, DMS, WBDA (CMR-CM01) 177
MODE Syntax. 178

MODIFY (CMR-CM01). 178
Introduction to the MODIFY Command. 178
Renumbering Lines (RENUMBER) 180
Deleting a Member’s Current Source Record (WIPE-OUT) 181
Inserting a New Line. 181
Deleting a Line (DELETE). 182
Replacing a Line 182
Copying a Line (COPY) 183
Moving a Line (MOVE) 184
Replacing a String (CHANGE). 185
Filling a Line or Column with Spaces (BLANK). 186
Filling Columns of Text with a Specified String (FILL) 186
Checking Changes Before they are Implemented (DISPLAY) 187
The LOCATE Feature. 188
Renumbering Lines (RENUMBER FROM, THRU, TO). 189
Terminating a MODIFY Command 191
Examples of the MODIFY Command 192
MODIFY Syntax. 193

PERFORM (DMR-DD1) 195
Introduction to the PERFORM Command 195
Performing Commands in a COMMAND-STREAM Member 196
Performing Commands in Input-Lines 197
Replacing Index-Names with the Perform Character 198
Selecting Members that are Locked 199
Selecting Members with Source Record. 199
Selecting Dummy Members 200
Selecting a List of Non-Dummy Members. 200
Selecting a List of Named Members. 201

Selecting Index-Names in a KEPT-DATA List 201
Selecting a List of Index-Names ALPHABETICALLY 202
Selecting or Excluding Index-Names and Member Types 203
Name, Status, Time and User-Related-Selections 203
The PRINT, PRINT FIRST and DISPLAY-ONLY Keywords 204
Selecting Members That Have Passed or Need Integrity Checks. 205
Selecting Members Based on a Specific Member Type 205
PERFORM Examples 205
PERFORM Syntax 207

PRINT (CMR-CM01) 208
How to Print a Member. 208
Printing Specific Lines of a Member 210
Printing Members in a Status Environment 211
Printing a Member's Source Record as it was at the Time it was Last Successfully Encoded 211
Print Output. 212
PRINT Syntax 213

PROTECT (CMR-DD3) 213
Protecting Individual Members of the Dictionary 214
Assigning Ownership to Members 215
PROTECT Syntax. 216

QUERY KEPT-DATA (CMR-CM01). 217
QUERY KEPT-DATA Syntax 218

QUERY MODE 218

READ and WRITE (CMR-FE01, CMR-TP7, CMR-TP8) 218
How to Read From, and Write To, an MVS Dataset 218
READ and WRITE Syntax 220
How to Read From, and Write To, CMS files 221
READ and WRITE Syntax 221

REINSTATE (CMR-CM01) 222
REINSTATE Syntax. 223

REMOVE (CMR-CM01) 223
REMOVE Syntax 224

RENAME (DMR-DD1). 224
Renaming an Existing Member. 224
How to Rename a Member Directly Used By Another Member 225
RENAME Syntax 226

REPLACE (CMR-CM01) 226
How to Replace the Source and Encoded Record of a Member. 227
How to Number Lines in a Member's Source Record 229
REPLACE Syntax. 230

REPORT 230
Reporting One or a List of Members 230
Reporting a Member in a Status Environment 231
Reporting a Member and the Members That it Uses Via an EA Relationship Type CONTAINS. 232
Reporting a Member and Listing the Members That it Uses Via Any Relationship Type. 232
Inserting a Blank Line or Page Break Between Reports. 233
Output for an Encoded Member 233
Output for a Dummy Member. 234
REPORT Examples. 235
REPORT Syntax 236

RFILE (CMR-FE01). 236
RFILE Syntax 237

SHOW ALIAS-TYPES (CMR-CM01) 237
SHOW ALIAS-TYPES Syntax. 238

SHOW MEMBER-TYPE (CMR-CM01) 238
The SHOW MEMBER-TYPE Command 238
Output from the SHOW MEMBER-TYPE Command. 239
SHOW MEMBER-TYPE Syntax 241

SHOW PRODUCE-OPTIONS (CMR-CM01). 242
SHOW PRODUCE-OPTIONS Syntax. 243

SHOW UDS (CMR-CM01) 244
Introduction to the SHOW UDS Command 244
Displaying the Full UDS Table. 244
Displaying Member Types 244
Displaying User-Defined Attributes Types. 246
Displaying ER Relationship Types 247
Displaying UDS Tables Other Than the Repository’s Current One. 247
SHOW UDS Syntax 247

SOURCE-FILE (CMR-FE01) 249
SOURCE-FILE Syntax. 250

STORE KEPT-DATA (DMR-DD1) 250
STORE KEPT-DATA Syntax. 251

UPDATE (CMR-FE01). 251
UPDATE Syntax. 252

VALIDATE (MMR-PF61) 252
VALIDATE Syntax 253

WBDA 253

WHAT (DMR-DD1) 253
Introduction to the WHAT Command 253

Selecting Encoded Members that Use a Given Member 254
Selecting Members Used By a Specified Encoded Member 255
Selecting Only Members in Your Status Window for Selection 256
Showing Relationships 256
Selecting Members by Catalog Classification 257
Combining Catalog Selection with Further Conditions 257
Selecting Members that Belong to a Specified Owner 258
Finding Out What an Index-Name Is 259
WHAT Examples 259
WHAT Syntax. 260

WHICH (DMR-DD1) 265
Introduction to the WHICH Command. 265
WHICH Command Selection 267
Selecting Members That Use a Specified Member. 269
Selecting Members That are Used By a Specified Member 271
Selecting Members That Use a Specified Member Via a Specified Relationship Type or Relationship Class. 273
Selecting Index-Names That are Used by a Specified Member Via a Specified Relationship Type or Relationship Class 275
Selecting Members by Catalog Classification 277
Selecting Members With a Given Clause or Attribute 277
Selecting Members Having an Attribute Containing a Specific Value. 278
Selecting Members Belonging to a Specified Owner 280
Selecting Members by a Usage or Reference Count. 281
Selecting Index-Names Kept in a KEPT-DATA List 281
Selecting Index-Names That Do or Do Not Meet a Combination of Conditions 282
Name, Status, Time and User-Related-Selections 283
WHICH Examples 283
WHICH Syntax 285

WHO (CMR-DD3) 290
WHO Syntax 290

WHOSE (DMR-DD1) 291
WHOSE Syntax 292

WRITE 292

Applying Commands to a Selection Of Index-names Based on Name, Status, Time and User 293
NAME-RELATED-SELECTION 294
Status-Related-Selection 299
TIME and USER-RELATED-SELECTION (CMR-CM01) 308

Collective Member Types 316

Definition Of Index-names 316

5 Dictionary Contents - Member Definition 317

Manager Products Member Types 317

Member Types Provided By Manager Products 318
Member Types which Document Data and the Processes that Act Upon it 320

Member Definition Statements 322
ITEM Member Definition Statement 322
GROUP Member Type 351
File Member Type. 357
FILE Member Definition 368
SYSTEM, PROGRAM and MODULE Member Definition 380
COMMAND-STREAM Member Definition 387
OBSOLETE-DEFINITION Member Type. 389

Common Clauses. 395
Common Clause Definitions. 395

6 Selectable Units for Use in Dictionary Management 403

Interfaces to Database Management Systems 404

Source Language Generation. 406

Interfaces to File Management Systems 407

Manager Product Integrator 407

The Status Facilities 407

Audit and Security 408

User Defined Syntax 409

Index. 411

Preface

This *ASG-Manager Products Dictionary/Repository User's Guide* is one of a series describing the Manager Family of Products for use on IBM System/370 Architecture and plug compatible computers.

This publication describes the Dictionary Management Software provided by ASG-ControlManager (herein called ControlManager) and ASG-DataManager (herein called DataManager). The capabilities provided by each of these Manager Products has been indicated in the descriptions of the individual commands and member types.

Within the Manager Family of Program Products (excluding ASG-MethodManager), ControlManager, and ASG-DictionaryManager (herein called DictionaryManager) are Co-Requisites of each other. Both are Environmental Prerequisites (EPR) in as much as they must be at the latest Version and Release Level for each and every other Manager Product to execute correctly. This EPR rule applies to Manager Products in both Mainframe Environments (MFE) and Programmable Workstation Environments (PWSE). ControlManager and DictionaryManager complement each other in providing a gateway environment to OSI (Open Systems Interconnection) across information engineering techniques and dictionaries/directories/repositories from ASG and other vendors. Thus, ControlManager and DictionaryManager enable Manager Products users to position themselves to take full advantage of the Manager Family in providing a Computer Aided Software Engineering (CASE) environment.

For Mainframe Environment (MFE)-based Manager Products, ASG provides Maintenance and Updating Service in IBM OS, DOS, VM/CMS and BS2000 environments; where the Version/Release of OS, DOS, VM/CMS or BS2000 in use is defined in an Appendix of the appropriate Manager Products installation publication. (DOS and BS2000 support is *not* currently available for ASG-MethodManager.) Particular Manager Products selectable units interface with certain IBM and/or Siemens program products that run within OS and/or DOS and/or VM/CMS and/or BS2000 environments and/or with other vendor program products: the Compatible Version/Release Levels at which these products interface with Manager Products is documented in an Appendix of the appropriate Manager Products Installation publication. Throughout ASG's Manager Products publications, the terms OS, DOS and VM/CMS respectively cover all those variants of OS and of DOS and of VM/CMS for which ASG has defined a Compatible Release Level.

The workstations currently supported by Manager Products are defined in an Appendix of the installation publications referenced above.

For Programmable Workstation Environment (PWSE)-based Manager Products, the Version/Release of Windows in use is defined in *ASG-ManagerView User's Installation and Environmental Tailoring*.

The hardware devices currently supported by the Manager Products Programmable Workstation Family are defined in *ASG-ManagerView User's Installation and Environmental Tailoring*. The original concept of the data dictionary, on which the dictionary element of the Manger Family of Program Products was based, has expanded significantly since its introduction. Different vendors have concentrated on different aspects of the functionality now attached to the data dictionary concept. These vendors have introduced different terms to differentiate their dictionary products from those of other vendors. Thus there is now available in the marketplace a range of different vendor products, using terms such as catalog, directory, encyclopedia, knowledge base, lexicon or repository. ASG's Manager Products have consistently remained in the forefront of this expanding technology, providing the full range of functionality in a Corporate Dictionary/Repository which can peacefully coexist with the more specialized and focused dictionary products of other vendors. Throughout ASG's Manager Products publications, for reasons of brevity, reference is made to the Manager Products Corporate Dictionary/Repository as the "dictionary." ASG has chosen to use dictionary as still most aptly describing the broad functionality ASG-Manager Products provide, and the environment which the Manager Family creates.

This publication is intended to describe a Corporate Dictionary/Repository and how it is used.

ASG welcomes your comments, as a preferred or prospective customer, on this publication or on ASG-Manager Products (herein called Manager Products).

About this Publication

The *ASG-Manager Products Dictionary/Repository User's Guide* consists of these chapters:

- Chapter 1, "The Corporate Dictionary/Repository," introduces the Corporate Dictionary/Repository, its benefits, uses and basic concepts.
- Chapter 2, "What Do You Want To Do?," asks you what you want to do and directs you to the relevant documentation.
- Chapter 3, "Using the Dictionary," overviews dictionary management commands grouped together according to the type of task they perform.
- Chapter 4, "Dictionary Management Commands," contains the syntax and a detailed explanation of the use of dictionary management commands.
- Chapter 5, "Dictionary Contents - Member Definition," contains an introduction to the member types provided by Manager Products. It gives a detailed explanation and the syntax of the member types useful in dictionary management and those, with the exception of the Database Management System member types, provided for the purpose of documenting data and the processes that act upon it.
- Chapter 6, "Selectable Units for Use in Dictionary Management," overviews the selectable units provided by ASG-DataManager and those provided by ASG-ControlManager that are of use in dictionary management.

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 is continued 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.

Convention	Represents

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──┘
```

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; one at least must be specified:

Convention	Represents

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

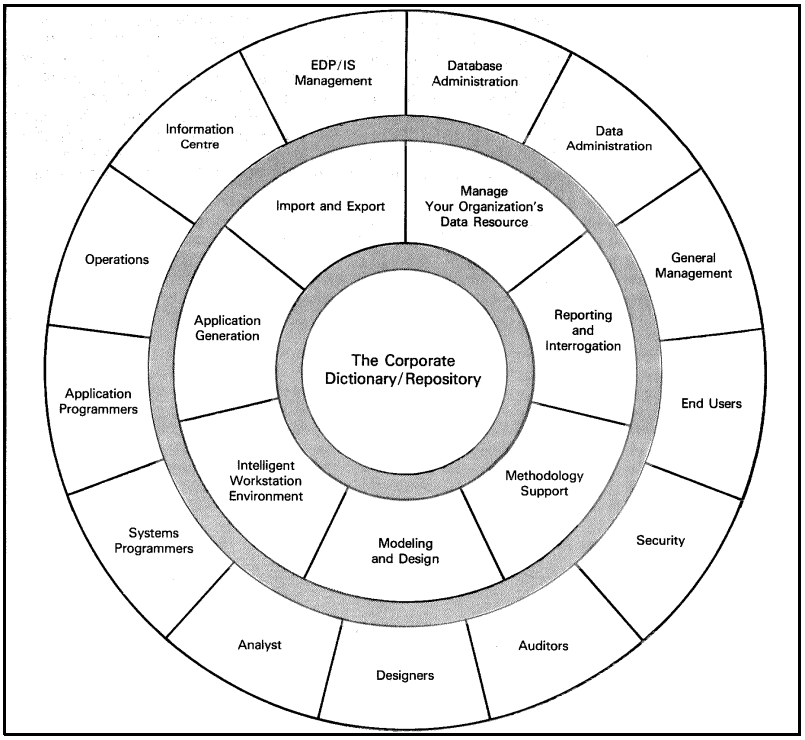
The repeat symbol above a list of keywords (all of which are below the main path) indicates that any one or more of the keywords maybe specified, but they are all optional:

```
                          <<<<<<<<<<<<<<<<
➤──────COMMAND──────────┬──KEYWORD1──┬──────────────────────────➤
                        ├──KEYWORD2──┤
```

Occurrences of syntax diagrams, other than those in Chapter 4, follow the syntax conventions described here:

- All words printed in capitals are statement identifiers or keywords that must be present in full or truncated form in the circumstances stated in the statement specification. The extent beyond which a word must not be truncated (because it would become ambiguous) is indicated by underlining of the characters that must be retained. Truncation limits for statement identifier keywords are not shown because they are dependent on the Manager Products configuration installed and could be affected by the presence of User Defined Commands or User Defined Syntax. The truncation limits for selection keywords in commands could also be affected by your Manager Products configuration and by User Defined Syntax.
- All words printed in lower case are variables for which the user must substitute a value consistent with the specification
- Material enclosed in square brackets [ ] or pointed brackets < > is an option which may be included or omitted as required
- Braces { } indicate that a choice must be made of one of the options enclosed within them
- Three periods or full stops . . . indicates that the material they immediately follow may be repeated. Where . . . immediately follows a closing square or pointed bracket or brace, the material that can be repeated is bounded by that square or pointed bracket or brace and the corresponding opening square or pointed bracket or brace. If material can be repeated only a limited number of times the repetition permitted is stated in the specification.
- Other punctuation marks and symbols must be coded as shown, subject to the implications of any square or pointed brackets or braces enclosing them; except that where a single quote, 1, is shown, a double quote, “, can alternatively be used, provided that the opening and closing quotes of any pair of quotes are the same character (single quote or double quote). Quote characters are used in syntax as delimiters, marking the beginning and end of a string of characters. Your Systems Administrator may have specified an additional character that may be used as an alternative delimiter character. You can find out if this has been done by entering the command: `QUERY STRING-DELIMITER;`.

Figure 1 • The Manager Products Corporate Dictionary/Repository: Users and Uses



Allen Systems Group, Inc. uses these conventions in technical 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 The Corporate Dictionary/Repository

This chapter includes these sections:

What is a Dictionary/Repository? 2
What is a Dictionary/Repository Used for? 3

What is a Corporate Dictionary/Repository? 4

Dictionary Management Software 5
Basic Dictionary Management 5
Extended Dictionary Management 5
Additional Dictionary Management Facilities 5
CASE and the Corporate Dictionary/Repository 6

Basic Concepts 9
Dictionary Member Types 9
Common Clauses 10
Dictionary Member Records 10
Physical Structure of a Dictionary/Repository 14
The Dictionary Controller 16
The Security System 17
The Automatic Recovery System 22

The Manager Family of Products provide Computer Aided Software Engineering (CASE) tools. These tools are designed to reduce the amount of time and resource required to build and maintain information systems and to improve the quality of those systems.

Manager Products enable you to document any system and any information, for example design or administrative information, about that system. They enable you to use that documentation to improve the development and maintenance of systems, by automation of the tasks involved: from planning and analysis through to physical implementation.

The documentation is held in a Corporate Dictionary/Repository. The tools which address the automation of the various stages of systems development and maintenance (sometimes referred to as the Systems Development Life Cycle) are applied to the information held in the Corporate Dictionary/Repository.

Note:

You can execute Manager Products in either a client/server (MPSF) mode or a non-client/server (standard) mode. This publication assumes the use of standard mode and documents only minor differences between the two modes. When executing under MPSF you can create repositories and MPAIDs using the Data-in-Virtual (DIV) access method. The DIV access method and significant MPSF installation and usage differences are documented in the *ASG-Manager Products Server Facility User's Guide*.

What is a Dictionary/Repository?

A dictionary or repository is an organized collection of definitions. A definition describes something. In the context of Manager Products, you document each component of a system and its relationship with the other components, in a definition which is stored in the dictionary/repository.

For example, you might document the *types of data* (such as names, addresses and account numbers, etc.) that are used by your organization. And you might document the processes (programs, modules, etc.) that operate on that data.

The documentation held in the dictionary is about the components of the system; it is not the real system. For example, a dictionary member called ACCOUNT- NUMBER would not actually contain an account number. It would contain information about the structure and use of account numbers: their length, the files in which they are held, etc.

The documentation is entered into and held in a central dictionary/repository on your organization's computer system. Once it is entered into the dictionary/repository, the documentation may be interrogated and manipulated in ways which enable you and your organization to improve the development and maintenance of the documented system. Throughout Manager Products documentation this central store of information is referred to as the *Corporate Dictionary/Repository* (or simply *dictionary*).

What is a Dictionary/Repository Used for?

Complete and reliable documentation is essential to the efficient management of any system.

Traditionally data dictionaries have been used in the data processing industry to manage computerized information systems such as database and conventional file systems. To manage such systems it is essential to know:

- Exactly what data is held
- Where it is held
- How it is defined
- How it is organized or structured
- The different forms in which it should appear for different purposes
- Possibly the different names under which it is known in different contexts
- How it relates to other data
- What programs it is used by

Organizations would document the data and the processes used in their computer systems in order to be able to manage, develop and maintain those systems efficiently.

Today, the traditional use of data dictionaries has been extended to include, for example, documentation about the organization that uses the information system, its business functions, plans and goals.

The information recorded in the Corporate Dictionary/Repository is manipulated, using the facilities provided by Manager Products, for these purposes:

- Planning of information system requirements/developments (business and design information is recorded in the dictionary)
- Analysis of existing systems (it is a simple matter to interrogate the dictionary to find out which parts of the system will be affected by a proposed development)
- Design of new systems
- Database implementation (generation of components direct from the dictionary/repository)
- Automatic construction of systems components (programs, screens etc.)
- Maintenance of the existing systems (you use the documentation to help resolve problems).

However, the dictionary may also be used in other contexts. For example, a state department (in the United States) has used a Manager Products dictionary to help manage a road system. And an oil company uses a Manager Products dictionary to hold information about its oil fields.

What is a Corporate Dictionary/Repository?

Manager Products provide a free-standing dictionary. It is not dependent upon any other system (other than the operating systems described in the Preface) and it has the ability to communicate with the dictionaries, directories, repositories, encyclopedias and catalogs provided by other vendors.

For example many database management systems (DBMSs) are supplied with, and are dependent upon, their own dictionaries/directories. Such directories may be necessary for the use of the DBMS with which they are supplied but (usually) they may be unsuitable for use with other DBMSs. A Manager Products dictionary can be used to document any DBMS and provides pre-defined interfaces to most of the major DBMSs available on IBM and Siemens mainframes.

If you use several DBMSs and associated dictionaries/directories you can use a Manager Products dictionary to provide corporate documentation that is independent of a particular application. For example, if you use IMS and DB2 you can:

- Document IMS in the dictionary
- Export complete definitions from the dictionary to the DB2 catalog

and thus maintain consistent documentation, centrally. This arrangement has these advantages:

- Data that is shared between the two systems is documented once only
- The dictionary becomes a better tool for analysts and developers because it provides information about both systems.

The Manager Products dictionary allows the import of information held in the dictionaries/directories associated with other vendors dictionary/directories. So, if your organization exploits several DBMSs, you can document them, together with any other pertinent information which is independent of a particular DBMS, in one place and in one format.

The dictionary can be used to manage systems and data that are distributed across several different locations. International companies, for example, may have computer installations in different countries and the dictionary can provide a centralized source of information about the physical location of data.

So, the Manager Products dictionary is a Corporate Dictionary/Repository because it may be used to document business plans and systems at a corporate rather than a local level. It enables centralized control across projects/systems and corporate-wide access to consistent information.

Dictionary Management Software

Basic Dictionary Management

Basic Dictionary Management Software (BDMS) is provided to all users of Manager Products. BDMS is the software necessary to create, maintain, and use a Manager Products Corporate Dictionary/Repository.

BDMS is supplied as part the ControlManager Nucleus which is a prerequisite for the installation of all other Manager Products. The ControlManager Nucleus also provides:

- The software necessary to install and use Manager Products at your installation
- A friendly and easy-to-use interface between you and Manager Products.

Basic Dictionary Management Software and how to use it is documented in this publication.

Extended Dictionary Management

Extended Dictionary Management Software (XDMS) provides powerful interrogation capabilities with which to extract information from the dictionary. In particular, XDMS provides the commands with which you can obtain information about the relationships between different components of a system.

Extended Dictionary Management is provided by the DataManager Nucleus.

A list of the commands provided by DataManager is included in Chapter 3, "Using the Dictionary," on page 29.

Extended Dictionary Management Software and how to use it is documented in this manual.

Additional Dictionary Management Facilities

Each Manager Product comprises the basic program (the software nucleus) and additional modules (the software *selectable units*) which provide additional facilities.

Thus a number of optional additional dictionary management facilities are available. Your organization can choose just those facilities for which it has a use. The available facilities include:

- Interfaces to database management systems (DBMSs) such as IBM's DB2, SQL/DS, IMS, IDMS and others
- Generation of COBOL, Assembler and PL/1 source language data descriptions
- Generation of Manager Products dictionary definitions from previously available COBOL or PL/1 sources (the Automation of Set Up facility)
- The ability to document dictionary definitions in various statuses, for example, superseded, current and test statuses, or statuses for different development areas.
- A security system whereby information held in the dictionary can be protected against unauthorized access, alteration or removal. The security system is part of the Audit and Security facility which also provides various audit reports based on information obtained from the dictionary's log.

  Refer to "The Security System" on page 17 for a description of the security system.
- The ability to extend the types of object and associated characteristics that you may document in the dictionary (user defined syntax).

As mentioned above, a description of the security system is included in this manual. The rest of the above facilities are described briefly in Chapter 6, "Selectable Units for Use in Dictionary Management," on page 403, which also provides references to the publications in which they are fully documented.

Dictionary Management Software (both Basic and Extended) and how you use it is the subject of this publication.

CASE and the Corporate Dictionary/Repository

Operating on the information stored in your Corporate Dictionary/Repository, Manager Products offer an integrated set of CASE tools.

Manager Products are based upon the Corporate Dictionary/Repository: they exploit your investment in the information recorded in it. The following paragraphs describe the purpose of each Manager Product and it how it uses the Corporate Dictionary/Repository.

An interactive data and enterprise modeling system for logical and physical database design is provided by DesignManager. It uses a logical subset of your Corporate Dictionary/Repository known as the Modeling Dictionary. The Modeling Dictionary is used to store details of a model of the data used by your organization (that is an entity model) and data access requirements specified by end users (userviews), so that first cut logical and physical database design models, which fulfill these requirements, can be produced automatically.

Both top-down entity modeling and bottom-up userview modeling may be employed as input to DesignManager's powerful data normalization capabilities. After normalization, DesignManager can generate logical and first-cut physical database design for DB2, SQL/DS and IMS.

These benefits are to be gained from this use of the dictionary:

- It helps designers and users achieve a common understanding about their organization's data resource
- Business users get the system they require
- Information system requirements that satisfy your organization's business plans are identified and developed
- Maximized operational efficiency and reduced maintenance overhead due to effective data structures
- Repeated extension and amendment of a database design is made simple and reliable.

An Intelligent Workstation environment in which you can use diagrams to input/output information to/from the dictionary/repository is provided by ManagerView. ManagerView's interactive diagramming features and Close Host Connectivity with the mainframe computer make it the perfect CASE environment in which to view and update the Corporate Dictionary/Repository.

ManagerView's graphic capabilities are very flexible and can easily be tailored to suit your development methodology.

These are the benefits to be gained from this use of the dictionary:

- Improved human interface with the dictionary/repository including Windows, Mouse, color and Interactive Graphics
- Diagrams promote better communication with end users
- The friendliness of personal workstations can be exploited and coupled with the power of the Corporate Dictionary/Repository.

A dictionary/directory interchange system is provided by DictionaryManager. It provides a user-definable system that imports and exports definitions between your Manager Products Corporate Dictionary/Repository and the (variously named) dictionaries/directories/ catalogs/repositories/encyclopedias of other vendors' products.

A benefit to be gained from using the dictionary is, maximized return on investment in your Corporate Dictionary/Repository by using it to centralize management of your data and information resource across all the computer environments (hardware and software) used by your organization.

A comprehensive Computer Aided Information Engineering tool and associated systems development methodology is provided by MethodManager. MethodManager exploits and extends dictionary/repository technology into an Information Engineering Knowledge Base. MethodManager's methodology for Strategic Information Planning and for Application Systems Development is based upon the best of proven systems development methodologies. And MethodManager exploits the intelligent workstation environment to enhance its capabilities and user-friendliness.

These are the benefits to be gained from this use of the dictionary:

- Comprehensive CASE solution
- Maximizes usage of human resource
- Introduces standardized proven methods
- Improved productivity through the introduction of Engineering type disciplines into the business of Software Analysis, Design and Development.

So, the Manager Family provides complete and integrated support for the Systems Development Life Cycle as follows:

- Corporate Dictionary/Repository (ControlManager and DataManager)
- Modeling and design (DesignManager)
- Intelligent workstation environment (ManagerView)
- Application generation
- Copy management (DictionaryManager)
- Automated Information Engineering methodology (MethodManager).

Manager Products are based upon use of the Corporate Dictionary/Repository.

The following sections describe concepts with which it is useful to be familiar before using your Corporate Dictionary/Repository.

Basic Concepts

Dictionary Member Types

Any definition of which a record exists in the dictionary is termed a *member* of that dictionary.

In order to prevent several definitions, each relating to a different component of a system being assigned the same name, the dictionary management software rejects the addition of a new member with a name that already exists. However, if you have one of the Status facilities (Basic Status or Advanced Status) you may define several definitions of the same member (a test and a production version for example) in a controlled manner.

Many of the members that you wish to add to your dictionary/repository are similar. For example, you might want to document several computer programs. Because computer programs have certain characteristics in common, it becomes useful to establish a *member type* called PROGRAM and to assign that member type to all such members.

Members of the same type have particular attributes/characteristics in common. All programs are likely to use files of data. And so it is useful for the PROGRAM member type to support INPUTS and OUTPUTS clauses (in which the names of the files used by the program may be specified) and other predefined clauses suited to the purpose of describing the characteristics common to programs. Use of such predefined clauses also ensures that the information is entered in a consistent manner by the dictionary user.

The situation is similar for relationships between members. Programs use files. The documentation of files in the dictionary will be supported by the member type FILE. In this context it is useful to control the type of relationship established between PROGRAMs and FILEs: PROGRAMS can input FILES, but FILES cannot input PROGRAMs, therefore the FILE member type should not support an INPUTs clause. Such control helps toward meaningful and reliable interrogation of the dictionary/repository: you can query which programs use a particular file via INPUTs without the possible confusion that in some instances the relationship has been defined in a different way.

So, the use of member types creates a dictionary structure (sometimes referred to as the dictionary schema) that can reflect the system that is being documented. This structure makes the dictionary/repository meaningful and easy to use: it becomes a model of the real system.

Refer to Chapter 5, "Dictionary Contents - Member Definition," on page 317 for details about member types.

Common Clauses

While member types provide clauses which reflect the characteristics common to particular types of member, several clauses are available for use in the definition of all members. They include an ALIAS clause which enables you to specify alternative names for a member and a CATALOG clause which enables you to classify a member in ways that you find useful and that are not catered for by the other clauses.

Refer to "Common Clauses" on page 395 for details of common clauses.

Dictionary Member Records

Dictionary members may have three different types of records:

- An index record
- A source record
- An encoded record

The dictionary member records are held in the Index dataset, Source dataset, and Data entries dataset.

The index, source and data entries datasets and physical structure of a Manager Products dictionary are described in "Physical Structure of a Dictionary/Repository" on page 14.

The commands which enable you to add definitions to the dictionary require specification of a name for the definition. That name is recorded in the index dataset and the definition is recorded in the source dataset. An encoded record is (normally) generated automatically when you add a definition to the dictionary. An encoded record is a condensed and reorganized form of a definition; it contains information about a member's relationships with other members and it enables the dictionary management software to respond efficiently when you use those commands provided for intelligent interrogation of the dictionary.

Definitions are written using Manager Products definition language.

For details of Manager Products languages refer to the *ASG-ControlManager User's Guide*.

Index Records

Each dictionary member has an index record in the index dataset (refer to "Physical Structure of a Dictionary/Repository" on page 14) to record its name and information, for use by the dictionary management software, about the physical location of the members encoded record.

Source Records

A member's source is its definition as entered into the dictionary. The source record is documentation only; it cannot be reported or interrogated. You can of course update a source record and you can print its contents using the PRINT command.

The definition contained in a source record differs from the definition you input, in two respects:

- The terminator is not included in the source record
- In the source record, the input lines are numbered. The line numbers are determined by the command(s) by which the source record was created and/or amended.

Source records are held in the source dataset (refer to "Physical Structure of a Dictionary/Repository" on page 14).

Encoded Records

An *encoded record* is a special form of a member definition which is processed when you use commands that interrogate and generate reports from the dictionary.

An encoded record can be generated from a source record either at the same time as the source record is added to the dictionary or subsequently, depending upon the command you use. Encoded records are held the in data entries dataset (refer to "Physical Structure of a Dictionary/Repository" on page 14).

The syntax of the member definition is validated when a member is encoded. The definition must conform to the syntactical requirements of the dictionary structure (refer to "Dictionary Member Types" on page 9).

An encoded record contains two *usage-tables*:

- A *used-by* table
- A *reference* table

to contain information about that members relationships with other members (if it has any).

A *reference table* contains pointers (*reference pointers*) to the members that the encoded member refers to. For example, if a relationship with a member called FILE-A is specified in the source record of PROGRAM-A, a reference pointer to FILE-A is set up in the reference table of PROGRAM-A, when the source record is successfully encoded.

The dictionary supports many different types of relationship. For example, one member may INPUT another, one member OUTPUT another and one file CONTAIN another file. Manager Products documentation uses the generic phrase refers to when discussing a relationship whereby a member has at least one such type of relationship with another.

Whenever a reference pointer is set up a corresponding used-by pointer is also set up.

A *used-by table* contains pointers (*used-by pointers*) to the members which use the encoded member. For example, if a relationship with a member called FILE-A is specified in the source record of PROGRAM-A, a used-by pointer to PROGRAM-A is set up in the used-by table of FILE-A, when the source record (of PROGRAM-A) is successfully encoded.

So, relationships between members are two-way:

```
NAME -----> refers to >-----> MEM-2
      <-----< is used by <-----
```

This arrangement enables the dictionary management software to respond to your interrogation commands faster than it could if relationships were only one-way. For example, if you want to list all the members that refer to a particular member, it is not necessary for the software to search the reference table of every member; it need only look at that member's used-by table.

If a definition (source record) includes aliases, catalog classifications or (see "User Defined Syntax" on page 409 if the User Defined Syntax facility is installed) user defined indexed attributes, separate encoded records are generated for each and added to the data entries dataset (and to the index dataset) when the definition is encoded.

If the encoded definition contains aliases and catalog classifications, separate data entries records are generated for each. If the encoded definition contains user defined indexed attributes (refer to "User Defined Syntax" on page 409 when the User Defined Syntax facility is installed) separate data entries records are generated for each.

When an encoded record is to be amended, the changes are first made to the source record and that source record is subsequently re-encoded.

Dummy Members

Encoded records can be generated in two ways:

- By encoding a member's source record
- By encoding the source record of a member that includes the specification of a relationship between itself and a member name which does not exist in the dictionary.

In the latter case both an index and an encoded record are generated automatically for the specified name although no definition (of that member) as such has been added to the dictionary. Such a record is known as a dummy member.

The creation of dummy members which are not fully defined in the dictionary is a significant feature. It enables a dictionary to become useful as a model of your system before you have documented every detail of the system and thereby allows you to adopt a top-down approach to documenting the system.

Usage Tables

The Manager Products Status facilities (Basic Status and Advanced Status) enable you to record several versions of the same member. For example, you might record a test and a production version of the same member. The dictionary is divided into several statuses and you may record several different definitions of the same member: one in each status.

When a member in a status uses another member, which has no definition, the dictionary management software creates a usage table for the undefined member. If the undefined member does not have an encoded definition in a visible status, it will appear as a dummy member. If the undefined member has an encoded definition in a visible status it will appear as a usage table.

Thus a usage table is similar (in concept) to a dummy member: it is created to contain information about a member's relationships with other members. However, usage tables will only be created when the dictionary contains several statuses. Dummy members may be created regardless of the dictionary containing statuses.

For information about statuses refer to *ASG-Manager Products Status Concepts* or InfoBank panel STATUS1000.

Unverified Members

Members with unencoded source records are referred to as *unverified* members.

Unencoded source records are created in these circumstances:

- When you add a definition to the dictionary using one of the commands provided specifically to enable you to prevent an encoded record being generated
- When you update a member, which already has an encoded record using one of the commands provided specifically to enable you to prevent an encoded record being generated from the source record
- When you update a member, which already has an encoded record, using one of the commands that will automatically attempt to generate an encoded record from the updated source record and the encoding process fails (because, for example, you have specified a relationship that is invalid for the member type)
- When the source records of dictionary members are moved into a dictionary (after being saved from another dictionary)

In all of the above cases a source record is created but not encoded; the source record is *unencoded*. Members in any of the conditions described above are referred to as *unverified* members because the source and the encoded records may not be in agreement.

Dummy members may also be regarded as unverified if they have a source but not an encoded record.

You can list unverified members selectively using the LIST command. Output from the command includes a CONDITION column that indicates if the member has an encoded record and/or a source record. Unverified members and dummy members are flagged with an asterisk (*).

Note:

When a member is unverified you may encounter contradictions between what you see when you look at the source record and the output you get from interrogation or reporting commands (which obtain their information from the encoded record) involving that member. Refer to Chapter 3, "Using the Dictionary," on page 29 for an overview of the commands used to add, update and encode dictionary members.

Summary of Dictionary Member Types

A member can have an index record, and a source record, and/or a data entries record. An encoded record can contain an encoded form of the definition held in the source record, Reference pointers, and Used by pointers. A dummy record contains Used by pointers only.

Physical Structure of a Dictionary/Repository

A Manager Products dictionary/repository comprises four or five datasets (file):

- The source dataset
- The data entries dataset
- The index dataset
- The error recovery dataset
- If logging is applied to the dictionary, a log dataset

The *index dataset* contains the name and the address of each member in the dictionary. The index is designed to achieve the fastest possible retrieval of source or encoded data.

The *source dataset* contains the definition as input or as subsequently amended. However, in order to save storage space, any consecutive space characters in a definition are held in compressed form and are expanded to the input format when you display or process the source record. The exception is any definition containing characters below X'40'; these lines are not compressed. This is important in installations using double-byte character sets, such as Kanji.

The *data entries dataset* contains encoded definitions generated by the dictionary management software from the definition in the source dataset. An encoded record is a condensed and reorganized form of a definition; it contains information about a member's relationships with other members and it enables the dictionary management software to respond efficiently when you use those commands provided for intelligent interrogation of the dictionary.

If a definition (source record) includes aliases, catalog classifications or (if the User Defined Syntax facility is installed—see "User Defined Syntax" on page 409) each user defined indexed attributes, separate encoded records are generated for each and added to the data entries dataset (and to the index dataset) when the definition is encoded.

The *error recovery dataset* is used as a temporary back-up file for the dictionary/repository's automatic error recovery system. The automatic error recovery systems is described in section "The Automatic Recovery System" on page 21.

If logging is applied to a dictionary, a log dataset is established as a fifth dataset of that dictionary. All updating commands, together with any associated member definitions or amendments, are then logged in that dataset as they occur, together with full date, time, user, status and physical input/output accesses information. The dictionary Controller has the option additionally to log all other commands issued in respect of the dictionary.

Occurrences of messages may also be logged selectively; this allows the dictionary Controller to log the occurrence of particular events/circumstances in the dictionary. For example, the logging of messages which are output in response to attempts to access the dictionary by unauthorized users could help a dictionary Controller maintain dictionary security.

The Dictionary Controller

The person directly responsible for the administration of a dictionary is referred to (throughout Manager Products documentation) as the dictionary Controller. Typically, the dictionary Controller will report to the person in your organization who is responsible for data administration; in some organizations both jobs might belong to the same person.

It is recommended that an organization maintain only two dictionaries:

- A Corporate Dictionary/Repository used to document the organization, its systems and related information
- A Manager Products Administration Dictionary for use in running Manager Products

The Manager Products Administration Dictionary should be the private resource of the Systems Administrator. The Systems Administrator is the person with overall responsibility for the installation and operation of Manager Products at an installation. The role of the Systems Administrator and the facilities provided to support that role are described in the *ASG-Manager Products Systems Administrator's Manual*.

No attempt is made here to lay down a job description for the dictionary Controller, but certain tasks must be kept within the Controller's personal province if a Corporate Dictionary/Repository is to be used with full success. The tasks are:

- The creation of the dictionary
- The reorganization of the dictionary when this becomes necessary
- Securing the dictionary against corruption or loss by maintaining back-up copies
- Securing the dictionary against unauthorized access or manipulation
- The transfer of information between dictionaries, if more than one dictionary is maintained

The commands available to the Controller for the fulfillment of these tasks are documented in *ASG-Manager Products Controller's Manual*. The circulation of *ASG-Manager Products Controller's Manual* is restricted to dictionary Controllers because it contains information about commands which are available to dictionary Controllers only.

Some of the tasks within the Controller's province involve the administration of the dictionary's physical datasets (files). These tasks can be delegated by the Controller to a Master Operator, under the security of a special password. The Master Operator is given authorization by the Controller to carry out these tasks for specific dictionaries. The authority given by the Controller enables that Master Operator to perform a limited subset of the commands available to the Controller, but not to access the information recorded in a dictionary. The Master Operator is therefore not regarded as a user of a dictionary.

Although the commands involved in securing the dictionary against unauthorized use should be known only to the Controller, it is desirable that other users should have a general knowledge of how the security of the dictionary is ensured.

The Security System

There are two aspects to security in the context of the dictionary/repository:

- Protecting the entire dictionary system against unauthorized access
- Controlling access to confidential information held in the dictionary

Facilities for restricting access to the dictionary are provided by the Basic Dictionary Management Software. Facilities for establishing owners of information in the dictionary and for controlling access to that information are provided by the optional additional facility called the Audit and Security facility (selectable unit CMR-DD3).

The full specifications of any command mentioned in this description of the security system are in Chapter 4, "Dictionary Management Commands," on page 47. However, actions mentioned as performed by the dictionary Controller involve private commands that are not available to other users; those commands are documented in the *ASG-Manager Products Controller's Manual*.

Access to the Dictionary

The name of each user who is authorized to access the dictionary is registered in the dictionary by the Controller. For each user, the Controller registers an individual password.

When you want to open a dictionary, you identify and open it using a DICTIONARY command. No commands for processing the opened dictionary are accepted until you enter an AUTHORITY command, specifying the password that the Controller has registered in the dictionary for your use, and the command is executed successfully.

Once a valid AUTHORITY command is received, then, subject to the security conditions discussed later in this section, subsequent commands are accepted as being issued by the identified user, until a further AUTHORITY command is received.

Access to Members of the Dictionary

The dictionary/repository is used to document a system: the information held in the dictionary is about the components of the system, it is not the *real* system.

For example, a dictionary member called ACCOUNT-NUMBER would not actually contain an account number. It would contain information about the structure and use of account numbers: their length, the files in which they are held etc.

The information in the dictionary is used by the people involved in the management, maintenance and development of the system rather than the people who actually use the system. It is important to keep a clear distinction between users of the dictionary/repository and users of the real system that is documented in the dictionary/repository.

Users

In the context of Manager Products documentation *user* means dictionary user. In other contexts user may be the term used by the people using Manager Products dictionary/repositories to refer to users of the real system (known, in some environments, as end users).

However, the users/departments that use the real system will have an interest in and are ultimately responsible for ensuring that the information in the dictionary reflects reality and their requirements. And, considerations of confidentiality or security sometimes make it necessary for departments to restrict access to the details of their system, in particular details about the data they use, as much as they would the system/data itself. For example, it might be considered important that the exact contents and layout of a payroll file is not generally known.

If the dictionary is expected to be reliable (used to support production systems perhaps) it may also be necessary to protect the information from unauthorized update.

Owners

Any definition in the dictionary/repository can have an owner ascribed to it. Typically an owner will be a department within the organization, but it might also be an individual systems analyst or a software development team. The dictionary Controller is responsible for establishing owners in the dictionary.

Once an owner is established, ownership of specific members can be attributed to that owner and it is possible to limit the users who may access (that is, read) remove and alter each of those members.

So, the dictionary/repository may contain owned and non-owned members. Owners can control which dictionary users may access and manipulate the information that belongs to them.

Security Levels

Security for the contents of a dictionary is implemented by the Controller using a system of *security levels*. There are several types of security level associated with the members of a dictionary and several others associated with the authorized users of the dictionary. For each security level type there are up to 255 levels. Each level is identified numerically using numbers in the range 0 to 255.

The Controller may establish these security levels for each dictionary user:

- A general security level
- An (owner) specific security level

For the dictionary as a whole the Controller may establish these security levels:

- An insertion security level
- A protection security level

A user's *general security level* is examined when that user attempts to access, alter, or remove members of the dictionary for which no owner has been established.

A user whose general security level is lower than the *insertion security level* established for the dictionary or for whom no general security level is established, is not permitted to insert members into the dictionary or to alter or remove members that are already in the dictionary.

A user whose general security level is lower than the *protection security level* established for the dictionary or for whom no general security level is established, is not permitted to establish protection for members of the dictionary.

An (owner) *specific security level* is examined when a user attempts to access, alter and remove members belonging to a particular owner.

How to Protect Repository Members

The PROTECT command enables the repository Controller (and any other user with a security level which is sufficiently high) to establish protection for repository members in the following ways:

- By declaring an owner for a member (that owner must have been established by the Controller)
- By defining an *access security level* for a member
- By defining an *alter security level* for a member
- By defining a *remove security level* for a member.

When a user attempts to process a repository member the security levels established for that user by the repository Controller and the security levels established for the member via a PROTECT command are compared. Processing is either allowed or inhibited depending upon the result of that compared.

Security levels are not specified for the repository Controller; all repository management commands issued by the repository Controller (as identified by the password) are accepted.

Protection for Owned Members

Once ownership of a member is established (using the PROTECT command) you may process that member only if a specific security level, in respect of that owner, is established for you. And, the way in which you may process the member may be inhibited as follows:

- If an access security level is defined for the member you may access the member only if your specific security level is equal to it or higher
- If an alter security level is defined for the member you may alter the member only if your specific security level is equal to it or higher
- If a remove security level is defined for the member you may remove the member only if your specific security level is equal to it or higher

Protection for Non-owned Members

Repository members for which no owner has been declared may be protected using the PROTECT command. The command is used to establish access, alter and remove security levels for a member. Once security levels are established for a member, you may process that member only if your general security level is sufficiently high enough. So, the way in which you may process such members may be inhibited as follows:

- If an access security level is defined for the member you may access the member only if your general security level is equal to it or higher
- If an alter security level is defined for the member you may alter the member only if your general security level is equal to it or higher
- If an remove security level is defined for the member you may remove the member only if your general security level is equal to it or higher

Protection - an Example

For example, if a user has a general security level of 100 and a specific security level in respect of owner A of 150, then he can:

- Access all members owned by owner A that have an access security level of 150 or less
- Alter all members owned by owner A that have an access security level of 150 or less
- Remove all members owned by owner A that have an access security level of 150 or less
- Access, alter and remove, respectively, all non-owned members for which access alter and remove security levels of 100 or less have been established
- Access alter or remove, without restriction, all repository members for which no protection has been established

Interrogation

Access to members affects interrogations in these ways:

- An interrogation is rejected if a member named in it is inaccessible
- If an interrogation encounters an inaccessible member, any members on the reference path beyond it are excluded from the command output

For example, you could define two members A and B and a relationship from A to B. Consider the command WHAT CONSTITUTES A.

If the relationship is an EA relationship then:

- If A is inaccessible then the command is rejected
- If A is accessible then the command returns B, whether or not B is accessible

If the relationship is an ER relationship then:

- If A is inaccessible then the command is rejected
- If A and the relationship member are accessible then the command returns B, whether or not B is accessible
- If A is accessible but the relationship member is inaccessible then the command returns no members

So, by establishing users and owners and by manipulating security levels the repository Controller can control:

- Which users can insert members
- Which users can alter and remove unprotected members
- Which users can protect members
- Which users can access protected members and which of the protected members they can access
- Which users can alter protected members and which of the protected members they can alter
- Which users can remove protected members and which of the protected members they can remove

The Automatic Recovery System

The automatic recovery system is designed to safeguard the repository against corruption or loss of data arising from machine malfunction, operator action or software failure.

How It Works

When you execute any function or command that updates the repository, an indicator is set in the repository's error recovery dataset. This indicator is called the update lock. As the command is processed, any repository records that are to be updated are first copied in their pre-update state into the repository's error recovery dataset. When processing of the command is completed, the update lock is set off and the records are cleared from the error recovery dataset. (The clearance is effected by writing an end of file marker into the first record).

If a session (interactive or batch) is interrupted before the processing of the updating command is completed, the update lock remains set on. This triggers the recovery system on the next occasion that the repository is processed in update mode; either when a following job is submitted or when a command is received from a concurrent user. The records in the error recovery dataset are copied back into the repository's other datasets, the update lock is set off and the error recovery dataset is cleared. This restores the repository to the state it was in immediately before the interrupted updating command was received.

At the end of the recovery process, the message:

```
DM010301 THE REPOSITORY HAS BEEN AUTOMATICALLY RECOVERED
```

is output and processing continues with the command that triggered the recovery process.

Because the error recovery dataset is not opened when a repository is opened in read-only mode, automatic error recovery cannot take place in read-only mode.

Some functions are provided which enable the updating of several members with a single execution. When you use such a function the recovery system operates as though a separate command were issued for each member processed; that is, the update lock is set and unset, and the error recovery dataset records are written and cleared, as each individual member is processed. Following the interruption of such commands, it is not necessary to process again those members whose processing was already completed.

The unrestricted functions (that is, functions that are not restricted to use by the repository Controller) with which the repository may be updated are documented in this publication. The restricted commands with which the repository may be updated are documented in *ASG-Manager Products Controller's Manual*.

The error recovery dataset must be present whenever the repository is processed in update mode. It need not be present if the repository is open in read-only mode.

2 What Do You Want To Do?

The purpose of this chapter is to direct users who are unfamiliar with Manager Products to the documentation relevant to the task they wish to perform. For each task the relevant InfoBank panel name is given together with the page reference in this manual.

Accessing or Leaving a Repository

Do you want to:	Panel Name	Page
Open a repository?	DMRDICT000	119
Access a repository?	DMRAUTH000	78
Leave the current repository?	CMRCOM1110	113

Defining the Targeting Mode

Do you want to:	Panel Name	Page
Target commands to the current repository or to the Workbench Design Area?	CMRCOM1143	177

Updating the Repository

Do you want to:	Panel Name	Page
Create or update a member interactively?	CMRCOM3148	251
Create a member and at the same time encode it?	DMRADD000	49
Create a member without encoding it?	DMRINS000	152
Update an existing member and encode the changes?	DMRMOD000	178
Update an existing member without encoding the changes?	DMRALT000	52
Amend by name individual clauses and keywords in a member's encoded record?	AMEND000	69
Encode one or more members?	DMRENC000	129
Encode a selection of members?	DMRBULKE00	80
Remove one or more members?	DMRREM000	223
Encode or remove members only if they pass integrity checks?	CHECK000	111
Check the integrity of a member and set its integrity state?	VALID000	252
Copy a member to another member?	DMRCOP000	117
Rename a member?	DMRREN000	224
Replace a member?	DMRREPL000	226
Reinstate the last encoded source record of a member?	DMRREIN000	222

Protecting Repository Members

Do you want to:	Panel Name	Page
Protect one or more members from being accessed, updated or removed?	DMRPROT010	213
Lock a member to prevent it being concurrently updated?	LOCK0000	173

Interrogating the Repository

Finding Out What an Index-name Is

Do you want to:	Panel Name	Page
Find out whether an index-name is a repository member, alias, catalog classification or user-defined indexed attribute?	DMRWHAT060	259

Listing Index-names

Do you want to list:	Panel Name	Page
Repository members, aliases, catalogs or indexed attributes?	DMRLIST000	158
The history of repository members, aliases, catalogs or indexed attributes?	DMRLIST020	161

Finding Out About the Relationships Between Members

Do you want to:	Panel Name	Page
Find out whether one member uses a given member?	DMRDOES000	126
Find out which members use a given member?	DMRWHAT010	254
Find out which of a selection of members use a given member?	DMRWHI040	269
Find out which members are used by a given member?	DMRWHAT020	255
Find out which of a selection of members are used by a given member?	DMRWHI050	273
Display the encoded record of a member and list or display the encoded record of the members it uses?	DMRREPO000	233
Display particular clauses and attributes of a selection of members and list the members they use or are used by?	DMRGLOS040	144
Interrogate via relationships of a particular type?	DMRWHI000	265

Finding Out About Catalogs, Aliases, Attributes and Clauses of Members

Do you want to:	Panel Name	Page
Find out which members are catalogued under a specified classification or combination of classifications?	DMRWHAT030	257
Find out which of a selection of members are catalogued under a specified classification or combination of classifications?	DMRWHI080	277
Find out which members have a specified alias or combination of aliases?	DMRWHOSE00	291
Find out the alias types available?	DMRSHA000	237
Find out which of a selection of members have specified attributes and clauses?	DMRWHI090	277
Display the aliases, catalogues, clauses and attributes of members?	DMRGLOS000	152

Finding Out About the Ownership of Members

Do you want to find out:	Panel Name	Page
Who owns a specified member?	DMRWHO000	290
Which members belong to a specified owner?	DMRWHAT040	258
Which of a selection of members belong to a specified owner?	DMRWHI100	280

Finding Out About the Member Types Available in Your Repository

Do you want to:	Panel Name	Page
Display the member types available in your repository?	UDS7000	244
Find out the clauses, keywords and user defined attributes permitted in the definition of a particular member type?	DYRCOM4600	238

Displaying Dictionary Members

Do you want to display:	Panel Name	Page
The source record of one or more members?	DMRPRI000	208
The source record of a selection of members?	DMRBULKP00	91
The encoded record of one or more members?	DMREPO000	230
The encoded record of a selection of members?	DMRBULKR00	100
The encoded record of a selection of members in a preferred alternative layout?	DMRFORM000	135
The encoded record of a member and the members it uses?	DMRREPO030	232
The clauses and attributes of a selection of members and list the members they use or are used by?	DMRGLOS040	144

Selecting Index-names by Name, Time, User and Status

Do you want to select index-names by:	Panel Name	Page
Name?	DMRNAM005	294
Time and user?	DMRTIM000	308
Status?	STATUS2209	299

Manipulating KEPT-DATA Lists

Do you want to:	Panel Name	Page
Create a KEPT-DATA list or add member names, aliases, catalogs or user defined indexed attributes to an existing KEPT-DATA list?	DMRKEEP000	156
Drop member names, aliases, catalogs or user defined indexed attributes from a KEPT-DATA list?	DMRDROP000	127
Apply commands to the member names, aliases, catalogs or user defined indexed attributes in a KEPT-DATA list?	DMRPERF045	201
Delete a KEPT-DATA list?	DMRERAK00	131

Do you want to:	Panel Name	Page
Store a KEPT-DATA list on the MP-AID?	DMRSTRK00	250
Retrieve a KEPT-DATA list from the MP-AID?	DMRFETK00	132
Find out what KEPT-DATA lists you have created?	CMRCOM1258	217
Find out which member names, aliases, catalogs or user defined index attributes are kept in a KEPT-DATA list?	DMRWHI130	281

Performing a Series of Commands

Do you want to:	Panel Name	Page
PERFORM a series of commands or a COMMAND-STREAM member?	DMRPERF000	195

Reading From, and Writing To, CMS Files and MVS Datasets

Do you want to:	Panel Name	Page
Write the contents of the current buffer to a CMS file or MVS dataset?	CMRCOM3152	218
Read the contents of a CMS file or MVS dataset into the current buffer?	CMRCOM3141	218

3 Using the Dictionary

This chapter includes these sections:

Introduction to Dictionary Management Commands 29

Access Security Commands 31

Dictionary Manipulation Commands 33
Commands Which Enter Members Into a Dictionary. 33
Commands Which Alter or Remove Members of a Dictionary 35
Commands Which Manipulate KEPT-DATA Lists 36

Documentation Commands 38

Generating Output in Alternative Formats 39

Targeting Commands 40

Interrogation Commands 41

Commands Which Activate Other Commands 43

Effects of the Status Facilities on Dictionary Management 44

Effects of User Defined Syntax on Dictionary Management. 44

Controller's Commands 45

Introduction to Dictionary Management Commands

This chapter gives an overview of the commands with which a dictionary may be accessed, manipulated, documented and interrogated.

All Basic Dictionary Management commands and Extended Dictionary Management commands have been detailed.

Basic Dictionary Management commands. Are provided by the ControlManager Nucleus (selectable unit CMR-CM01).

Extended Dictionary Management commands. Are provided by the DataManager Nucleus (selectable-unit DMR-DD1).

Access security commands and the majority of dictionary manipulation and documentation commands are Basic Dictionary Management commands. Powerful documentation commands are provided by the ControlManager Nucleus and extended interrogation commands by the DataManager Nucleus.

These Basic Dictionary Management commands are provided by ControlManager:

- ADD
- ALTER
- AUTHORITY
- COPY
- DICTIONARY
- FORMAT AS
- INSERT
- LIST
- MODIFY
- PRINT
- REINSTATE
- REMOVE
- REPLACE
- REPORT
- SHOW

These commands are all Extended Dictionary Management commands provided by DataManager:

- ALSO
- BULK
- DOES
- DROP
- ERASE KEPT
- FETCH KEPT-DATA
- GLOSSARY
- KEEP
- PERFORM
- RENAME
- STORE KEPT-DATA
- WHAT
- WHICH
- WHO
- WHOSE

Some optional additional facilities provide further commands and keywords which are of value in dictionary management or enhance the operation of individual Basic and Extended Dictionary Management commands.

Where this is the case or where the facilities themselves affect the operation of dictionary management, then the documentation of these facilities, at command level, has been integrated in this overview.

These additional commands are available to users who have the necessary selectable units installed: FILE, SFILE, PROTECT, STATUS, RFILE, UPDATE, and REVERT

Refer to *ASG-Manager Products Advanced Status* and *ASG-Manager Products Basic Status* for details of the REVERT command.

The various DataManager Database Management System Interface facilities do not include separate commands but require variation in certain commands such as the provision of an additional keyword stating the database concerned.

Refer to the relevant DataManager Interface Facility manual for details of these variants.

Access Security Commands

These commands are concerned with access to a dictionary:

DICTIONARY. Opens a dictionary.

AUTHORITY. Accesses a dictionary.

PROTECT. Establishes security levels for dictionary members.

STATUS. Specifies the status in which the user will work.

A DICTIONARY command identifies and opens the dictionary that is required for processing.

More than one dictionary may exist in any installation, and more than one dictionary can be accessed during a Manager Products session, though only one can be open at any one time.

In the DICTIONARY command, the user declares the name of the dictionary that is to be opened. The dictionary name will have been established by the Controller, and is related via the job control statements to the physical files that contain the dictionary's datasets.

Once opened, a dictionary remains open until a further DICTIONARY command is input or until the end of the current Manager Products session. An optional keyword in the command enables the dictionary to be opened in read only mode.

The DICTIONARY command is a Basic Dictionary Management command provided by ControlManager.

An AUTHORITY command establishes the identity of a user requiring access to a dictionary.

Only users authorized to do so by the Controller can access any particular dictionary.

In the AUTHORITY command, the user declares a password which will have been registered in the dictionary by the Controller as belonging to that user.

Until an AUTHORITY command with a valid password for the dictionary currently opened has been received, no commands other than a DICTIONARY command can be accepted.

Thus, the first two commands in any Manager Products session must be a DICTIONARY command followed by an AUTHORITY command.

Once an AUTHORITY command has been accepted, any security levels that may have been registered in the dictionary by the Controller as pertaining to the user are applied, as may be appropriate, to all subsequent commands, until another AUTHORITY or DICTIONARY command is received. Security levels form part of the Audit and Security facility (selectable unit CMR-DD3).

The AUTHORITY command is a Basic Dictionary Management command provided by ControlManager.

A PROTECT command enables a user to establish security levels for individual members of the dictionary.

The command can establish progressively higher security levels that must be met if a user is to be permitted to access, alter or remove a member of the dictionary.

The same command can assign ownership of individual members.

For the command to be accepted, the user's general security level must be equal to or higher than the protection security level registered in the dictionary by the Controller.

The PROTECT command forms part of the Audit and Security facility (selectable unit CMR-DD3).

The STATUS command specifies the current status in which the ensuing commands are to operate.

The status facilities also provide the ability to list statuses, display statuses and, if the Advanced Status facility is installed, manipulate a status window. Status is provided by the Basic and Advanced Status facilities (selectable units CMR-DD2 and CMR-AD21 respectively).

Refer to Chapter 4, "Dictionary Management Commands," on page 47 for further details of the DICTIONARY, AUTHORITY and PROTECT commands.

Refer to *ASG-Manager Products Advanced Status* and *ASG-Manager Products Basic Status* manuals for details of the status facilities.

Dictionary Manipulation Commands

The commands for manipulating a dictionary can be divided into these:

- Commands which enter members into a dictionary:

ADD	FILE
BULK ENCODE	INSERT
COPY	SFILE
ENCODE	UPDATE

- Commands which alter or remove members of a dictionary:

ALTER	REPLACE
FILE	RFILE
REINSTATE	SFILE
REMOVE	UPDATE
RENAME	

- Commands which manipulate KEPT-DATA lists:

ALSO	FETCH KEPT-DATA
DROP	KEEP
ERASE KEPT-DATA	STORE KEPT-DATA

Commands Which Enter Members Into a Dictionary

An INSERT command inserts a member's source record into the dictionary's source dataset, and updates the index dataset. The source record is not validated at this stage.

An ENCODE command generates an encoded record from a member's source record, validates the syntax of the member definition, and, if no syntax errors are discovered, inserts the encoded record into the data entries dataset. The index dataset is updated. Any number of members named in the command can be encoded by a single command.

A BULK ENCODE command enables all or selected categories of members having source records to be encoded by a single command, without the need to name the members individually in the command. Otherwise it operates in the same way as ENCODE.

The BULK command has two other forms:

- BULK PRINT
- BULK REPORT

which display members of the dictionary.

An ADD command has the effect of INSERT plus ENCODE. It enables a new member's source record to be inserted into the source dataset and encoded, and, if no syntax errors are discovered, the encoded record to be inserted into the data entries dataset, by a single command. The index dataset is updated.

The UPDATE command displays an empty Update Buffer into which the source record of single member may be entered interactively. A subsequent SFILE command will close the buffer, insert the source record into the source dataset and update the index dataset. Alternatively, a FILE command will in addition validate and encode the source record and enter it into the data entries dataset.

The UPDATE command therefore combines the functions of the ADD, INSERT and ENCODE commands, as well as the member amendment commands ALTER, MODIFY and REPLACE.

A COPY command inserts into the source dataset a copy of the source record of an existing member of the dictionary as the source record for a new member and updates the index dataset. An alternative form of the command takes a copy of the source record of an existing member from one dictionary status and inserts it into another status.

The ADD, COPY, ENCODE and INSERT commands are Basic Dictionary Management commands provided by ControlManager.

The BULK command is an Extended Dictionary Management command provided by DataManager.

The UPDATE, FILE and SFILE commands are provided by the ControlManager Extended Interactive facility (selectable unit CMR-FE01).

Refer to "Documentation Commands" on page 38 for an overview of the BULK PRINT and BULK REPORT commands.

Refer to "Commands Which Alter or Remove Members of a Dictionary" on page 35 for an overview of the ALTER, MODIFY and REPLACE commands.

Refer to Chapter 4, "Dictionary Management Commands," on page 47 for further details of the INSERT, ENCODE, BULK ENCODE, ADD and COPY commands.

Refer to *ASG-ControlManager User's Guide* for further details of the UPDATE, SFILE and FILE commands.

Commands Which Alter or Remove Members of a Dictionary

An ALTER command amends a member's source record.

A MODIFY command has the effect of ALTER plus ENCODE. It enables a member's source to be amended, the amended record to be encoded, and, if no syntax errors are discovered, the encoded record to be inserted into the data entries dataset (replacing any previous data entries record of that member), by a single command. The index dataset is updated if necessary.

A REPLACE command enables both the source record and the data entries record of a member to be replaced by new records for that member, by a single command. The new source record replaces the old, is encoded, and, if no syntax errors are discovered, the newly encoded record replaces any previous data entries record of the member. The index dataset is updated if necessary.

An UPDATE command can display the source record of a specified member which may then be altered interactively. The source dataset alone may be altered by entering a subsequent SFILE command, or alternatively both the source dataset and the data entries dataset may be modified and, if necessary, the index dataset updated by encoding the updated source record with a FILE command.

An UPDATE command in which no member name is specified will display an empty Update Buffer into which a source record may be entered interactively. A subsequent RFILE command will result in this source record replacing the existing source of whichever member is specified in the command. The source dataset, data entries dataset and, if necessary, the index dataset are all updated.

The UPDATE command therefore combines the functions of the ALTER, MODIFY and REPLACE commands, as well as the member insertion commands ADD, ENCODE and INSERT.

The REINSTATE command will cause the last encoded source record of a member as held in the data entries dataset to replace the current source record as held in the source dataset when the two differ. The command will remove from the dictionary a member which has no encoded source record.

A REMOVE command deletes a member from the dictionary, provided that no other encoded members refer to that member in their encoded source record.

A RENAME command changes a member's name, provided that no other encoded members refer to that member in their encoded source record.

The ALTER, MODIFY, REPLACE, REINSTATE and REMOVE commands are Basic Dictionary Management commands provided by ControlManager.

The RENAME command is an Extended Dictionary Management command provided by DataManager.

The UPDATE, FILE, SFILE and RFILE commands are provided by the ControlManager Extended Interactive facility (selectable unit CMR-FE01).

Refer to "Commands Which Enter Members Into a Dictionary" on page 33 for an overview of the ADD, ENCODE and INSERT commands.

Refer to Chapter 4, "Dictionary Management Commands," on page 47 for further details of the ALTER, MODIFY, REPLACE, REINSTATE, REMOVE and RENAME commands.

Refer to *ASG-ControlManager User's Guide* for further details of the UPDATE, SFILE, FILE, and RFILE commands.

Commands Which Manipulate KEPT-DATA Lists

DataManager enables you to accumulate one or more KEPT-DATA lists of member/index-names in main storage and subsequently manipulate them with particular Manager Products commands. These KEPT-DATA lists may be saved as members on the MP-AID between Manager Products sessions.

A KEEP command prefixing particular Manager Products commands will accumulate a KEPT-DATA list of member/index-names in main storage from the output of the incorporated command. These KEPT-DATA lists may be given a name. You can hold one unnamed and one or more named KEPT-DATA lists in main storage.

An ALSO command prefixing particular Manager Products commands will add further member/index-names to a KEPT-DATA list from the output of the incorporated command.

A further KEEP command not preceded by the keyword ALSO and not specifying a KEPT-DATA list-name will overwrite the member/index-names previously accumulated in the existing unnamed KEPT-DATA list.

A further KEEP command not preceded by the keyword ALSO and specifying an existing KEPT-DATA list-name will overwrite the member/index-names previously accumulated in that named KEPT-DATA list.

A DROP command prefixing certain Manager Products commands will remove from the KEPT-DATA list those member/index-names output by the incorporated command.

You can find out which member/index-names are held in a specified KEPT-DATA list with the WHICH command.

For example, to find out which aliases are held in the unnamed KEPT-DATA list, enter:

```
WHICH ALIASES ARE KEPT;
```

You can see a list of current named and unnamed KEPT-DATA lists with the QUERY KEPT-DATA command.

To erase specified or all KEPT-DATA lists from main storage, there is an ERASE KEPT-DATA command.

The STORE KEPT-DATA command stores the member/index-names held in a specified KEPT-DATA list as a member on the MP-AID and allows them to be retained across executions of Manager Products. These MP-AID members are KEPT-DATA-LIST members. The FETCH KEPT-DATA command will retrieve the member into main storage.

The number of KEPT-DATA-LIST members which may be created on the MP-AID is dependent on the space for USER-MEMBERS available with the Logon Identifier in use. KEPT-DATA-LIST member names can be a maximum of ten characters in length and need not be delimited if made up of characters from the standard character set.

KEPT-DATA-LIST members can be displayed using the MP-AID PRINT command and listed with the MP-AID LIST command. They cannot be edited, and can only be removed by means of the MP-AID DELETE command.

The FETCH KEPT-DATA, MP-AID PRINT, MP-AID LIST and MP-AID DELETE commands can only be applied to those KEPT-DATA-LIST members stored on the MP-AID by a user with the same Logon Identifier or by the Systems Administrator. The QUERY KEPT-DATA command will only list KEPT-DATA lists accumulated in main storage by the current user.

KEPT-DATA lists can be invoked in certain Manager Products commands so that the operation of these commands only applies to the member/index-names contained in them.

Refer to "Definition Of Index-names" on page 316 for a definition of member/index-names.

Refer to Chapter 4, "Dictionary Management Commands," on page 47 for further details of the KEEP, ALSO, DROP, ERASE KEPT-DATA. FETCH KEPT-DATA, STORE KEPT-DATA and QUERY KEPT- DATA commands.

Refer to *ASG-ControlManager User's Guide* for further details of the MP- AID LIST. MP-AID PRINT and MP-AID DELETE commands, and for the rules governing the naming of MP-AID members.

Documentation Commands

These commands display all or selected categories of the members of a dictionary in varying formats:

PRINT command. Outputs the source records of selected members of a dictionary or selected parts of the source record of one member of a dictionary.

BULK PRINT command. Enables the PRINT command to be applied to all members of a dictionary that come within a selected category, without the need to name the members individually in the command.

REPORT command. Outputs details of selected encoded members of a dictionary, including information that cannot be obtained from their source records; for, example information concerning protection, interrelationships with other members, and when and by whom the record was last encoded.

BULK REPORT command. Enables the REPORT command to be applied to all members of a dictionary that come within a selected category, without the need to name the members individually in the command.

There is another version of the BULK command, BULK ENCODE.

FORMAT command. Enables the output of REPORT and BULK REPORT to be formatted in any of the ASG-preferred set of alternative layouts.

LIST command. Lists the names of all members of a dictionary, or of dictionary members that come within selected categories. For each member listed one line is output, giving details of the state of the member's records and the user's security standing with respect to the member.

LIST HISTORY is an alternative form of the LIST command which lists for each member or selected categories of members:

- The user's security standing with respect to the member
- When and by whom the member was inserted, last altered, and last encoded, within each status of the dictionary.

The LIST command can also list aliases, catalog classifications and (if the User Defined Syntax facility, selectable unit CMR-UD1, is installed) user-defined indexed attributes.

GLOSSARY command. Outputs any selection of the clauses that are common to all member types, for all encoded members of the dictionary, or for selected categories of such members, and/or lists all references to and/or from such members. Certain other attributes and clauses of FILE, MODULE. PROGRAM and SYSTEM member types can also be selected for output by the command. All attributes and clauses of any other member type can also be selected for output by the command.

The PRINT, REPORT and LIST commands are Basic Dictionary Management commands provided by ControlManager.

The BULK PRINT, BULK REPORT and GLOSSARY commands are Extended Dictionary Management commands provided by DataManager.

Refer to "Commands Which Enter Members Into a Dictionary" on page 33 for an overview of the BULK ENCODE command. Refer to "Generating Output in Alternative Formats" on page 39 for an overview of the FORMAT command.

Refer to "Dictionary Management Commands" on page 47 for further details of the PRINT, BULK PRINT, REPORT, BULK REPORT, LIST and GLOSSARY commands.

Generating Output in Alternative Formats

The output of Manager Products commands is normally generated in a layout that is predefined by ASG. You can change the layout of output by specifying an alternative format or by using Executive Languages. You can also translate Manager Products member types into the format of equivalent member types in other Manager Products dictionaries or in other vendors dictionary and/or directory systems.

The SET FORMAT command will establish an alternative format for the output of all subsequent REPORT and BULK REPORT commands. Both the standard format or any alternative format previously specified in a SET FORMAT command can be overridden by prefixing a REPORT or BULK REPORT command with a FORMAT AS or FORMAT VIA command. The alternative format specified in a FORMAT AS or FORMAT VIA command will only apply to the output generated by the particular REPORT or BULK REPORT command it prefixes.

The SET FORMAT and FORMAT AS commands are provided by ControlManager. The FORMAT VIA command is provided by the User Defined Output facility (selectable unit DYR-UD15).

Executive Languages can be used to change the output of all Manager Products commands. Executive Languages are provided by the User Defined Commands facility (selectable unit CMR-UD05). Refer to *ASG-DataManager User Defined Output* for further details.

The facility to translate Manager Products member types into the format of equivalent member types in another dictionary or directory is provided by DictionaryManager. Refer to *ASG-DictionaryManager User's Guide* for further details.

Refer to Chapter 4, "Dictionary Management Commands," on page 47 for details of the FORMAT AS, REPORT and BULK REPORT commands.

Refer to *ASG-DataManager User Defined Output* for details of the FORMAT VIA command.

Refer to *ASG-ControlManager User's Guide* for details of the SET FORMAT command.

Targeting Commands

If DesignManager is installed, the following commands may be applied either to the currently open dictionary, or to the current Workbench Design Area (WBDA):

- LIST
- REPORT
- MERGE

You must instruct the Integrated Dialogue Director to target the above commands at the WBDA or at the dictionary, as you require. To do so you must specify the Targeting Mode in which you want to work:

- DMS (to work with the dictionary)
- WBDA (to work with the Workbench Design Area).

When the Targeting Mode is DMS (DICTIONARY-MANAGEMENT-SOFTWARE) the above mentioned commands are applied to the currently open dictionary. When the Targeting Mode is WBDA (DESIGN-AREA) they are applied to the Workbench Design Area. As supplied by ASG the default Targeting Mode is DMS.

The MODE command is used to define the Targeting Mode during the current session or (when it is placed in Executive Routines or in Logon, Global or User-Defined Profiles) as a default.

Targeting Prefix commands (DMS and WBDA) are also provided for those occasions when you wish to override the current Targeting Mode without actually changing it.

Targeting Mode and Targeting prefix commands relate only to those commands (as above) which may be applied to the Workbench Design Area or to the dictionary.

Refer to "Documentation Commands" on page 38 for an overview of the LIST and REPORT commands.

Refer to Chapter 4, "Dictionary Management Commands," on page 47 for further details of the MODE, DMS and WBDA commands.

Refer to *ASG-DesignManager User's Guide* for details of the MERGE command.

Interrogation Commands

These commands enable members of a dictionary to be interrogated regarding their member type, aliases, catalogs, user-defined indexed attributes, ownership and interrelationships with other members:

DOES command. Enables a user to ascertain whether a specified member of a dictionary uses (refers to), directly or indirectly via other members, another specified member of a dictionary.

SHOW command. Comes in three forms:

- SHOW PRODUCE-OPTIONS is used to ascertain the currently active settings for each option of the Source Language Generation macro.
- SHOW UDS is used to ascertain which member types are available in a dictionary. If the User Defined Syntax facility (selectable unit CMR-UD1) has not been installed and the UDS table has not been adjusted then the ASG supplied table will be displayed.
- SHOW ALIAS-TYPES is used to ascertain the specific alias-type keywords that are available for use within a dictionary, and the number of general aliases that can be used in the definition of a member in a dictionary.

WHOSE command. Ascertain the member name(s) of the member(s) to which a stated alias applies.

WHAT command. Identifies a name and selects encoded members that meet certain conditions.

IS ascertains the member type of a named member. or whether the name stated in the command is an alias, catalog classification or (if the User Defined Syntax facility, selectable unit CMR-UD1, is installed) a user-defined indexed attribute. If the name stated in the command is an alias, the response also gives the member name(s) of the member(s) to which the alias applies.

USES ascertains which members use (refer to), a specified member of a dictionary. directly or indirectly via other members.

CONSTITUTES ascertains which members are used by (referenced by) a specified member of a dictionary. directly or indirectly via other members.

BELONGS-TO ascertains which members of a dictionary are owned by a specified owner. This form of the command is provided by the Audit and Security facility (selectable unit CMR-DD3).

FORMS ascertains which members of a dictionary are catalogued under a specified classification or combination of classifications. A series of AND USES, AND CONSTITUTES, AND BELONGS-TO, or, OR USES, OR CONSTITUTES, OR BELONGS-TO conditions can be included in the WHAT FORMS command to select members catalogued under a classification, or combination of classifications, that also meet all or some of the conditions specified in the command. The conditions available with the variants of the WHICH command can also be included in the WHAT FORMS command in this manner.

WHICH command. Ascertains which of a selected category of encoded members, aliases, catalog classifications or (if the User Defined Syntax facility, selectable unit CMR-UD1, is installed) user-defined indexed attributes satisfy specified conditions.

USES ascertains which of a selected category of members use (refer to) a specified member of a dictionary, directly or indirectly via other members.

CONSTITUTES ascertains which of a selected category of members' aliases, catalog classifications or user-defined indexed attributes are used by (referenced by) a specified member of a dictionary. directly or indirectly via other members.

FORMS ascertains which of a selected category of members of a dictionary are catalogued under a specified classification or combination of classifications.

HAS or HAVE ascertains which of a selected category of members of a dictionary have particular clauses and attributes in their definition.

ARE KEPT and ARE KEPT IN ascertains which of a selected category of members of a dictionary or aliases, catalog classifications or user-defined indexed attributes are kept in a specified KEPT-DATA list.

ARE USED-BY ascertains which of a selected category of members of a dictionary or aliases, catalog classifications or user-defined indexed attributes are directly used by (referenced by) a specified number of members.

ARE REFERRING-TO ascertains which of a selected category of members of a dictionary directly use (refer to) a specified number of members.

BELONGS-TO ascertains which of a selected category of members of a dictionary are owned by a specified owner. This form of the command is provided by the Audit and Security facility (selectable unit CMR-DD3).

A series of conditions can be specified, each separated by the keyword AND or OR, so as to ascertain which members, aliases, catalog classifications or user-defined indexed attributes meet all or some of the conditions. The conditions available with the variants of the WHICH command can also be included in the WHAT FORMS command in this manner.

The keywords DOES NOT can be specified so as to ascertain which members, aliases, catalog classifications or user-defined indexed attributes do not meet one or more conditions.

WHO command. Ascertains the ownership of specified members of a dictionary.

The WHO command forms part of the Audit and Security facility (selectable unit CMR-DD3).

The SHOW ALIAS-TYPES and SHOW UDS commands are Basic Dictionary Management commands provided by ControlManager.

The DOES, SHOW PRODUCE-OPTIONS WHOSE, WHAT and WHICH commands are Extended Dictionary Management commands provided by DataManager.

Refer to Chapter 4, "Dictionary Management Commands," on page 47 for further details of the DOES, SHOW. WHOSE, WHAT, WHICH and WHO commands.

Commands Which Activate Other Commands

A PERFORM command enables a series of other Manager Products commands to be set up dynamically to operate on all members or on selected categories of members of the dictionary.

An alternative form of the command activates a stream of Manager Products commands that have been stored in the dictionary, in a COMMAND-STREAM member. Selection of members to be processed by the stored commands is specified in the PERFORM command.

The PERFORM command is an Extended Dictionary Management command provided by DataManager.

Refer to "PERFORM (DMR-DD1)" on page 195 for further details of the PERFORM command.

Effects of the Status Facilities on Dictionary Management

In relation to the ControlManager optional additional facilities, Basic Status (selectable unit CMR-DD2) and Advanced Status (selectable unit CMR-AD21) command specifications can be regarded as falling into three groups.

The first group consists of those commands whose specifications are unaffected by the presence or absence of the Status facilities, for example, the AUTHORITY command.

The second group comprises those commands whose format includes keywords that relate to Status, for example, the BULK and GLOSSARY commands. For these commands, the definition of the affects of the Status facilities on the operation of the command is integrated into the command specification. Users who have not implemented Status facilities can simply ignore all references to statuses in the command specifications. The STATUS command itself can be ignored by such users.

The third group contains those commands that have no directly status-related keywords but whose operation can be affected by the Status facilities, for example, the ADD and WHAT commands.

Refer to *ASG-Manager Products Advanced Status* and *ASG-Manager Products Basic Status* for further details of the effects of status on dictionary management.

Effects of User Defined Syntax on Dictionary Management

The specifications of some commands could be affected in points of detail where the optional additional facility, User Defined Syntax (selectable unit CMR-UD1), is installed. These are the main areas in which commands could be affected:

- The member-type keywords available in some commands. These keywords may be:
  - Selection keywords, to limit the action of the command to members of specified types, for example, in the command:

    ```
    LIST member-type;
    ```

  - Keywords to limit the extent to which reference paths from a member are followed, for example, in the command:

    ```
    REPORT member-name DOWN-TO member-type;
    ```

  - Keywords in interrogation commands, to limit the response to the names of members of required types, for example, in the command:

    ```
    WHICH member-type USE member-name;
    ```

- The validation of the member types of members to which reference is made
- From specific clauses of a member's definition. For example, the member types which can be referred to in a CONTAINS clause and accepted by an ENCODE command can differ.
- The types of dummy members that can be created, for example by an ADD command, or processed, for example by the following BULK command:

  ```
  BULK PRINT DUMMY member-type;
  ```

- The attributes that can be accepted on encoding or whose absence can cause encoding to fail. User-defined attributes will also affect the output from REPORT commands. If attributes are defined as indexed attributes, then they appear in the output from a LIST command, and can be selected by a LIST selection command or a GLOSSARY selection command. Whether or not attributes are indexed, they can be specified for output by a GLOSSARY command and interrogated by a WHICH command or a WHAT FORMS command.

In environments in which User Defined Syntax is installed, users should adjust command specifications to take account of the member types, member-type relationships and attributes defined for their installation.

Controller's Commands

Other commands are available to the Controller for initializing and reorganizing a dictionary, and for controlling its security and its statuses. Details of these commands are given in *ASG-Manager Products Controller's Manual*.

4 Dictionary Management Commands

This chapter describes the dictionary management commands provided by Manager Products. The commands are documented in alphabetic order of command name and include:

ADD (CMR-CM01) **49**
ALSO. **52**
ALTER (CMR-CM01) **52**
AMEND (CMR-CM01) **69**
AUTHORITY (CMR-CM01) **78**
BULK ENCODE (DMR-DD1) **80**
BULK PRINT (DMR-DD1) **91**
BULK REPORT (DMR-DD1) **100**
CHECK (MMR-PF61) **111**
CLOSE Command. **113**
CONTRAST **115**
COPY (CMR-CM01). **117**
DICTIONARY (CMR-CM01) **119**
DMS. **125**
DOES (DMR-DD1) **126**
DROP (DMR-DD1) **127**
ENCODE (CMR-CM01) **129**
ERASE KEPT-DATA (DMR-DD1) **131**
FETCH KEPT-DATA (DMR-DD1). **132**
FILE (CMR-FE01) **133**
FORMAT AS (CMR-CM01) **135**
GLOSSARY (DMR-DD1) **136**
INSERT (CMR-CM01) **152**
KEEP and ALSO (DMR-DD1). **156**
LIST (CMR-CM01) **158**
LOCK (CMR-WS01). **173**

MODE, QUERY MODE, DMS, WBDA (CMR-CM01) **177**
MODIFY (CMR-CM01) **178**
PERFORM (DMR-DD1) **195**
PRINT (CMR-CM01) **208**
PROTECT (CMR-DD3) **213**
QUERY KEPT-DATA (CMR-CM01) **217**
QUERY MODE **218**
READ and WRITE (CMR-FE01, CMR-TP7, CMR-TP8) **218**
REINSTATE (CMR-CM01) **222**
REMOVE (CMR-CM01) **223**
RENAME (DMR-DD1) **224**
REPLACE (CMR-CM01) **226**
REPORT **230**
RFILE (CMR-FE01) **236**
SHOW ALIAS-TYPES (CMR-CM01) **237**
SHOW MEMBER-TYPE (CMR-CM01) **238**
SHOW PRODUCE-OPTIONS (CMR-CM01) **242**
SHOW UDS (CMR-CM01) **244**
SOURCE-FILE (CMR-FE01) **249**
STORE KEPT-DATA (DMR-DD1) **250**
UPDATE (CMR-FE01) **251**
VALIDATE (MMR-PF61) **252**
WBDA **253**
WHAT (DMR-DD1) **253**
WHICH (DMR-DD1) **265**
WHO (CMR-DD3) **290**
WHOSE (DMR-DD1) **291**
WRITE **292**
Applying Commands to a Selection Of Index-names Based on Name, Status, Time and User 293
Collective Member Types **316**
Definition Of Index-names **316**

ADD (CMR-CM01)

The ADD command inserts a member's source record into the dictionary, generates an encoded record from the newly inserted record, and inserts the encoded record into the Data Entries dataset.

How to Insert and Encode a Member

To add a member in the current status, enter:

```
ADD member-name ;
member-definition
;
```

where:

member-name is the name of a member that does not already exist in the dictionary, or of a member that exists as a dummy.

member-definition is the member's source record.

For the syntax of member definition, enter:

```
HELP member-type-name-S ;
```

where:

member-type-name is the name of the type of member you want to enter into the dictionary. For example:

```
HELP MODULE-S ;
```

Use the SHOW UDS command to find out which member types are available in your dictionary.

For example, to ADD the member MOD-DEPT-HIST, enter:

```
ADD MOD-DEPT-HIST ;
MODULE
    INPUTS NO-EMP-BFR-UPD, NO-EMP-LEAV, NO-EMP-NEW
    UPDATES NO-EMP-AFT-UPD
;
```

If a source record does not already exist for member-name, then a source record (containing the member definition lines) is entered into the dictionary, and an encoded record is generated from this source record.

If a source record already exists for member-name the command and member definition are rejected and processing continues with the next command.

Unless you include a NUMBER clause in your ADD command, lines in the source record are numbered in increments of 100, starting with 100.

You can only ADD a member if your general security level is equal to or higher than the dictionary's insertion security level and is sufficient for you to access any members directly used by the added member.

The ADD command is treated as an updating command by the automatic error recovery system.

Note:

- Details relating to security levels are applicable only if the Audit and Security facility (selectable unit CMR-DD3) is installed
- Details relating to statuses are applicable only if either the Basic or Advanced Status facility (selectable units CMR-DD2 and CMR-AD21 respectively) is installed.

Refer to *ASG-ControlManager User's Guide* for the member naming rules.

Refer to "Security Levels" on page 18 for details of security levels.

Refer to the *ASG-Manager Products Status Concepts* manual for details of statuses.

How to Number Lines in a Member's Source Record

To ADD a source record with lines numbered in specified increments, enter:

```
ADD member-name NUMBER n ;
member definition
;
```

where *n* is an unsigned integer in the range 1 to 10000 (inclusive). The lines in the source record are numbered in increments of *n*, the first line being number *n*.

For example, to add a member named MOD-DEPT-HIST with lines numbered in increments of 200, enter:

```
ADD MOD-DEPT-HIST NUMBER 200 ;
MODULE
     INPUTS NO-EMP-BFR-UPD, NO-EMP-LEAV, NO-EMP-NEW
     UPDATES NO-EMP-AFT-UPD
;
```

If you do not specify a NUMBER clause, lines in the source record are numbered in increments of 100, starting with 100.

If you specify an integer of less than 1 or greater than 10000 this message displays:

```
DM01182I n NOT VALID INCREMENT - '100' ASSUMED
```

and *n* is assumed to be 100.

If you specify a signed integer in the range 1 to 10000 (inclusive) this message displays:

```
DM00127W n INVALID INTEGER
```

the sign is automatically removed, and *n* is treated as a valid integer.

Similarly, if you do not specify an integer this message displays:

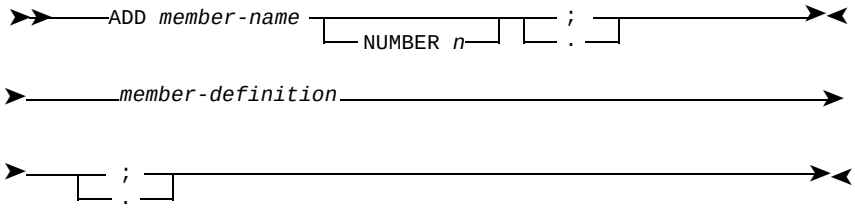
```
DM00127W ; INVALID INTEGER
```

and *n* is assumed to be 100.

If you specify an integer that causes the line numbers to extend into more than five digits, then, with one exception, only the last five digits are output in any printout or display of the source record. The exception is the response to a DISPLAY amendment line in an ALTER or MODIFY command; then all eight digits display.

Refer to "ALTER (CMR-CM01)" on page 52 for details of the DISPLAY amendment line.

ADD Syntax



where:

member-name is the name of the dictionary member

n is an unsigned integer in the range 1 to 10000, being the line number for the first line and the increment for subsequent line numbers of the member's source record

member-definition is the member's source record.

ALSO

Refer to "KEEP and ALSO (DMR-DD1)" on page 156 for details of the ALSO command.

ALTER (CMR-CM01)

To alter the source record of an individual member.

Refer to"ALTER Syntax" on page 67 for the syntax of the ALTER command.

Introduction to the ALTER Command

The ALTER command enables you to alter the source record of a member in the current status. (You can create data entries records from individual source records with the ENCODE command or from a selection of source records by using the BULK ENCODE command.)

The ALTER command is entered in two stages.

The first stage identifies the member to be altered. You can also specify one of the following optional keywords:

- WIPE-OUT deletes a member's existing source record
- RENUMBER renumbers the lines within a member's source record
- PRINT defines whether or not the altered member is displays on your screen once the ALTER command is processed.

For example:

```
ALTER PROJECT-NO WIPE-OUT ;
```

deletes PROJECT-NO's source record.

The second stage comprises one or more amendment lines.

An amendment line can be:

- An insertion line or replacement line. These lines either have no initial keyword, but commence with a line number, or commence with the keyword LINE.
- An editing instruction, for example, BLANK, CHANGE, COPY, DELETE, FILL or MOVE
- A control instruction, for example, ACCEPT, DISPLAY, END, LOCATE, QUIT, RENUMBER, a semi-colon or a full stop.

You can enter amendment lines in any order, but each amendment line must be wholly contained on one input-line.

You must enter the first element of each amendment line (that is, a line number, LINE or any of the editing or control instructions listed above) in column 1. You must also enter the final terminator in column 1.

For example:

```
10 ITEM
20 HELD-AS NUMERIC 5
;
```

creates a new source record for PROJECT-NO with line numbers increasing in steps of ten.

If you enter the syntax of the first stage incorrectly, the second stage is not processed.

If you enter the syntax of the first stage correctly, the member you are ALTERing is still available to other users for updating purposes while you enter the amendment lines in the second stage.

If another user does update the member between acceptance of the first stage and completion of the second stage of your ALTER command, the amendment lines are abandoned and the entire ALTER command rejected. Processing continues with the next command.

Use the LOCK or PROTECT commands before the ALTER command to prevent other users concurrently updating a member that you want to ALTER.

You can only ALTER a member if your security level is equal to or higher than the dictionary's insertion security level and is sufficient for you to alter that member.

ALTER is treated as an updating command by the automatic recovery system.

Note:

- Details relating to locking only apply if the Workstation Interface facility (selectable unit CMR-WS01) is installed.
- Details relating to security levels only apply if the Audit and Security facility (selectable unit CMR-DD3) is installed.
- Details relating to statuses only apply if either the Basic or Advanced Status facilities (selectable units CMR-DD2 and CMR-AD21 respectively) are installed.

Refer to *ASG-Manager Products Status Concepts* for details of updating members in a multi-status environment.

Refer to "The Security System" on page 17 for details of the Security System.

Renumbering Lines (RENUMBER)

To renumber *all* the lines of a member's source record and change each line number by a specific integer, commencing with that integer, enter:

```
ALTER member-name RENUMBER n ;
```

where:

member-name is the name of the dictionary member to be altered.

n is an unsigned integer with a value of 10000 or less.

For example, if *n* is 50, the lines of a member's new source record are renumbered in steps of 50, commencing at 50.

If you do not include an integer in your command, 100 is assumed.

You can also include a RENUMBER amendment line in the second stage of your command to renumber specific lines within the member's source record.

If you want to enter more than one of the optional keywords RENUMBER,

WIPE-OUT and PRINT or NOPRINT, you can include them in the first stage of the command in any order.

Deleting a Member's Current Source Record (WIPE-OUT)

To delete a member's current source record so that you can redefine the entire member, enter:

```
ALTER member-name WIPE-OUT ;
```

where *member-name* is the name of the dictionary member to be altered.

The optional keyword WIPE-OUT is useful if, for example, you want to rewrite a member's entire definition and reorganize its clauses so that each clause appears on a specific line number.

If you want to enter more than one of the optional keywords RENUMBER, WIPE-OUT and PRINT or NOPRINT, you can include them in the first stage of the command in any order.

Controlling Output from a Command

To display any amendments you make to a member and to display the member's altered source record upon termination of the ALTER command, enter:

```
ALTER member-name PRINT ;
```

To suppress the output of the member's new source record, enter:

```
ALTER member-name NO-PRINT ;
```

where *member-name* is the name of the dictionary member to be altered.

NO-PRINT can be replaced with the alternative keyword NOPRINT.

By default, the amendment lines and the member's new source record are output.

If you want to enter more than one optional keywords RENUMBER, WIPE-OUT and PRINT or NOPRINT, you can include them in the first stage on the command in any order.

Inserting a New Line

To insert a new line, include the amendment line:

```
m source-line
```

where:

m is the number of the new line, followed by a space.

source-line is the line to be inserted. It is assumed to begin immediately after the space.

If *m* identifies an existing line, that line is replaced by the new line (source-line).

To insert a line of spaces, include:

```
m
```

Deleting a Line (DELETE)

To delete a line, include the amendment line:

```
DELETE line-number
```

To delete lines with numbers in the range *m* to *n* include:

```
DELETE m THRU n
```

THRU can be replaced with the alternative keyword TO.

To delete the first line in the member containing a given string, include:

```
LOCATE 'string'
DELETE LINE
```

Replacing a Line

To replace a line, include the amendment line:

m `source-line`

where:

> *m* is the number of the line you want to replace, followed by a space.
>
> *source-line* is the new replacement line. It is assumed to begin immediately after a space

If the replacement line (source-line) is shorter than the line it replaces, the rest of the line is filled with spaces.

To replace a line with spaces, include:

m

To replace each line numbered in the range *m* through *n* with a character string, include:

`FILL` *m* `THRU` *n* `'string'`

To replace a line numbered *n* by a line numbered *m*, include:

```
MOVE m TO n
```

Line *m* is moved to line *n* and is deleted from its original position.

To replace a line with a copy of another line, include:

`COPY` *m* `TO` *n*

The line numbered *n* is replaced by a copy of the line numbered *m*.

To replace the first line in the member containing a given string of characters, include:

```
LOCATE 'string'
LINE source-line
```

You cannot alter the whole of a line which is already completely full by a replacement line (unless a different input device which has a longer record length than that used for the original insertion of the line is used).

The reason for this is that the replacement line must include the line number, or the keyword LINE, together with a space before the replacement text can commence. Thus, if you want to replace text on line 200 (in which every character space is currently used), you would enter:

```
200 source-line
```

As 200 takes up four character spaces, you cannot specify new text for the last four character spaces of the line. These four characters will appear as blank spaces.

If you want to change these spaces to text you should use either the CHANGE or the FILL amendment lines.

Copying a Line (COPY)

To copy a line, include the amendment line:

```
COPY line-number BEFORE r
```

where:

> *line-number* is the number of the line to be copied.
>
> *r* is a line number identifying the position before which the copied line is to be inserted. The line which is copied is left unchanged in its original position.

To copy lines with numbers in the range *m* to *n* inclusive, include:

```
COPY m THRU n BEFORE r
```

To copy the first line in the member containing a given string, include:

```
LOCATE 'string'
COPY LINE BEFORE r
```

To insert a line before the first line containing a given string, include:

```
LOCATE 'string'
COPY line-number BEFORE LINE
```

To insert the line(s) after *r* instead of before, replace *BEFORE* with AFTER in the formats given above.

To replace a line numbered *r* with a copy of another line numbered *m* include:

```
COPY m TO r
```

The maximum number of lines that can be inserted by a COPY amendment line is 32,767.

The line numbering of the inserted lines is controlled automatically. If necessary, existing lines may also be renumbered.

Moving a Line (MOVE)

To move a line, include the amendment line:

```
MOVE line-number BEFORE r
```

where:

line-number is the number of the line to be moved.

r is a line number identifying the position before which the line is to be inserted.

It does not need to be the number of an existing line.

The line which is moved is deleted from its original position.

To move lines with numbers in the range *m* to *n* inclusive, include:

```
MOVE m THRU n BEFORE r
```

where *r* must not be within the range of lines to be moved.

To move the first line in the member containing a given string, include:

```
LOCATE 'string'
MOVE LINE BEFORE r

LOCATE 'string'
MOVE line-number BEFORE LINE
```

To insert the line(s) after *r* instead of before, replace *BEFORE* with AFTER in the formats given above.

The numbering of inserted lines is controlled by default. If necessary existing lines may also be renumbered.

To move a line numbered *m* to replace a line numbered *r*, include:

```
MOVE m TO r
```

The maximum number of lines that can be inserted by a MOVE amendment line is 32,767.

Replacing a String (CHANGE)

To change all occurrences of a string to another string, include the amendment line:

```
CHANGE ALL 'string-1' TO 'string-2'
```

where:

string-1 is the string to be changed

string-2 is the new string and may be shorter or longer than *string-1*.

For example, to delete all occurrences of the string APRIL from a member, include:

```
CHANGE ALL 'APRIL' TO ' '
```

To change all occurrences of a string in a given line, include:

```
CHANGE line-number ALL 'string-1' TO 'string-2'
```

To change all occurrences of a string in lines with numbers in the range *m* to *n* inclusive, include:

```
CHANGE m THRU n ALL 'string-1' TO 'string-2'
```

If you do not include the keyword ALL in the above amendment line, only the first occurrence of the string in each line is changed.

For example, to change the first occurrence of the string JAN to the string FEB, for each line in which it occurs, enter:

```
CHANGE 'JAN' TO 'FEB'
```

To change the first occurrence of a string in the member, include:

```
LOCATE 'string-1'
CHANGE LINE 'string-1' TO 'string-2'
```

Filling a Line or Column with Spaces (BLANK)

To fill the whole of a line with spaces, include the amendment line:

```
BLANK line-number
```

To blank out every line with numbers in the range *m* to *n* inclusive, include:

```
BLANK m THRU n
```

To blank out columns *a* to *b* of every line, include:

```
BLANK COLUMNS a THRU b
```

If THRU *b* is omitted, only column *a* is blanked out.

To blank out columns *a* to *b* in a block of lines, include:

```
BLANK m THRU n COLUMNS a THRU b
```

If THRU *n* is omitted, only the specified columns in the line numbered *m* are blanked out.

To blank out the first line in the member containing a given string, include:

```
LOCATE 'string'
BLANK LINE
```

Filling Columns of Text with a Specified String (FILL)

To replace columns *a* to *b* of every line with a specified string, include the amendment line:

```
FILL COLUMNS a THRU b 'string'
```

If *string* is shorter than the columns to be filled, the string is extended with spaces.

If *string* is longer, the amendment-line fails. If you omit THRU *b*, only column *a* is replaced and string must be a single character.

To replace columns *a* to *b* in a given line with a string, include:

```
FILL line-number COLUMNS a THRU b 'string'
```

To replace columns in lines with numbers in the range *m* to *n* inclusive with a string, include:

```
FILL m THRU n COLUMNS a THRU b 'string'
```

To fill columns in the first line of the member containing a given string, include:

```
LOCATE 'string-1'
FILL LINE COLUMNS a THRU b 'string-2'
```

Checking Changes Before they are Implemented (DISPLAY)

To check any changes that you have made using amendment lines before the member is copied into the source dataset, include one of the following amendment lines.

To print or display the whole of the working storage record, include the amendment line:

```
DISPLAY
```

To print or display lines with numbers in the range *m* to *n* inclusive, include:

```
DISPLAY m THRU n
```

If you omit THRU *n*, only the line numbered *m* is output.

To output the whole of the record from a given line onwards, include:

```
DISPLAY m THRU END
```

To output the first line in the member containing a given string, include:

```
LOCATE 'string'
DISPLAY LINE
```

The CHANGE, FILL or BLANK amendment lines can be followed by PRINT to display the changes made by the lines, for example:

```
CHANGE ALL 'string-1' TO 'string-2' PRINT
```

In interactive environments you can correct a mistake, or abandon the amendment by including QUIT. In batch environments DISPLAY followed by QUIT allows you to check the effect of the amendments without them being executed.

For example:

```
ALTER EMP-IDENT ;
CHANGE 'ABC' TO 'DEF'
DISPLAY
QUIT
```

displays the amendment line, followed by the number of changes made and the member, with all instances of ABC changed to DEF without altering the source record of EMP-IDENT.

If you are happy with the result, you can run the amendments again without the DISPLAY and the QUIT.

The LOCATE Feature

LOCATE scans the whole or part of the member for a character string. If the string is found, the number of the line containing it is stored in memory and is printed or displayed. If an amendment line subsequently refers to LINE, the amendment is made to the LOCATEd line whose number is stored in memory.

For example:

```
LOCATE 'ABC'
DELETE LINE
```

deletes the first line in the member containing the character string, `ABC`.

To scan the whole of the member and to output every line in which the specified string is detected, include the amendment line:

```
LOCATE ALL 'string'
```

No line number is stored in memory.

To locate the first occurrence of a string in lines with numbers in the range *m* to *n* inclusive, include:

```
LOCATE m THRU n 'string'
```

To locate the first occurrence of a string from line *m*, include:

```
LOCATE m THRU END 'string'
```

To locate the first occurrence of a string from the line located by the last LOCATE command, include:

```
LOCATE NEXT 'string'
```

To locate the next occurrence of the string specified in the last LOCATE command, include:

```
LOCATE NEXT
```

The number of the line stored in memory by LOCATE is updated if the line is renumbered by a RENUMBER, MOVE or COPY amendment line. If the line is MOVEd, the line number is deleted from memory.

If the string is not found, an error message is output in batch mode and a warning message in online mode. The member will not be filed or encoded unless the ALTER command is terminated by the ACCEPT keyword.

If you use the ACCEPT keyword, the syntax of the altered member is not checked for errors; its use may result in a corrupted source member.

Renumbering Lines (RENUMBER FROM, THRU, TO)

To renumber lines with numbers in the range *n1* to *n2* inclusive, in steps of *n3*, include the amendment line:

```
RENUMBER n1 THRU n2 IN n3
```

where:

n1 and *n2* are line numbers.

n3 is an unsigned integer defining the increment by which line numbers are increased.

The number of the last renumbered line must not be greater than the number of the line following *n2*.

For example, to renumber the lines numbered 1600 to 3800 in increments of 200, include the amendment line:

```
RENUMBER 1600 THRU 3800 IN 200
```

To renumber all lines in the range *n1* to *n2* commencing with the integer *n3* in increments of *n4*, enter:

```
RENUMBER n1 THRU n2 FROM n3 IN n4
```

where:

n1 and *n2* are line numbers.

n3 is an unsigned integer defining the line number of the first of the newly renumbered lines.

n4 is an unsigned integer defining the increment by which line numbers are increased.

When renumbered, *n2* must be less than the number of the following line.

The integer *n3* must be greater than the number of the line preceding *n1*.

For example, to renumber 10 lines numbered 10 to 100 in increments of 2, commencing with the number 50, include the amendment line:

```
RENUMBER 10 THRU 100 FROM 50 IN 2
```

If, when the member's new source record is renumbered, the increment is such that the new line numbers are greater than five digits long, only the last five digits of each line displays.

If you need to display the full line numbers use a DISPLAY amendment line, whereupon all eight digits display on any output.

If you renumber a member's line numbers, any amendment lines that you enter subsequently must take account of the new numbering.

You can optionally include leading zeros in your RENUMBER amendment line, provided that the number you specify does not exceed eight digits.

If you do not include RENUMBER in your command, the line numbering is controlled automatically.

The following is an example of how lines are renumbered when you enter a COPY amendment line without also entering a RENUMBER amendment line.

If you enter:

```
COPY 200 THRU 400 AFTER 600
```

the lines inserted are numbered:

```
100 XXX    ====>    100 XXX
200 CCC             200 CCC
300 CCC             300 CCC
400 CCC             400 CCC
500 XXX             500 XXX
600 XXX             600 XXX
700 XXX             610 CCC
                    620 CCC
                    630 CCC
                    700 XXX
```

If the new lines cannot be contained within the existing line numbering, subsequent lines are also renumbered in steps of 100.

If you do not want to adopt the automatically inserted line numbers, you can use a RENUMBER amendment line to define your own line numbering system for the affected lines.

For example, if you want to move ten lines numbered 10 to 100 after the line currently numbered 800, you would enter:

```
MOVE 10 THRU 100 AFTER 800
```

The inserted lines are automatically renumbered 900 to 1800.

To specify that you want the steps between the inserted lines to remain as 10, enter:

```
RENUMBER 900 THRU 1800 IN 10
```

Terminating an ALTER Command

To terminate your ALTER command and implement the changes specified in the amendment lines included in the command, enter one of the following amendment lines:

```
ACCEPT
END
;
.
```

In this context, END is interpreted as an amendment terminator, not as a LOGOFF command.

To terminate your ALTER command and abandon any changes specified in the amendment lines included in the command, enter:

```
QUIT
```

This message is output on acceptance of the QUIT amendment line:

```
DM01165E AMENDMENT ABORTED
```

Enter the END, ACCEPT and QUIT amendment lines in upper-case.

Examples of the ALTER Command

To delete lines 300 and 400 from PROJ-NAME and insert three new lines, enter:

```
ALTER PROJ-NAME ;
DELETE 300
400   RANGE 'HQ100A' TO 'HQ999Z'
410        ELSE RANGE 'NA100A' TO 'NZ999Z'
420               IF DEPARTMENT = 'NORTHERN'
;
```

To insert after the first line of OFF-TEL-NO, a line containing the string TELEPHONE CONTACT, enter:

```
ALTER OFF-TEL-NO ;
LOCATE 'TELEPHONE CONTACT'
190000 'OR IF NIGHT SHIFT: EXT. 210'
MOVE 190000 AFTER LINE
;
```

To renumber all the lines of a member named TAX-CODE in steps of 10, with the first line being numbered 10, enter:

```
ALTER TAX-CODE RENUMBER 10 ;
```

To erase the contents of the current source record of a member named PROD-REVIEW, enter:

```
ALTER PROD-REVIEW WIPE-OUT ;
```

To prevent a copy of the source record of a member named DEBIT-FORM, which you have altered, being printed, enter:

```
ALTER DEBIT-FORM NOPRINT ;
```

Messages and Error Correction

If no source record exists for the specified member-name, the command (including the amendment lines) is rejected and this message is output:

```
DM01190E  member-name HAS NO SOURCE TO BE ALTERED
```

If someone else has written another amended version of the member to the Source dataset while your ALTER command was being processed, this message is output:

```
DM01170E  member-name SOURCE MEMBER UPDATED BY ANOTHER USER
```

If an error condition arises during the processing of amendment lines, then:

- In an interactive processing run, you can correct the error by submitting a further amendment line, or you can abandon the ALTER command by entering a QUIT amendment line
- In a batch processing run, subsequent amendment lines are processed if possible, so that any subsequent errors in the attempted amendments can also be detected. The amended version is not written to the source dataset, unless you submit an amendment line of ACCEPT.

If, in a RENUMBER amendment line, you enter an integer which is less than 10000 but is signed, the command is accepted but this message is output:

```
INVALID INTEGER - SIGN REMOVED
```

If you enter a QUIT amendment line to prevent the amended version of the record being written to the source dataset, the following message is output:

```
DM01165E  AMENDMENT ABORTED
```

and if you are working interactively, processing continues with the next command you enter.

ALTER Syntax

```
>>──ALTER member-name ┬──────────────┬──┬─ ; ─┬──────────────────────────────────────><
                      └─NUMBER n─────┘  └─ . ─┘
>──┬──────────────────────────────────────────┬──┬──────────────────────────────┬──┬─ ; ─┬──>
  └─RENUMBER ──┬──────────────┬──┘  ├─NOPRINT ───┤  └─ . ─┘
               └─integer ─────┘    ├─NO-PRINT ──┤
                                   ├─PRINT ─────┤
                                   └─WIPE-OUT ──┘
>──┬──────────────────────────────────────────┬──────────────────────────────────────>
   │  <<<<<<<<<<<<<<<                         │
   └─amendment-line ──────────────────────────┘
>──┬─ ACCEPT ─┬──────────────────────────────────────────────────────────────────────><
   ├─ QUIT ───┤
   ├─ END ────┤
   ├─ ; ──────┤
   └─ . ──────┘
```

where:

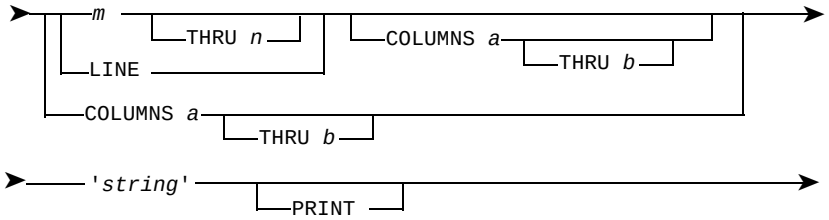
member-name is the name of a dictionary member

integer is an unsigned integer in the range 1 to 10000

amendment-line is:

```
┌─┬─┬─ m ──────────────────────────────────────────────────────────────┐
│ │ │   LINE ──┬──────────────────────────────────────────────────────┤
│ │ │          └── source-line ──┘                                    │
│ ├─DELETE ─┬─ LINE ──────────────────────────────────────────────────┤
│ │         │ m ──┬──────────────────────────────────────────────────┤
│ │         │     │ THRU ──┬──────────────────────────────────────────┤
│ │         │     │        │                                          │
│ │         │     │ TO ────┘                                          │
│ │         │     └──────────────────────────────────────── n ────┬───┤
│ │         │                                                     │   │
│ ├─MOVE ───┬─ LINE ──┬──────────────────────────────────────────┬──┤
│ │ │       │         │                                          │  │
│ │ │       │         │                                          │  │
│ │ │       │         │                                          │  │
```

fill-amendment-line is:

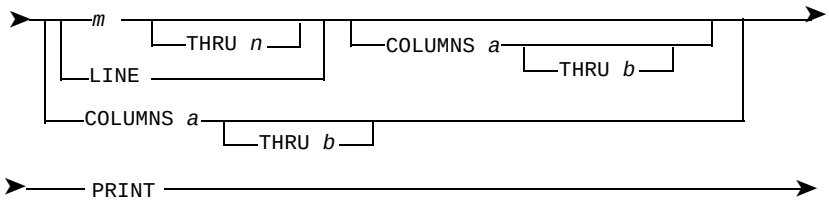


where:

m, *n* and *string* are as defined above.

a and *b* are character positions in a source record line.

blank-amendment-line is:



where:

m, *n*, *a* and *b* are as defined above.

i is an integer of increment.

AMEND (CMR-CM01)

The AMEND command is used to change, add or delete, by name, the keywords and clauses of a member's encoded record.

Refer to "AMEND Syntax" on page 77 for the syntax of the AMEND command.

Introduction to the AMEND Command

The AMEND command enables you to amend, by name, the keywords and clauses in a member's encoded record.

The encoded record of the amended member is always displayed on completion of the command. You can also display the member during an amendment, using the DISPLAY keyword. This can be included any number of times in the command.

You can safely test a particular AMEND command for correctness, without encoding the results, by using the QUIT keyword.

By default, an AMEND command that contains errors is rejected, but you can force correct amendments to be accepted by using the ACCEPT keyword.

The AMEND command can be issued in batch or online. It can be used to amend a large number of members by executing a PERFORM command on a selection of members.

In order to AMEND a member, you must be in an update status, the member must not be locked by another user, and your security level must be sufficient for you to alter the member.

If you do not have sufficient authority to access all the members directly referred to by the member being amended, its source record will be amended but not encoded.

Note:

- Characters entered in the command area are converted to upper-case. If you wish to change lower-case characters with an AMEND command, you must execute the command in a buffer.
- The keyword and clauses of an amended member are reordered in ASG-defined order, which may not be the same as the previous order.

Adding a Keyword or Clause

To add a new keyword or clause to a member, enter:

```
AMEND member-name ADD clause-name value;
```

where:

member-name is the name of an encoded member in the repository.

clause-name is the name of a keyword or clause that does not currently exist in the member's definition.

value is the value of the clause.

For example:

```
AMEND ADMIN ADD CATALOG "PAYROLL";
```

adds a CATALOG of PAYROLL to the member ADMIN.

Deleting a Keyword or Clause

To delete a keyword or clause from a member, enter:

```
AMEND member-name DELETE clause-name;
```

where:

member-name is the name of an encoded member in the repository.

clause-name is the name of a keyword or clause of the member.

For example:

```
AMEND PAYROLL DELETE NOTE;
```

deletes the NOTE clause from the member PAYROLL.

You can also delete part of a clause. For example:

```
AMEND ADMIN DELETE FIRST ALIAS;
```

deletes the first alias of the member ADMIN.

Changing Text in a Clause

To change all occurrences of a particular string of text in a clause of a member, enter:

```
AMEND member-name CHANGE clause-name ALL "string1" TO "string2";
```

where:

member-name is the name of an encoded member in the repository.

clause-name is the name of a text or free-form text clause of the member.

string1 is the string to be changed.

string2 is the replacement string.

For example:

```
AMEND ADMIN CHANGE ALL NOTE "0" TO "9";
```

changes to “9” all the occurrences of the character “0” in the NOTE clause of the member ADMIN. For example, a NOTE value of “000” would be changed to “999”.

To change only the first occurrence of the string in each line of the clause, omit the ALL keyword. For example:

```
AMEND ADMIN CHANGE NOTE "0" TO "9";
```

changes a NOTE value of "000" to "900".

You can also restrict the change to specific lines. For example:

```
AMEND ADMIN FIRST 5 DESCRIPTION CHANGE ALL "MSP" TO "EHDL";
```

changes to "EHDL" occurrences of "MSP" in the first five delimited strings of the DESCRIPTION clause of member ADMIN.

Appending to a Clause

To append to the end of a clause of a member, enter:

AMEND *member-name* APPEND *clause-name addition*;

where:

member-name is the name of an encoded member in the repository.

clause-name is the name of a repeating clause or a text clause of the member.

addition is the text, string or value to be added.

For example:

```
AMEND ADMIN APPEND NOTE "END";
```

adds the text "END" to the end of the NOTE clause of the member ADMIN.

You can append to any part of a clause. For example, to insert a catalog of "SIXTH" after the fifth catalog of the member MIDDLE, enter:

```
AMEND MIDDLE APPEND OCCURRENCE 5 CATALOG "SIXTH";
```

You can insert a value as the first line of a clause, by specifying an OCCURRENCE of 0.

You can only append to a repeating clause or to a text clause. For example, if you try to append to the EFFECTIVE-DATE clause of the member ADMIN by entering:

```
AMEND ADMIN APPEND EFFECTIVE-DATE 12/02/91;
```

an error message displays and processing is aborted.

There are two types of repeating clause. The first comprises a keyword followed by a list of values, for example:

```
ALIAS 'ARTICLE-NUMBER', 'PART-NUMBER'
```

The second allows the repetition of the entire clause, for example:

```
SELECT 1 MSP0000
SELECT 2 DMRWHI00
```

A text clause can contain many lines of text, either undelimited (free-form) or delimited.

You can use the SHOW MEMBER-TYPE command to show which clauses of a particular member type are text or repeating clauses.

Replacing a Clause

To replace a clause of a member, enter:

AMEND *member-name* REPLACE *clause-name value*;

where:

member-name is the name of an encoded member in the repository.

clause-name is the name of the clause to be replaced.

value is the value of the clause.

For example:

```
AMEND ADMIN REPLACE ALIAS "SOLITARY";
```

replaces all of the ALIAS clause in the member ADMIN with an ALIAS of SOLITARY.

You can also replace parts of a clause. For example:

```
AMEND ADMIN REPLACE FIRST ALIAS "ONE";
```

replaces the first alias of the member ADMIN with the alias ONE.

Amending Part of a Clause

To amend particular lines of a clause in a member, enter:

AMEND *member-name amendment-type line clause-name value*;

where:

member-name is the name of an encoded member in the repository.

amendment-type is one of these keywords:

- DELETE
- APPEND
- REPLACE
- CHANGE

line is one of the following:

- FIRST
- FIRST *n* *
- OCCURRENCE *n* TO *m* *
- OCCURRENCE *n*
- LAST
- LAST *n* *

Note:

The keywords marked * are not permitted with the APPEND keyword.

n and *m* are integers indicating lines or occurrences of the clause.

m can be END.

The OCCURRENCE keyword can be replaced by the alternative keyword LINE, a more accurate description when the values of an clause appear on different lines, for example:

```
NOTE 'ONE'
   , 'TWO'
   , 'THREE'
```

as opposed to:

```
NOTE 'ONE', 'TWO', 'THREE'
```

clause-name is the name of a clause.

value is the value of the clause (only specified with the APPEND and REPLACE keywords).

For example:

```
AMEND PARTS DELETE FIRST 2 CATALOG;
```

deletes the first two values in the CATALOG clause of the member PARTS.

```
AMEND PARTS APPEND LINE 3 NOTE "FOURTH";
```

inserts the string “FOURTH” after the third value in the NOTE clause of the member PARTS.

Displaying an Amended Member

You can display the amended member definition at any point during an amendment, using the DISPLAY keyword. No headings or line numbers display. For example:

```
AMEND ADMIN DISPLAY DELETE NOTE DISPLAY ADD CATALOG "1";
```

displays the record of the member ADMIN before and after the deletion of the NOTE clause. On completion of the command, the amended member is printed, showing the addition of a CATALOG clause and the deletion of the NOTE clause.

Performing a Dummy Amendment

You can abort the amendment of a member, cancelling any updates, by terminating the AMEND command with the QUIT keyword. This enables you to test an AMEND command for correctness, without implementing any changes. For example:

```
AMEND ADMIN ADD SIZE 100 KBYTES QUIT;
```

tests whether the given SIZE clause could be added to the member ADMIN. An error message displays if ADMIN is not a FILE member, since only a FILE member can contain a SIZE clause.

You can obtain more information about the success of a given AMEND command by including a DISPLAY keyword after each amendment. For example:

```
AMEND ADMIN ADD SIZE 100 KBYTES DISPLAY QUIT;
```

displays the result of trying to add the SIZE clause, and then aborts the amendment.

Ignoring Erroneous Amendments

By default, an AMEND command that contains erroneous amendments (such as the deletion of a clause that is not allowed in the syntax of the member) is aborted with no amendments being made. However, if you terminate the command with the ACCEPT keyword, erroneous amendments are ignored, and correct amendments are processed. For example:

```
AMEND ADMIN DELETE VATALOG APPEND NOTE "OK" ACCEPT;
```

accepts the correct appending of the NOTE clause, but rejects the deletion of the non-existent VATALOG clause. Without the ACCEPT keyword, both amendments would have been rejected.

Making the Same Amendment to Many Members

You can amend many members identically, using the PERFORM command. For example:

```
PERFORM 'AMEND * DELETE NOTE' ITEMS;
```

removes the NOTE clause from all ITEM members that you can see from your current status.

```
PERFORM 'AMEND * REPLACE EFFECTIVE-DATE 02/12/91' KEPT-DATA;
```

changes to 02/12/91 the EFFECTIVE-DATE clause of all members in the unnamed KEPT-DATA list.

Amending Free-Form Text Clauses

To amend a *free-form text* clause you can either:

- Enter the command in the command area, with line-end characters to marking the end of each line of text, or
- Execute the command in a buffer, which requires no line-end characters. The lines of text in the buffer correspond to the lines of free-form text in the member.

Since the default is for no line-end character, you must set the line-end character, using the SET LINE-END command, before entering free-form text amendments in the command area.

Assuming a line-end character set to "@", you would enter:

```
AMEND ADMIN REPLACE FIRST CONTENTS @FIRST LINE@SECOND LINE@;
```

to replace the current free-form text in the CONTENTS clause of the member ADMIN with two lines of text, FIRST LINE and SECOND LINE.

The free-form text amendment must appear at the end of the command, so that it is terminated by the command-end character ";" as the first character of the line.

AMEND Command Examples

To add a CATALOG clause containing two values, MANAGER and NEW, to the member PAYROLL, enter:

```
AMEND PAYROLL ADD CATALOG 'MANAGER', 'NEW';
```

To append to the CONTAINS clause of the GROUP member named ORDER:

```
GROUP
CONTAINS
  ARTICLE-NO,
  CUST-NO
;
```

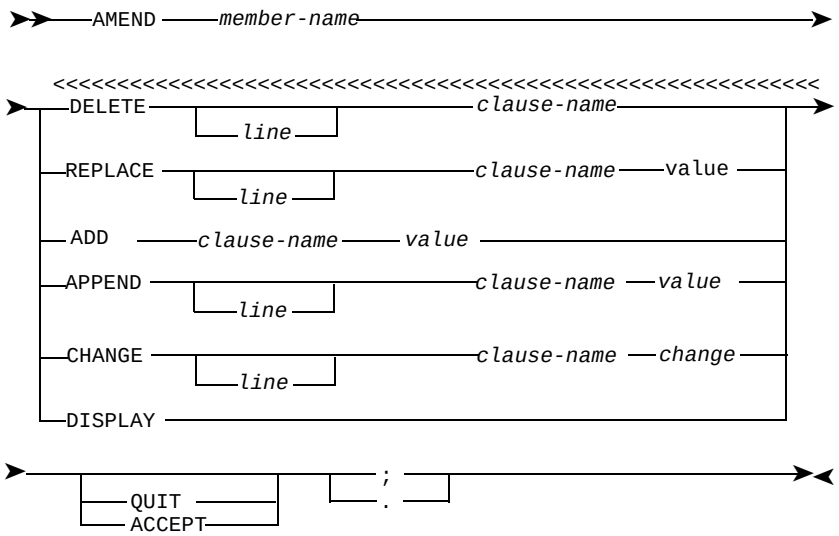
enter:

```
AMEND ORDER APPEND CONTAINS SUPPL-ORDER-NO IF REC-TYPE < 20
                                        ELSE PROD-ORDER-NO ;
```

resulting in this definition:

```
GROUP
CONTAINS
  ARTICLE-NO,
  CUST-NO,
  SUPPL-ORDER-NO IF REC-TYPE < 20
  ELSE PROD-ORDER-NO
;
```

AMEND Syntax



where:

member-name is the name of a member in the repository

line is:

```
➤──┬─FIRST──┬──┬──────────────────────────────────────────────────────────────────────────────►
   │        │  │                                                                                │
   │        │  │                                                                                │
   │        │  │                                                                                │
```

The AUTHORITY command remains in effect until you give another AUTHORITY or DICTIONARY command, or end the session of Manager Products.

Because passwords control access to the dictionary, you should comply with any procedures set up by the dictionary Controller to ensure the privacy of passwords.

You can prevent your password being printed:

- When working in batch, by entering it on a different line from the AUTHORITY command
- When working interactively, by switching the terminal to full-duplex operation. (For details, refer to the documentation supplied with your terminal, or ask the person responsible for the maintenance of your terminal).

For further security, the AUTHORITY command is suppressed from QUERY RECALL-BUFFER command output and from the redisplay of previously-issued commands invoked by pressing the RECALL PF key, if one is assigned. (For further information, refer to *ASG-ControlManager User's Guide*.)

If you use the Master Operator's password, Master Operator commands are the only commands accepted. To use any other command you must first do one of these:

- Enter a further AUTHORITY command
- Enter a DICTIONARY command and an AUTHORITY command
- End the session of Manager Products

If the Audit and Security facility (selectable unit CMR-DD3) is installed, the security level assigned to the password by the dictionary Controller applies to all commands. The security level remains in effect until you do one of the following:

- Enter a further AUTHORITY command
- Enter a DICTIONARY command and an AUTHORITY command
- End the session of Manager Products

Refer to "Security Levels" on page 18 for details of security levels.

Refer to *ASG-Manager Products Controller's Manual* for details of password and of Master Operator commands.

AUTHORITY Command Syntax

```
>>─┬─AUTHORITY ──────────┬─password──────┬─;─┬──────────────────><
   └─AU ─────────────────┘               └─.─┘
```

where *password* is a character string of up to eight printable or non-printable characters, being the identification issued by the Controller to the particular user or to the Master Operator. This must be delimited if it contains characters from the extended character set.

BULK ENCODE (DMR-DD1)

The BULK ENCODE command encodes all or selected members of the repository.

Refer to "BULK ENCODE Syntax" on page 88 for the syntax of the BULK command.

Introduction to the BULK ENCODE Command

The BULK ENCODE command is used to encode all or a selection of repository members.

Encoding normally consists of syntax checking. A member encodes successfully if it passes this check.

If you specify the CHECK keyword encoding consists of:

- Syntax checking
- Integrity checking

In this case a member only encodes successfully if it passes both these checks.

To encode all members that have source records in the current status, enter:

```
BULK ENCODE ;
```

Note:

Do *not* use this form of the command unless you really need to, as the command can take a long time to process if your repository contains a large number of members.

Alternatively, you can encode a selection of members based on one or more of these:

- LOCKED: currently locked members
- KEPT-DATA: members kept in a KEPT-DATA list
- MEMBERS: a list of specific member names
- SOURCES: members' source records
- One or more specific member types
- EXCEPT: to exclude specific member types
- DUMMY, DUMMYS, DUMMIES: dummy members
- REAL: encoding non-dummy members, that is, those that have been defined
- GENERIC: member types based on a member type
- CHECK-OK, CHECK-NEEDED: members that have passed or need integrity checks

You can further refine this selection by specifying one or more of these secondary selections:

- Name: selecting members whose names meet given criteria
- Status: selecting members that exist, or were processed, in specific statuses, or members whose source and encoded records are out of step
- Time: selecting members that were, or were not, processed at a given time or during a given period of time
- User: selecting members that were, or were not, processed by a certain user

Note:

The keyword OLD-DATE is only available to the repository Controller.

When you use the BULK ENCODE command, only source members in the current status are encoded. The current status must be an update status.

You can only encode a member if your security level is sufficient for you to alter that member and to access all members directly used by that member.

BULK ENCODE is treated as an updating command by the recovery system, which operates individually in respect of each member processed.

Refer to "LOCK (CMR-WS01)" on page 173 for details of member locking.

Refer to *ASG-Manager Products Status Concepts* for details of encoding members in a status environment.

Refer to "Security Levels" on page 18 for details of security levels.

Checking Syntax and Integrity

To check syntax and integrity, enter:

```
BULK CHECK ENCODE ;
```

Encoding is only successful if a member passes both checks.

Refer to *ASG-MethodManager Administration* for details of integrity checking.

Encoding Locked Members

To encode all locked members with a source record, enter:

```
BULK ENCODE LOCKED ;
```

You must enter the LOCKED keyword before any other selection criteria in your BULK ENCODE command.

You cannot encode any members that have been locked by another user.

A member needs an existing source record in order to be processed by a BULK ENCODE command. Hence, you cannot encode any member that has been locked with a LOCK NEW command.

Refer to "LOCK (CMR-WS01)" on page 173 for details of member locking.

Encoding Members in a KEPT-DATA List

To encode members in an unnamed KEPT-DATA list, enter:

```
BULK ENCODE KEPT-DATA ;
```

To encode members in a named KEPT-DATA list, enter:

BULK ENCODE KEPT-DATA IN *list-name* ;

list-name is the name of a particular KEPT-DATA list.

Members in a KEPT-DATA list are processed in the order in which they appear in the KEPT-DATA list, unless you specify the keyword ALPHABETICALLY in the command, in which case the members are processed in alphanumerical order.

For example:

```
BULK ENCODE KEPT-DATA IN ADDRESS ALPHABETICALLY ;
```

processes the members kept in a KEPT-DATA list named ADDRESS in alphanumerical order.

If you specify KEPT-DATA and MEMBERS in your BULK ENCODE command, then any members that fall within either category are encoded. Members that appear in both categories are processed only once.

If you include any other selection categories in your BULK ENCODE command, only those members which satisfy all of the included categories are encoded.

Encoding Particular Members

To encode one or more named members, enter:

```
BULK ENCODE MEMBERS member-name-list ;
```

where *member-name-list* is one or more member names separated by commas.

Members in a list of member names are processed in the order they are specified in the *member-name-list*, unless you specify the keyword ALPHABETICALLY in the command in which case the members are processed in alphanumerical order.

For example:

```
BULK ENCODE MEMBERS EMP-NAME, ACC-RPT ALPHABETICALLY ;
```

processes the member ACC-RPT first, then EMP-NAME.

If you specify MEMBERS and KEPT-DATA in your BULK ENCODE command, then any members that fall within either category are encoded. Members that appear in both categories are processed only once.

If you include any other selection categories in your BULK ENCODE command, only those members which satisfy all of the included categories are encoded.

Encoding Unencoded Source Records

To encode all members with a source but no encoded record in the current status, enter:

```
BULK ENCODE SOURCES ;
```

You can also use the BULK ENCODE SOURCES command to encode source members that have a dummy data entries record.

Encoding Dummy Members

To encode all members with a dummy data entries record and a source record in the current status, enter:

```
BULK ENCODE DUMMIES ;
```

DUMMIES can be replaced with the alternative keyword DUMMYS.

To encode all dummy members of a given member type enter:

```
BULK ENCODE DUMMY member-type-list ;
```

where *member-type-list* is the interrogate keyword for any member type, or any collective member type, available in your repository. If more than one member type is specified each must be separated by commas.

Use the SHOW UDS command to find out which member types are available in your repository.

To encode all dummy members, except those with a member type of either PROGRAM or SYSTEM, enter:

```
BULK ENCODE DUMMIES EXCEPT PROGRAMS, SYSTEMS ;
```

Refer to "Collective Member Types" on page 316 for details of collective member types.

Encoding Non-Dummy Members

To encode all non-dummy members, that is, those members that have been defined, enter:

```
BULK ENCODE REAL ;
```

The REAL keyword may be used in conjunction with other keywords. For example, to encode all non-dummy members of one or more specific types, enter:

```
BULK ENCODE REAL member-type-list ;
```

where *member-type-list* is the interrogate keyword for any member type, or any collective member type, available in your repository. When more than one member type is specified each must be separated by a comma.

Encoding Specific Member Types

To encode members of a specific member type, enter:

```
BULK ENCODE member-type-list ;
```

where *member-type-list* is the interrogate keyword for any member type, or any collective member type, available in your repository. When more than one member type is specified each must be separated by a comma.

For example, to encode only locked members of the types GROUP and ITEM, enter:

```
BULK ENCODE LOCKED GROUPS, ITEMS ;
```

To encode all members, except those of a specified member type, enter:

```
BULK ENCODE EXCEPT member-type-list ;
```

For example, to encode all members except those with either a member type of PROGRAM or SYSTEM, enter:

```
BULK ENCODE EXCEPT PROGRAMS, SYSTEMS ;
```

Use the SHOW UDS command to find out which member types are available in your repository.

Refer to "Collective Member Types" on page 316 for details of collective member types.

Encoding Members in Alphanumerical Order

To process all members selected in a BULK command in alphanumerical order, enter:

command selection ALPHABETICALLY ;

where:

command is a BULK command

selection is any of the selections available with the BULK command.

Encoding Members in a Status Environment

You can only encode a member if it has a source record in the current status and the status is not read-only.

If you attempt to encode a member in a read-only status, this message displays:

```
DM00093S COMMAND NOT ALLOWED IN A READ-ONLY STATUS
```

and the ENCODE command is rejected.

If there is no source record in the update status, this message displays:

```
DM01290E member-name NO SOURCE PRESENT
```

An ENCODE command is rejected even if a source record exists in another status.

If, as a result of a member being encoded, a dummy data entries record is created, it is created in the current status.

Changes to used-by tables are also made in the current status. If there is no used-by table for the relevant member in the current status, Manager Products looks first in the direct base status and then in each indirect base status in turn, until the root status is reached. The first used-by table it finds is copied to the current status, and the changes made to it.

If the encoding of a member results in a change of its member type in the current status as compared with its member type in the direct base status, this change is reflected in the uses tables in the current status of all the members that use or are used by it.

Status, Name, Time and User-Related-Selections

Refer to "Applying Commands to a Selection Of Index-names Based on Name, Status, Time and User" on page 293 for details of the name, status, time and user-related-selections.

Encoding Members Based on a Specific Member Type

To encode members of types based on a specific member type, enter:

```
BULK ENCODE GENERIC member-type ;
```

where *member-type* is the interrogate keyword of any base member type.

Encoding Members That Have Passed or Need Integrity Checks

To select check-ok members, members that have passed integrity checks, include the CHECK-OK keyword in the command.

To select check-needed members, members that need integrity checks, include the CHECK-NEEDED keyword in the command.

For further details of check-ok and check-needed members refer to *ASG-MethodManager Administration*.

Preserving Previous Encode Details

Only the repository Controller can preserve previous encode details using the OLD-DATE keyword.

You may have to make a global change to repository members, due to:

- Changes to the repository information model
- Restoring members' source records from a SAVE tape

When you encode these members your identity, that is MASTER, and the date and time that you encode the members is recorded, overwriting previous details.

To preserve the existing user, date and time encode details when you encode members, enter:

```
BULK ENCODE OLD-DATE ;
```

BULK ENCODE Command Examples

To encode members in an unnamed KEPT-DATA list in alphanumerical order, enter:

```
BULK ENCODE KEPT-DATA ALPHABETICALLY ;
```

To encode members in the KEPT-DATA list named ADDRESS and the members RPT-PAGE and RPT-TITLE, enter:

```
BULK ENCODE KEPT-DATA IN ADDRESS MEMBERS RPT-PAGE, RPT-TITLE ;
```

To encode the members DEV-RPT1 and DEV-RPT2, enter:

```
BULK ENCODE MEMBERS DEV-RPT1, DEV-RPT2 ;
```

To encode all currently locked members, enter:

```
BULK ENCODE LOCKED ;
```

To encode all members of the member type ITEM or GROUP and preserve the existing user, date and time details for each member, enter:

```
BULK ENCODE ITEMS, GROUPS OLD-DATE ;
```

To encode all members excluding those of the member type GROUP, enter:

```
BULK ENCODE EXCEPT GROUPS ;
```

To encode all dummy members, enter:

```
BULK ENCODE DUMMIES ;
```

To encode all members whose names begin with the letter A, enter:

```
BULK ENCODE ONLY A ;
```

To check syntax and integrity of all members that have a source record but not an encoded record, enter:

```
BULK CHECK ENCODE SOURCE ;
```

Encoding is only successful if a member passes both checks.

To encode members of types based on the member type FILE, enter:

```
BULK ENCODE GENERIC FILES ;
```

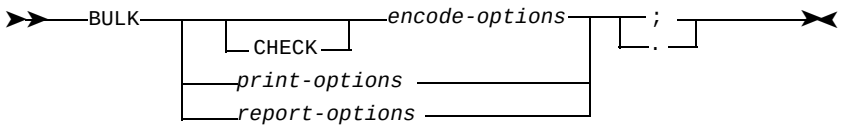
To retain the user, date and time information of all locked members when you encode them, enter:

```
BULK ENCODE LOCKED OLD-DATE ;
```

To retain the user, date and time information of all members with names beginning with TAX when you encode them, enter:

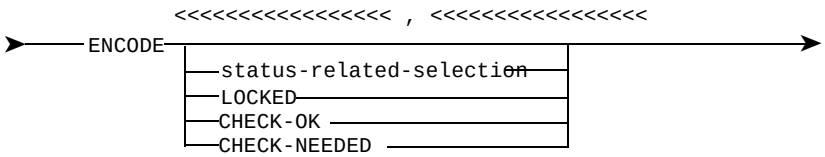
```
BULK ENCODE OLD-DATE ONLY TAX ;
```

BULK ENCODE Syntax



where:

encode-options is:



```
──┬──────────┬──┬──────────┬──┬──────────────┬──────────────────────────►
  │selection │  │OLD-DATE  │  │ALPHABETICALLY│
  └──────────┘  └──────────┘  └──────────────┘

──┬──────────────────────────┬──────────────────────────────────────────►
  │name-related-selection    │
  └──────────────────────────┘

──┬──────────────────────────────────────────┬──────────────────────────►
  │time-and-user-related-selection           │
  └──────────────────────────────────────────┘
```

Note:

The keyword OLD-DATE is restricted to the Controller.

where *selection* is:

```
──┬──────────────────────────────────────────────────────────────────►
  │KEPT-DATA──────────────────────────────────────────────────────────┐
  │                   ┌──────────────────────┐                        │
  │                   │IN list-name          │                        │
  └───────────────────┴──────────────────────┘                        ┘

──┬──────────────────────────────────────────────────────────────────►
  │          <<<< , <<<<                                              │
  │MEMBERS member-name                                                │
  └───────────────────────────────────────────────────────────────────┘

──┬──────────────────────────────────────────────────────────────────►
  │                          <<< , <<<                                │
  │──────────────────────────type-name────────────────────────────────┐
  │   ┌──────┐                                                        │
  │   │DUMMY │                                                        │
  │   │EXCEPT│                                                        │
  │   │REAL  │                                                        │
  │                                                                   │
  │──────DUMMYS───────────────────────────────────────────────────────┐
  │   ┌───────┐   ┌──────────────────────────────┐                    │
  │   │DUMMIES│   │          <<< , <<<           │                    │
  │   │REAL   │   │EXCEPTS-type-name             │                    │
  │                                                                   │
  │SOURCES────────────────────────────────────────────────────────────┘
```

where:

list-name is the name of a KEPT-DATA list

member-name is the name of a repository member

type-name is:

```
──┬──────────┬──────────────────────────────────────────────────────►
  │          │                 member-type
  │GENERIC   │
  └──────────┘
```

where:

member-type is the interrogate keyword for any member type, or any collective member type, available in your repository.

Use the SHOW UDS command to find out which member types are available in your repository.

status, *name*, *time* and *user-related-selection* are defined in "Applying Commands to a Selection Of Index-names Based on Name, Status, Time and User" on page 293.

print-options is:

```
PRINT ──┬──────────────┬──┬──────────────────────────────┬──
        │ALL-STATUSES  │  │ENCODED-SOURCE ┬──────────┬   │
        └──────────────┘  │               │ALSO      │   │
                          │               └──────────┘   │

──┬──┬──────────────────────┬──┬──────────┬──┬──────────────────────────┬──
  │  │status-related selection│  │selection │  │                         │
  │  │LOCKED                │  └──────────┘  │                         │
  │  │CHECK-ON              │                │                         │
  │  │CHECK-NEEDED          │                │                         │

──┬──────────────┬──┬──────────────────────────┬──
  │ALPHABETICALLY│  │name-related-selection    │
  └──────────────┘  └──────────────────────────┘

──┬──────────────────────────────────────────────┬──
  │time-and-user-related-selection               │
  └──────────────────────────────────────────────┘
```

where:

selection is defined above

status, *name*, *time* and *user-related-selection* are defined above.

report-options is:

```
REPORT ──┬──────────────┬──┬──────────────┬──
         │HIERARCHY     │  │ALL-STATUSES  │
         └──────────────┘  └──────────────┘

──┬──────┬──┬──┬──────────────────────────┬──
  │SKIP  │  │  │status-related-selection │
  │NOSKIP│  │  │LOCKED                   │
  │NO-SKIP│  │  │CHECK-OK                 │
  └──────┘  │  │CHECK-NEEDED             │
            └──┴─────────────────────────┘

──┬──────────┬──┬──────────────┬──
  │selection │  │ALPHABETICALLY│
  └──────────┘  └──────────────┘

──┬──────────────────────────────┬──
  │name-related-selection        │
  └──────────────────────────────┘

──┬──────────────────────────────────┬──
  │time-and-user-related-selection   │
  └──────────────────────────────────┘

──┬──────────────────────┬──
  │DOWN-TO member-type   │
  └──────────────────────┘
```

Note:

The keyword SOURCES is not valid with BULK REPORT.

where:

selection is defined above

status, *name*, *time* and *user-related-selection* are defined in "Applying Commands to a Selection Of Index-names Based on Name, Status, Time and User" on page 293.

BULK PRINT (DMR-DD1)

The BULK PRINT command prints the source records of all, or a selection of, members.

The command is described in the following numbered items:

Refer to "BULK ENCODE Syntax" on page 88 for the syntax of the BULK command.

Introduction to the BULK PRINT Command

Any members you want to print must have a source record in the repository.

You can print the source records of all members from the current status, by entering:

```
BULK PRINT ;
```

Do not use this form of the command unless you really need to, as processing can be very time-consuming and the output lengthy.

Alternatively, you can print a selection of members' source records based on one or more of the following:

- LOCKED: currently locked members; and/or
- KEPT-DATA: members kept in a KEPT-DATA list; and/or
- MEMBERS: a list of specific member names
- One or more specific member types
- EXCEPT: to exclude specific member types
- DUMMY, DUMMIES: dummy members
- REAL: encoding non-dummy members, that is, those that have been defined
- CHECK-OK, CHECK-NEEDED: members that have passed or need integrity checks
- GENERIC: members based on a member type.

You can further refine this selection by specifying one or more of the following secondary selections:

- Name: selecting members whose names meet given criteria
- Status: selecting members that exist, or were processed, in specific statuses
- Time: selecting members that were, or were not, processed at a given time or during a given period of time
- User: selecting members that were, or were not, processed by a certain user.

Having decided which members you want to print, you can include the following keywords in your BULK PRINT command to define the type of information you require:

- ALL-STATUSES to apply the command to all the statuses currently available to you and/or
- ENCODED-SOURCE to print source records of the members as they stood at the time the members were last successfully encoded.

If you include ENCODED-SOURCE and/or ALL STATUSES in your BULK PRINT command, you can enter them in any order but in relation to the rest of the command they must be entered between the PRINT keyword and any selection criteria.

Output from the BULK PRINT command is affected by the operation of the SET PRINT-DELIMITER and SET PRINT-SHIFT commands.

The security facility controls access to information held in the repository. For details of how security affects interrogations refer to "Access to the Dictionary" on page 17.

Refer to *ASG-Manager Products Status Concepts* for details of the Status facilities.

Printing Members in a Status Environment

Unless you specify otherwise, when you print a member, the member is printed from the current status or, if the member does not have a source record in the current status, from the nearest base status that contains a source record of the member.

If you include the ALL-STATUSES keyword in your command, the member's source record is printed from every status that contains a source record of the member and that you have access to.

For example, if you have a four-status repository which contains:

- A member named ACC-NO with a source record in two statuses, and
- A member named ACC-NAME with a source record in each of the four statuses.

and you want to output a copy of each of these two members from every status in which they appear, enter:

```
PRINT ACC-NO, ACC-NAME ALL-STATUSES ;
```

or

```
BULK PRINT ALL-STATUSES MEMBERS ACC-NO, ACC-NAME ;
```

In this example, two separate versions of ACC NO and four separate versions of ACC-NAME would be output (assuming you have access to all four statuses in your repository).

Printing a Member's Source Record as it was at the Time it was Last Successfully Encoded

To output a member's source record as it was when the member was last successfully encoded, include the ENCODED-SOURCE keyword.

To output the current source record of a member and its source record as it was when the member was last successfully encoded, include the keywords ENCODED-SOURCE and ALSO.

For example, if the source records of two members named EMP-LIST and EMP-CODE have been updated since they were last encoded, you can compare the two different versions of each of the members' source records by entering:

```
PRINT EMP-LIST, EMP-CODE ENCODED-SOURCE ALSO ;
```

or

```
BULK PRINT ENCODED-SOURCE ALSO MEMBERS EMP-LIST, EMP-CODE ;
```

Printing Locked Members

To print only those members which are locked, enter:

```
BULK PRINT LOCKED ;
```

You must enter the LOCKED keyword before any other selection criteria in your BULK PRINT command.

Refer to "LOCK (CMR-WS01)" on page 173 for details of member locking.

Printing Members in a KEPT-DATA List

To print members in an unnamed KEPT-DATA list, enter:

```
BULK PRINT KEPT-DATA ;
```

To print members in a named KEPT-DATA list, enter:

```
BULK PRINT KEPT-DATA IN list-name ;
```

where *list-name* identifies a particular named KEPT-DATA list.

Members in a KEPT-DATA list are processed in the order in which they appear in the KEPT-DATA list, unless you include the keyword ALPHABETICALLY in your BULK PRINT command, in which case the members are processed in alphanumerical order.

If you specify KEPT-DATA and MEMBERS in your BULK PRINT command, then any members that fall within either category are encoded. Members that appear in both categories are processed only once.

If you include any other selection categories in your BULK PRINT command, only those members which satisfy all of the included categories are printed.

For example:

```
BULK PRINT KEPT-DATA IN NEW-EMP MEMBERS EMP-CODE, EMP-NAME ;
```

prints the member names in the KEPT-DATA list named NEW-EMP followed by the members EMP-CODE and EMP-NAME, but if either of the members EMP-CODE or EMP-NAME also appears in the KEPT-DATA list named NEW-EMP, then they are only printed once.

Printing Particular Members

To print one or more named members, enter:

```
BULK PRINT MEMBERS member-name-list ;
```

where *member-name-list* is one or more member names separated by commas.

Member names in a list of members are processed in the order they are specified in the *member-name-list*, unless you include the keyword ALPHABETICALLY in your BULK PRINT command, in which case the members are processed in alphanumerical order.

For example:

```
BULK PRINT MEMBERS EMP-ADDR, ACC-CODE ALPHABETICALLY ;
```

processes the member ACC-CODE first, then EMP-ADDR.

If you specify MEMBERS and KEPT-DATA in your BULK PRINT command, then any members that fall within either category are encoded. Members that appear in both categories are processed only once.

If you include any other selection categories in your BULK PRINT command, only those members which satisfy all of the included categories are printed.

Printing Source Members

To print all members with a source but no encoded record, enter:

```
BULK PRINT SOURCES ;
```

This command can also be used to print members that have an unencoded source record and a dummy data entries record.

Printing Dummy Members

To print all dummy members, enter:

```
BULK PRINT DUMMIES ;
```

DUMMIES can be replaced with the alternative keyword DUMMYS.

Only those members that have a dummy data entries record and a source record in the current status are printed.

To print all dummy members of a given member type enter:

```
BULK PRINT DUMMY member-type-list ;
```

where *member-type-list* is the interrogate keyword for any member type, or any collective member type, available in your repository. If more than one member type is specified each must be separated by commas.

Use the SHOW UDS command to find out which member types are available in your repository.

You can refine the BULK PRINT DUMMIES variant of the command to exclude specific member types from being printed by using EXCEPT.

For example, the command:

```
BULK PRINT DUMMIES EXCEPT GROUPS, ITEMS ;
```

prints all dummy members, excluding those members with a member type of GROUP or ITEM.

Printing Non-Dummy Members

To print all non-dummy members, that is, all members that have been defined, enter:

```
BULK PRINT REAL ;
```

You may use the REAL keyword in conjunction with other keywords to restrict printing to non-dummy members. For example,

BULK PRINT REAL *member-type-list*;

where *member-type-list* is one or more interrogate keywords for any of the member types, or collective member types, available in your repository. When more than one member type is specified each must be separated by a comma.

Printing Particular Member Types Only

To print only those members of a specified member type, enter:

BULK PRINT *member-type-list* ;

where *member-type-list* is one or more interrogate keywords for any of the member types, or collective member types, available in your repository. When more than one member type is specified each must be separated by a comma.

Use the SHOW UDS command to find out which member types are available in your repository.

For example, to print only locked members which have a member type of either GROUP or ITEM, enter:

```
BULK PRINT LOCKED GROUPS, ITEMS ;
```

To print all members, except those of a specified member type, enter:

```
BULK PRINT EXCEPT member-type-list ;
```

For example, to print all members that have been amended in the current status except those which have a member type of PROGRAM or SYSTEM, enter:

```
BULK PRINT AMENDED EXCEPT PROGRAMS, SYSTEMS ;
```

Processing Members Alphabetically

To process all members selected in a BULK command in alphanumerical order, enter:

command selection ALPHABETICALLY ;

where:

command is a BULK command

selection is any of the selections available with the BULK command.

Format of the Output

By default, print output is preceded by a statement that identifies the member being printed, whether the member is encoded or unverified, and from which status the member's source record has been printed.

An encoded member is preceded by:

```
PRINT OF member-name ENCODED IN STATUS status-name
```

A member with a source record but no encoded record is preceded by:

```
PRINT OF member-name UNVERIFIED IN STATUS status-name
```

For an unverified member, the current source record is preceded by:

```
PRINT OF member-name UNVERIFIED IN STATUS status-name
```

and the source record as it was when the member was last successfully encoded is preceded by:

```
PRINT OF member-name ENCODED IN STATUS status-name
```

Each source record of the member being printed is followed by:

```
END OF PRINT
```

You can suppress these print delimiters by using the SET PRINT-DELIMITERS OFF command.

By default, print output contains leading spaces and source line numbers. However, these can be removed using variants of the SET PRINT-SHIFT command.

If you try to print a member that has no source record, then the message:

```
DM01250E member-name NOT PRESENT ON DICTIONARY
```

is output, and processing continues with the next member name or the next command.

If you try to print the encoded record of a member with a source record but no encoded record, then the message:

```
DM01601E member-name NOT ENCODED
```

is output, and processing continues with the next member name or the next command.

Status, Name, Time and User-Related-Selection

Refer to "Applying Commands to a Selection Of Index-names Based on Name, Status, Time and User" on page 293 for details of the name, status, time and user-related-selections.

Printing Members Based on a Specific Member Type

To print members of types based on a specific member type, enter:

```
BULK PRINT GENERIC member-type ;
```

where *member-type* is the interrogate keyword of any base member type.

Selecting Members That Have Passed or Need Integrity Checks

To select check-ok members, members that have passed integrity checks, include the CHECK-OK keyword in the command.

To select check-needed members, members that need integrity checks, include the CHECK-NEEDED keyword in the command.

For further details of check-ok and check-needed members, refer to *ASG-MethodManager Administration*.

Bulk Print Examples

To print the source records of all members, enter:

```
BULK PRINT ;
```

To print the source records of the members in the KEPT-DATA list named EMP-LIST and the members ACC-TOT and EMP-SAL, enter:

```
BULK PRINT KEPT-DATA IN EMP-LIST MEMBERS ACC-TOT, EMP-SAL ;
```

To print the source records of the members EMP CODE and EMP-NAME from every status in which they exist and to which you have access, enter:

```
BULK PRINT ALL-STATUSES MEMBERS EMP-CODE, EMP-NAME ;
```

To print the source records of all currently locked members, enter:

```
BULK PRINT LOCKED ;
```

To print all dummy members except for those that have a member type of ITEM or GROUP, enter:

```
BULK PRINT DUMMIES EXCEPT ITEMS, GROUPS ;
```

To print all dummy members with a member type of FILE, enter:

```
BULK PRINT DUMMY FILES ;
```

To print the current source record of all members whose names begin with the character string EMP, and which were updated after the 31st July 1990, by the user named PERSONNEL, together with a copy of the source records of those members as they were at the time the members were last successfully encoded, enter:

```
BULK PRINT ENCODED-SOURCE ALSO ONLY EMP IF AMENDED BY PERSONNEL
                                   AFTER '31-JULY-1990' ;
```

To print members in the KEPT-DATA list named ADDRESS in alphanumeric order, enter:

```
BULK PRINT KEPT-DATA IN ADDRESS ALPHABETICALLY ;
```

To print check-needed members, members that need integrity checks, enter:

```
BULK PRINT CHECK-NEEDED ;
```

To print members of a type based on the member type GROUP, enter:

```
BULK PRINT GENERIC GROUPS ;
```

Bulk Print Syntax

Refer to "BULK ENCODE Syntax" on page 88 for the syntax of the BULK command.

BULK REPORT (DMR-DD1)

The BULK REPORT command reports all, or a selection of, encoded members.

Refer to "BULK ENCODE Syntax" on page 88 for the syntax of the BULK command.

Introduction to the BULK REPORT Command

The BULK REPORT command is used to report the encoded records of one or a selection of repository members.

You can report all members in a repository by entering:

```
BULK REPORT ;
```

Do not use this form of the command unless you really need to, as the output will be very lengthy if your repository contains a large number of members.

Alternatively, you can report a selection of members based on one or more of the following:

- LOCKED: currently locked members; and/or
- KEPT-DATA: members kept in a KEPT-DATA list; and/or
- MEMBERS: a list of specific member names
- One or more specific member types
- EXCEPT: to exclude specific member types
- DUMMY, DUMMIES: dummy members
- REAL: encoding non-dummy members, that is, those that have been defined
- CHECK-OK, CHECK-NEEDED: members that have passed or need integrity checks
- GENERIC: members based on a member type.

You can further refine this selection by specifying one or more of the following secondary selections:

- Name: selecting members whose names meet given criteria
- Status: selecting members that exist, or were processed, in specific statuses
- Time: selecting members that were, or were not, processed at a given time or during a given period of time
- User: selecting members that were, or were not, processed by a certain user.

Having decided which members you want to report, you can include one of the following keywords in your BULK REPORT command to define the type of information you require:

- ALL-STATUSES to apply the command to all the statuses currently available to you; and/or
- HIERARCHY to report a selection of members and the members used by these members via an EA relationship CONTAINS
- DOWN-TO to report a selection of members and the members used by these members, via any relationship type, down to a specific level in the member-type hierarchy.

You cannot enter the keywords HIERARCHY and DOWN-TO in the same BULK REPORT command.

You can then define the way the output from your BULK REPORT command is displayed by including one of the following keywords in the command:

- SKIP to insert a page break between each report
- NO-SKIP to insert a blank line between each report.

An alternative layout for the output from a BULK REPORT command can be specified in the FORMAT command.

If you include HIERARCHY and/or ALL-STATUSES and/or SKIP, or NO-SKIP in your BULK REPORT command, you can enter them in any order but in relation to the rest of the command they must be entered between the REPORT keyword and any selection criteria.

The members selected must have a dummy or an encoded record in the current or a visible status.

Unless you specify otherwise, when you use the BULK REPORT command, only those members visible from the current status are reported.

The security facility controls access to information held in the repository.

Reporting a Member and the Members That it Uses Via an EA Relationship Type CONTAINS

To report members and the members those members directly and indirectly use via an EA relationship type CONTAINS, include the HIERARCHY keyword.

The CONTAINS clause must be in a member of one of these types:

- ITEM
- GROUP
- FILE
- TOTAL FILE
- ADABAS FILE
- TOTAL DATABASE.

Reporting a Member in a Status Environment

Unless you specify otherwise, when you use the REPORT command, members are reported from the current status or, if they do not have a data entries record in the current status, from the nearest base status that does contain a data entries record of any of the members being reported.

To report encoded and dummy members in all the statuses currently available to you, include the ALL-STATUSES keyword in your command. A separate report is created for each status.

In the output of a report command, the STATUS line in the report relates to the name of the status from which the report is drawn, not the current status.

If a member exists as a dummy member in the status in which you are working, details are output only if the member has different used-by pointers in that status when compared with the nearest of the base statuses in which it exists (if any).

If a member does not have a data entries record in any status, one of these messages display:

```
DM01601E member-name NOT ENCODED
```

or

```
DM01250E member-name NO SOURCE PRESENT
```

and processing continues with the next member name or the next command.

Reporting Locked Members

To report only those members which are currently locked, enter:

```
BULK REPORT LOCKED ;
```

Any member which has been locked by a LOCK NEW command, but is not encoded, is not reported by a BULK REPORT command.

You must enter the LOCKED keyword before any other selection criteria in your BULK REPORT command.

Reporting Members in a KEPT-DATA List

To report members in an unnamed KEPT-DATA list, enter:

```
BULK REPORT KEPT-DATA ;
```

To report members in a named KEPT-DATA list, enter:

```
BULK REPORT KEPT-DATA IN list-name ;
```

where *`list-name`* is the name of a KEPT-DATA list.

If you specify the keyword ALPHABETICALLY in the command, the member names are processed in alphanumerical order, otherwise the members in the KEPT-DATA list are processed in the order in which they appear in the KEPT-DATA list.

If you specify the selections KEPT-DATA and MEMBERS in your BULK REPORT command, then any members that fall within either category are reported. Members that appear in both categories are reported only once. If you include any other selection categories in your BULK REPORT command, only those members which satisfy all of the included categories are reported.

For example:

```
BULK REPORT KEPT-DATA IN NEW-EMP MEMBERS EMP-CODE, EMP-NO ;
```

reports the member names in the KEPT-DATA list named NEW-EMP followed by the members EMP-CODE and EMP-NO, but if either of the members EMP-CODE or EMP-NO also appears in the KEPT-DATA list named NEW-EMP, then they are reported once.

Reporting Particular Members

To report one or more named members, enter:

```
BULK REPORT MEMBERS member-name-list ;
```

where *member-name-list* is one or more member names separated by commas.

If you specify the keyword ALPHABETICALLY in the command, the members are processed in alphanumerical order, otherwise the member names in the list of members are processed in the order they appear in the member-name-list.

For example:

```
BULK REPORT MEMBERS EMP-ADDR, ACC-CODE ALPHABETICALLY ;
```

processes the member ACC-CODE first, then EMP-ADDR.

If you specify MEMBERS and KEPT-DATA in the command, then any members that fall within either category are reported. Members that appear in both categories are reported only once.

If you include any other selection categories in the command, only those members which satisfy all of the included categories are reported.

For example:

```
BULK REPORT MEMBERS EMP-CODE, EMP-NAME DUMMY ITEMS ;
```

reports the members EMP-CODE and EMP-NAME provided they are dummy members with a member type of ITEM.

Reporting Dummy Members

To report all dummy members, enter:

```
BULK REPORT DUMMIES ;
```

To report all dummy members of a given member type, use the keyword DUMMY as follows:

```
BULK REPORT DUMMY member-type-list ;
```

where *member-type-list* is the interrogate keyword for any member type, or any collective member type, available in your repository. If more than one member type is specified, each must be separated by commas.

Use the SHOW UDS command to find out which member types are available in your repository.

To report all dummy members excluding those of a specific member type, use the keywords DUMMIES and EXCEPT.

For example:

```
BULK REPORT DUMMIES ONLY A EXCEPT GROUPS, ITEMS ;
```

reports all dummy members whose names start with the character string A, excluding those members with a member type of GROUP or ITEM.

Refer to "Collective Member Types" on page 316 for details of collective member types.

Reporting Non-Dummy Members

To report all non-dummy members, that is, all members that have been defined, enter:

```
BULK REPORT REAL ;
```

You may use the REAL keyword in conjunction with other keywords to specify a selection of non-dummy members. For example:

BULK REPORT REAL *member-type-list* ;

where *member-type-list* is the interrogate keyword for any member type, or any collective member type, available in your repository. When more than one member type is specified, each must be separated by a comma.

Reporting Specific Member Types

To select members of one or more member type, enter:

BULK REPORT *member-type-list* ;

where *member-type-list* is the interrogate keyword for any member type, or any collective member type, available in your repository. When more than one member type is specified, each must be separated by a comma.

Use the SHOW UDS command to find out which member types are available in your repository.

For example:

```
BULK REPORT FILES, GROUPS ;
```

reports the encoded record of all members with a member type of FILE or GROUP.

To report all members, except those of a specified member type, enter:

```
BULK REPORT EXCEPT member-type-list ;
```

For example, to report all locked members except for those of the member type GROUP or ITEM, enter:

```
BULK REPORT LOCKED EXCEPT GROUPS, ITEMS ;
```

Processing Members Alphabetically

To process all members selected in a BULK command in alphanumerical order, enter:

command `selection ALPHABETICALLY ;`

where:

command is a BULK command.

selection is any of the selections available with the BULK command.

Inserting a Blank Line or Page Break Between Reports

The first report always commences on a new page. By default, when several members are to be reported, each report is followed by a page break.

To insert a blank line, rather than a page break, after a report of a member, include the NO-SKIP keyword in your command.

To commence a report on a new page and also insert a page break after the report, in the same manner as the default, include the SKIP keyword in your command.

Note:

If you specify more than one member name, the NO-SKIP or SKIP keywords only apply to the member name they immediately follow.

Reporting a Member and Listing the Members That it Uses Via Any Relationship Type

To report a member and list the names and member types of the members it directly and indirectly uses via any relationship type, down to a specific level in the member type hierarchy, include the DOWN-TO keyword.

The member type stated in the DOWN-TO clause must be at an equal or lower level in the hierarchy than the member type of the member you are reporting.

To list the member-type keywords used in the DOWN-TO clause, enter:

```
SHOW UDS MEMBER-TYPES REPORT-DOWN-TO KEYWORDS ;
```

A reference path is not traced beyond a member used by a member at a lower level in the member type hierarchy.

Output for an Encoded Member

These details of an encoded member are reported:

- Its member type
- Its edition number (the number of times it has been encoded)
- The name of the user who last encoded it
- The date and time it was last encoded
- The name of the status from which the report is drawn
- With the Audit and Security facility, the type of protection the member has and the name of its owner
- Information from its clauses
- The number of direct references to it
- The number of direct references from it
- Details appropriate to its member type
- The member type and member name of each member directly used by it via a CONTAINS clause.

For example:

```
REPORT OF GROUP EMP-IDENT
EDITION 2 ENCODED BY MASTER AT 13.09.03 ON 06 MAR 1988
STATUS DEV1
PROTECTION: OWNER,REMOVE,ALTER
OWNED BY PERSONNEL
ALIAS
    EMPLOYEE-IDENTIFICATION-RECORD
CATALOGUED AS
EMPLOYEE
SEE DEPARTMENT
KEYS EMP-NO
USER-EXIT PROJ
DESCRIPTION
    THIS GROUP CONTAINS INFORMATION ABOUT EMPLOYEE'S NAME
THIS MEMBER IS DIRECTLY REFERRED TO ONCE
THIS MEMBER CONTAINS 4 DIRECT REFERENCES
GROUP EMP-IDENT
FORM  HELD-AS
CONTAINS
    ITEM EMP-NAME
    ITEM EMP-ADDR
    GROUP TELEPHONE
```

Output for a Dummy Member

The following details of a dummy member are reported:

- The name of the status from which the report is drawn
- Its member type
- The line: EDITION **DUMMY**
- The name of the user who encoded the member that created it
- The time and date it was created
- The number of direct references to it
- The number of direct references from it.

For example:

```
REPORT OF MODULE DMC28
  EDITION **DUMMY** CREATED BY MASTER AT _12.15.36 ON 09 JUL 1992
  STATUS PROG
  PROTECTION: NONE
  THIS MEMBER IS DIRECTLY REFERRED TO ONCE
  THIS MEMBER CONTAINS 1 DIRECT REFERENCE
```

Name, Status, Time and User-Related Selections

Refer to "Applying Commands to a Selection Of Index-names Based on Name, Status, Time and User" on page 293 for details of the name, status, time and user-related-selections.

Selecting Members That Have Passed or Need Integrity Checks

To select check-ok members, members that have passed integrity checks, include the CHECK-OK keyword in the command.

To select check-needed members, members that need integrity checks, include the CHECK-NEEDED keyword in the command.

For further details of check-ok and check-needed members, refer to *ASG-MethodManager Administration*.

Reporting Members Based on a Specific Member Type

To report members of types based on a specific member type, enter:

```
BULK REPORT GENERIC member-type ;
```

where `member-type` is the interrogate keyword of any base member type.

BULK REPORT Examples

To report all repository members, enter:

```
BULK REPORT ;
```

To report, in alphanumeric order, the members in the KEPT-DATA list ACC-YEAR, and the members ACC-JUL and ACC-AUG, enter:

```
BULK REPORT KEPT-DATA IN ACC-YEAR MEMBERS ACC-JUL, ACC-AUG
ALPHABETICALLY ;
```

To report the members EMP-CODE and EMP-NAME from every status in which they exist and to which you have access, enter:

```
BULK REPORT ALL-STATUSES MEMBERS EMP-CODE, EMP-NAME ;
```

To report all currently locked members, enter:

```
BULK REPORT LOCKED ;
```

To report all dummy members of the member type FILE, enter:

```
BULK REPORT DUMMIES FILE ;
```

To report the encoded records of all members of the type GROUP and also report the encoded records of any members which those GROUP members use, enter:

```
BULK REPORT HIERARCHY GROUP ;
```

To report the members whose names begin with the character string JAN- and which were updated by the user named ACCOUNTS, and insert a page break between each report, enter:

```
BULK REPORT SKIP ONLY JAN- IF AMENDED BY ACCOUNTS ;
```

To report members in the KEPT-DATA list PRODUCTION, excluding those of the member type ITEM, enter:

```
BULK REPORT KEPT-DATA IN PRODUCTION EXCEPT ITEMS ;
```

To report the members named EMP-NAME, EMP-SAL and PROD-NEW, together with a list of the names and member types of any members which these members use, down to members of type FILE, enter:

```
BULK REPORT MEMBERS EMP-NAME, EMP-SAL, PROD-NEW DOWN-TO FILE ;
```

To report check-needed members, members that need integrity checks, enter:

```
BULK REPORT CHECK-NEEDED ;
```

To report members of a type based on the member type GROUP, enter:

```
BULK REPORT GENERIC GROUPS ;
```

BULK REPORT Syntax

Refer to "BULK ENCODE Syntax" on page 88 for the syntax of the BULK command.

CHECK (MMR-PF61)

The CHECK command performs integrity checking as repository members are encoded or removed.

Refer to "CHECK Syntax" on page 112 for the syntax of the CHECK command.

Introduction to the CHECK Command

To execute a command and perform integrity checking, enter:

```
CHECK command ;
```

where *command* is a command that encodes or removes members.

For example:

```
CHECK ENCODE member-name ;
```

where *member-name* is the name of a repository member.

If you do not use the CHECK prefix then the command is executed and impacted members are marked as check-needed. Effectively the integrity checking is postponed.

There are two cases to cover:

- Encoding members
- Removing members

For full details of integrity refer to *ASG-MethodManager Administration*.

You cannot use the CHECK prefix command with these commands:

- AUPD
- AUPDATE
- DCUPD
- PUR
- PURGE

Encoding Members

To encode members and perform integrity checking, enter:

```
CHECK command ;
```

If *command* is the ENCODE command, the encoded record is only altered if the member passes any integrity checks.

If *command* is an encode command other than ENCODE then:

- If the member already exists, its source record is altered but its encoded record is only altered if the member passes any integrity checks
- If the member is new, a source record is created, but an encoded record is only created if the member passes any integrity checks.

To update a member and perform integrity checking, enter:

```
CHECK UPDATE member-name ;
```

where *member-name* is the name of a repository member.

When you have finished updating the member, enter:

```
FILE ;
```

Removing Members

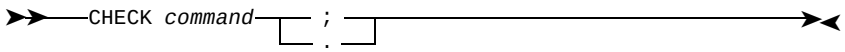
To perform integrity checking as you remove members, enter:

```
CHECK command ;
```

where *command* is either REMOVE or REVERT.

The command has no effect unless all remaining affected members would pass integrity checking after execution of the command.

CHECK Syntax



where *command* is a command that encodes or removes members, except for the following commands:

- AUPD
- AUPDATE
- DCUPD
- PUR
- PURGE

CLOSE Command

The CLOSE command is used to close a dataset external to Manager Products or to close a dictionary. (CMR-CM01)

Refer to "CLOSE Syntax" on page 115 for the syntax of the CLOSE command.

Closing a Dictionary

To close a dictionary that is currently open without opening another dictionary or ending the Manager Products session, enter either:

```
CLOSE ;
```

or

```
CLOSE DICTIONARY ;
```

When the CLOSE command is executed, the contents of any KEPT-DATA lists not stored on the MP-AID are deleted.

Closing a dictionary releases a considerable amount of virtual storage, which can then be used for non-dictionary activity. In environments where virtual storage is limited, this can be useful if you wish you carry out tasks that may require large amounts of virtual storage, such as loading new versions of InfoBank onto the MP-AID.

If you want to close one dictionary and open another, you do not have to enter a CLOSE command; when you enter a DICTIONARY command to open a dictionary, any dictionary that is already open is automatically closed.

To close a dictionary and end the Manager Products session, enter:

```
LOGOFF ;
```

Closing an External Dataset

An external dataset can be created either by Manager Products or software external to Manager Products. The term external is used because, unlike the MP-AID and dictionary datasets, the dataset can be created or accessed directly by software external to Manager Products.

The following Manager Products commands can be used to create external datasets:

- POST/MAIL
- PRODUCE
- SWITCH OUTPUT
- TRANSFER.

You can only process an external dataset after the dataset is closed.

To close an external dataset, enter:

`CLOSE` *`keyword`* `data-set-name ;`

where:

`keyword` is one of: DATASET, FILE or EXTERNAL (these keywords are synonymous)

`data-set-name` is the logical file name (ddname or dtfname) as declared in job control statements and as used to indicate the name of the external dataset (physical file) you wish to CLOSE.

If you generate output to an external dataset using the PRODUCE or the TRANSFER command, several external datasets may be kept open simultaneously. The CLOSE command allows you to close individual datasets, so that they can be used for other functions, while keeping others open. While a dataset is open, any data directed to it is appended to data already held in the dataset. If, once the dataset has been closed, output is directed to a dataset of the same name, the contents are overwritten by the new data.

If you generate output to an external dataset using either the POST/MAIL or the SWITCH OUTPUT command, only one external dataset can be open at a time, since opening a second external dataset automatically closes the first. However, if you wish to close one dataset without opening another, you can use the CLOSE command. Output held in the closed dataset can then be processed when required.

Before closing a dataset opened by a POST/MAIL command, you should first redirect output to another dataset if you wish to retain the contents of the closed dataset. If you close the dataset first, output from any command prefixed with POST/MAIL continues to be directed to the closed dataset, overwriting any data it contained before the CLOSE command was issued.

Before closing a dataset opened by a SWITCH OUTPUT command, you should first SWITCH OUTPUT to another dataset or to MPOUT if you wish to retain the contents. If you close the dataset first, output continues to be directed to the closed dataset, overwriting its contents.

You cannot close external datasets in CICS environments using the CLOSE command.

CLOSE Syntax

```
>>──CLOSE ┬──────────────────────────────┬──┬─ ; ─┬──>>
          ├──DICTIONARY──────────────────┤  └─ . ─┘
          ├──┬─ DATASET ──┬──data-set-name──┤
          ├──┤─ FILE ─────┤                 │
          └──┴─ EXTERNAL──┴                 │
```

where:

data-set-name is the logical file name (ddname or dtfname) as declared in job control statements and as used to indicate the name of the external dataset (physical file) you wish to CLOSE.

The name can be up to a maximum of 8 characters for OS, CMS or BS2000 environments or 7 characters for DOS.

CONTRAST

The CONTRAST command allows the comparison of either the contents of a dictionary member, or the output of a command, with a second member or command in the same or different status.

Any circumstance where the comparative contents of dictionary members or commands have a significance.

If both the selections for comparison are for a MEMBER name then the contents of the encoded member(s) are compared at a clause/attribute level. Specific clauses can be selected or omitted by use of a SELECT/OMIT list (see "CONTRAST Syntax" on page 116).

If both selections for comparison are for command, then the comparison is done on the lines of output from the two commands.

If one selection is for a MEMBER and the other for command, then the comparison is done on the lines of output of an AMEND.DISPLAY command for the MEMBER against the lines of output from the specified command.

Output

Output from the command is in the form:

```
                CONTRASTselection-1
                WITHselection 2

¬    =    SEE
          +    1MMT-ACCESSVIEW
                    -    1          MMT-FILE
          +    2                    MMT-CONCEPTUAL-RECORD
          +    11                   MMT-GOAL
               12    <-   2         MMT-GROUP
               13    <-   3         MMT-ITEM
               14                   MMT-LOCATION
```

- The symbols = = are used to show that the whole comparison is of equal values.
- The symbols ¬ = are used to show that a clause or attribute have different values.
- The symbol + is used to show that the first selection has additional values/lines, or that a clause/attribute is now present.
- The symbol - is used to show that the second selection has additional values/lines, or that a clause/attribute is now missing.
- The symbol <- is used to show that the lines or values are in a different position (the order has been deliberately changed).

If GIVING DETAILS is not specified, then only the significant differences between the members/commands are shown.

If GIVING DETAILS is specified then everything is shown.

- The symbol = is used to show that the line is equal.
- The symbol <= is used to show that the lines or values are now in a different position because of the addition or removal of lines.

CONTRAST Syntax

```
>>──CONTRAST──selection-1──┬──────────────┬──────────────────────────>
                           └──STATUS status──┘
>──WITH ──selection-2──────┬──────────────┬──────────────────────────>
                           └──STATUS status──┘
>──┬──────────────┬──┬──────────────────────────────────────────────><
   └──GIVING DETAILS──┘  │                  <<<<,<<<<
                         ├──SELECT ──┬──clause──┬
                         └──OMIT ────┘
```

where:

selection-1 and *selection-2* is:

```
➤──┬─ MEMBER member-name ─┬──────────────────────────────────➤
   └─COMMAND 'command'────┘
```

CONTRAST Examples

To contrast the NOTE clauses of two members:

```
CONTRAST MEMBER MMG-ONE WITH MEMBER MMG-TWO SELECT NOTE;
```

To contrast the list of MPAID EXEC members on different MPAIDs:

```
CONTRAST COMMAND 'MP LIST EXEC'
WITH     COMMAND 'MP LIST CONCAT MMR2210 EXEC';
```

COPY (CMR-CM01)

The COPY command copies the source record of a member and either inserts that copy as the source record of a new member or inserts that copy as the source record of an existing member (replacing the previous source record).

To copy the source record of an existing member and create the source record of a new member with that copy, enter:

```
COPY member-name-1 TO member-name-2 ;
```

where:

member-name-1 is the name of an existing member.

member-name-2 is the name of the new member.

Member-name-2 must not be the name of a member that already exists in the dictionary and must obey the rules governing member names.

For example, to copy the member MOD-STD-EDIT to a new member named MOD-CFH-EDIT, enter:

```
COPY MOD-STD-EDIT TO MOD-CFH-EDIT ;
```

If a source record already exists for *member-name-2*, the error message:

```
DM01638E member-name-2 SOURCE ALREADY EXISTS
```

is output, and the command is rejected. Processing continues with the next command.

No encoded record is generated from the new source record. To do this, you must either UPDATE and then FILE the member, or ENCODE it.

To copy the source record of an existing member and replace the source record of another member with that copy, enter:

```
COPY member-name-1 TO member-name-2 REPLACE ;
```

A copy of the source record of *member-name-1* replaces the source record of *member-name-2*.

No encoded record is generated from the replaced source record. If that member already has an encoded record it will remain unchanged, and the member will be flagged as unverified; that is, the source record has been changed since the encoded record was generated. To generate a new encoded record to replace the existing one you must either UPDATE and then FILE the member, or ENCODE it.

To copy the source record of a member as it was when it was last encoded successfully, specify `ENCODED-SOURCE`. For example:

```
COPY EMP-CODE ENCODED-SOURCE TO EMP-REC ;
```

ENCODED-SOURCE is useful when you are copying unverified members.

In order to use the COPY command, you must have update access to the dictionary.

If the Audit and Security facility is installed (selectable unit CMR-DD3). You can copy a member only if its access security level is lower than or equal to your access security level. You can create a new member only if the insertion security level of the dictionary is lower than or equal to your insertion security level.

If the Workstation Interface facility is installed (selectable unit CMR-WS01). You cannot copy to or from a member that is locked by another user. If either *member-name-1* or *member-name-2* is locked, the COPY command is rejected and processing continues with the next command.

The automatic error recovery system treats COPY as an updating command.

Refer to *ASG-ControlManager User's Guide* for the member naming rules.

Refer to "LOCK (CMR-WS01)" on page 173 for details of member locking.

Copy Syntax

```
>>---COPY member-name-1 ------------------------------------------>
>---+---TO member-name-2 ---+---FROM status-name ---+-------------->
    |                       |                      |
    +---FROM status-name ---+---TO member-name-2 ---+
>---+---ENCODED-SOURCE---+---+---REPLACE ---+---+---;---+---------><
    |                    |   |             |   |   .   |
```

where:

member-name-1 is the name of the existing dictionary member whose source record is to be copied

member-name-2 is the name of the new dictionary member

status-name is the name of a status.

DICTIONARY (CMR-CM01)

The DICTIONARY command opens a dictionary. Optionally you can specify buffer pool size and monitor buffer steals.

Refer to "DICTIONARY Syntax" on page 125 for the syntax of the DICTIONARY command.

How To Open a Dictionary

To open a dictionary, enter:

```
DICTIONARY dictionary-name ;
```

To use the dictionary, you must follow the DICTIONARY command with a valid AUTHORITY command.

For example, to open and use the DEMO dictionary, enter:

```
DICTIONARY DEMO ;
AUTHORITY ANALYST ;
```

The procedure for opening a dictionary may be tailored at your installation and may differ from this. The Systems Administrator may have set up your Logon Profile so that you automatically access a particular dictionary each time you log on to Manager Products. If so, the following messages:

```
DM01023I dictionary-name DICTIONARY OPEN
DM01060I AUTHORITY RECOGNISED
```

display, and you do not need to enter the DICTIONARY and AUTHORITY commands. These messages display in response to a valid DICTIONARY and AUTHORITY command no matter how they are executed.

Only one dictionary can be open at any one time. When you enter a DICTIONARY command, any dictionary that is already open is closed and any named or unnamed KEPT-DATA lists are deleted, even if the command does not execute successfully.

A dictionary can be opened in either *update* or *read-only mode*, depending on:

- The environment
- The setting of the DCUST macro
- The setting of the SET DICTIONARY-UPDATE command
- Whether non-updating commands and messages are logged
- Whether the keywords READ or UPDATE have been specified in the DICTIONARY command.

If, when you open a dictionary, the message:

```
NO ALTERNATE LOG AREA AVAILABLE
```

displays, inform the dictionary Controller immediately. The alternative area of the log dataset must be archived. If it is not archived and the present area becomes full, the dictionary will become unusable.

If the Basic or Advanced Status facility is installed (selectable units CMR-DD2 and CMR-AD21 respectively) and the newly opened dictionary has a named status, this message displays:

```
DM00095I opening-mode STATUS status-name ASSUMED
```

where:

opening-mode is READ-ONLY or UPDATE.

status-name is the name of the default status determined for the dictionary by the dictionary Controller.

Refer to *ASG-Manager Products Status Concepts* for details of statuses.

Dictionary Opening Modes

A dictionary can be opened in either update or read-only mode.

In *read-only mode*, neither the Recovery nor Log dataset is opened.

Non-updating commands are processed faster if a dictionary is opened in read-only mode.

In *update mode*, the Recovery dataset is automatically opened. If logging is applied, the Log dataset is also opened. If messages or non-updating commands are logged, you must open the dictionary in update mode.

The *open mode* is determined by the value of the *UPDICT keyword of the DCUST installation macro*. The dictionary is opened in update mode if the value is YES and in read-only mode if the value is NO.

In particular circumstances, depending on the environment and whether non-updating commands and/or messages are logged, the default opening mode can be changed by specifying the READ or UPDATE keywords in the DICTIONARY command.

To open a dictionary in read-only mode, enter:

```
DICTIONARY dictionary-name READ ;
```

To open a dictionary in update mode, enter:

```
DICTIONARY dictionary-name UPDATE ;
```

The System Administrator's Environmental Control Facility (selectable unit CMR-SC05) provides the SET DICTIONARY-UPDATE command, which enables the System Administrator to control the mode in which all dictionaries in an installation will be opened for individual, and/or groups of, users.

For details of the DCUST installation macro, refer to the appropriate Manager Products installation manual.

Effects on Opening Modes of the Logging of All Commands and/or Messages

Dictionary opening modes when messages and/or all commands are *not logged*:

	DCUST MACRO SETTING: UPDICT = YES	UPDICT = NO
DICTIONARY COMMAND: DICTIONARY dict-name;	Update	Read
DICTIONARY dict-name READ;	Read	Read
DICTIONARY dict-name UPDATE;	Update	Not Opened

When messages and/or all commands *are logged*:

	DCUST MACRO SETTING: UPDICT = YES	UPDICT = NO
DICTIONARY COMMAND: DICTIONARY dict-name;	Update	Not Opened
DICTIONARY dict-name READ;	Not Opened	Not Opened
DICTIONARY dict-name UPDATE;	Update	Not Opened

Note:

The above tables do not include the effects of the SET DICTIONARY-UPDATE command.

For details of the DCUST installation macro, refer to the appropriate Manager Products installation manual.

Opening a Dictionary Using the SYSNAME Clause

To open a dictionary using the SYSNAME clause, enter:

`DICTIONARY` *`dictionary-name`* `SYSNAME` *`ddname`* `;`

where *`ddname`* is the DDNAME used in an OS job control statement to specify the external dataset name (physical file name) of the dictionary's Index dataset.

The SYSNAME clause is available in OS environments only. It is provided to execute Manager Products using a single JCL procedure. Substitution is performed to specify the required physical dataset names. Substitution cannot be performed for the DDNAME field of a DD statement. Using the SYSNAME clause allows the DDNAME to remain fixed while opening a dictionary of a different name.

For example, if the procedure is invoked specifying DICT=POOL and includes:

```
//DICT    DD   DSN=&DICT..INDEX, etc
//DICTD   DD   DSN=&DICT..DATA.ENTRIES, etc
//DICTS   DD   DSN=&DICT..SOURCE, etc
//DICTE   DD   DSN=&DICT..RECOVER, etc
//DICTJ   DD   DSN=&DICT..LOG, etc
```

then the command:

```
DICTIONARY POOL SYSNAME DICT ;
```

opens a dictionary named POOL, for which the DSN names would be POOL.INDEX, POOL.DATA.ENTRIES, POOL.SOURCE, POOL.RECOVER, and POOL.LOG.

When you specify the SYSNAME clause, ControlManager recognizes the dictionary name given in the DICTIONARY command and outputs it in page headings and in any messages that refer to the dictionary by name.

Refer to *ASG-Manager Products Installation in OS Environments* for details of Manager Products job control requirements.

How to Specify Dictionary Buffer Pool Sizes

To specify the size of the dictionary buffer pools, enter:

```
DICTIONARY dictionary-name  IBUF n1  SBUF n2  DBUF n3 ;
```

where *n1*, *n2*, *n3* are integers in the range 2 to 32,000 inclusive, being the number of input/output buffers to be allocated to the buffer pool for the Index, Source and Data Entries datasets, respectively.

The IBUF, SBUF and DBUF clauses are all optional.

For a DIV dictionary, the clauses are accepted, but ignored. Users of a DIV dictionary access a shared, fixed size buffer.

Very large buffer pools can only be allocated in MVS/XA and MVS/ESA environments, subject to the availability of sufficient virtual storage.

The value you specify in each clause overrides the value of the corresponding keyword in the DCUST installation macro. The IBUF clause overrides the IBUF keyword; the SBUF clause overrides the SBUF keyword; the DBUF clause overrides the DBUF keyword.

For details of the IBUF, SBUF and DBUF keywords of the DCUST installation macro, refer to the appropriate Manager Products installation Manual.

Monitoring Buffer Steals

To record and report the number of buffer steals during a dictionary session, enter:

```
DICTIONARY dictionary-name MONITOR ;
```

If MONITOR is not specified, then in the output of the QUERY DICTIONARY BUFFERS command, asterisks appear in the fields for the number of buffer steals. Also, command execution is slightly faster.

For a DIV dictionary, the MONITOR keyword is accepted, but ignored. Buffer steals do not occur with a DIV dictionary.

How a DICTIONARY Command is Validated

The sequence of actions when a DICTIONARY command is entered, and the checks made to determine whether the command is valid, are:

- Any dictionary currently open is closed
- A check is made that all of the dictionary datasets are allocated. If the dictionary is being opened in read-only mode, or the value of the UPDICT keyword of DCUST is NO, then the Log dataset is not checked.

  If the check fails this message displays:

  ```
  type DATA SET NOT ALLOCATED/CREATED
  ```

  where *type* is INDEX, DATA, SOURCE, RECOVERY or LOG.
- A check is made that the control block record in each dataset of the dictionary corresponds with the dictionary-name specified in the command. If the check fails, this message displays:

  ```
  name NOT FOUND IN DICTIONARY DATA SET CONTROL BLOCK
  ```

  where *name* is the name of the required dataset with which a control block fails to correspond. This check is made to ensure that datasets from different dictionaries are not brought together.
- A check is made that the date and time of creation of each dataset of the dictionary agrees with the date and time of creation of the Index dataset.

  If the check fails, this message displays: *name* `NOT CREATED AT SAME DATE/TIME AS INDEX`. Where *name* is the name of the dataset whose date and/or time of creation differ from those recorded in the index. This check is made to ensure that different versions of datasets of the same dictionary are not incorrectly associated (for example, that old and new datasets are not wrongly brought together after a restore) and that an unrelated dataset having the same name as a dictionary dataset is not accessed in place of a dictionary dataset.

- The automatic error recovery system recovers the dictionary if necessary.
- A check is made that the update control numbers of the Source, Data Entries and Error Recovery datasets of the dictionary agree with the index control number.

  If the check fails, this message displays:

  name `DOES NOT HAVE SAME UPDATE CONTROL NUMBER AS INDEX`

  where *name* is the name of the dataset concerned. A check is made that the control number of the log dataset agrees with that recorded for it in the index dataset.

  If the check fails this message:

  `TRANSACTION CONTROL NUMBER MISMATCH`

  displays. This is a further check that different versions of the dataset are not wrongly associated after a restore (for example, that a restore of a single dataset has not taken place).
- If all checks succeed, the DICTIONARY command is accepted as valid.

DICTIONARY Syntax

```
>>──┬─DICTIONARY─┬──dictionary-name──┬──────────┬──────────────────────────────>
    └─DI─────────┘                   ├─READ─────┤
                                     └─UPDATE───┘

>──┬──────────────────┬──┬──────────┬──┬──────────┬──┬──
```

DOES (DMR-DD1)

The DOES command determines whether one repository member uses another.

To find out whether one member directly uses another member, enter:

```
DOES member-name-1 DIRECTLY USE member-name-2 ;
```

The answer is either YES or NO.

To find out whether a member directly or indirectly uses another member, enter:

```
DOES member-name-1 INDIRECTLY USE member-name-2 ;
```

The answer is either YES or NO.

Note:

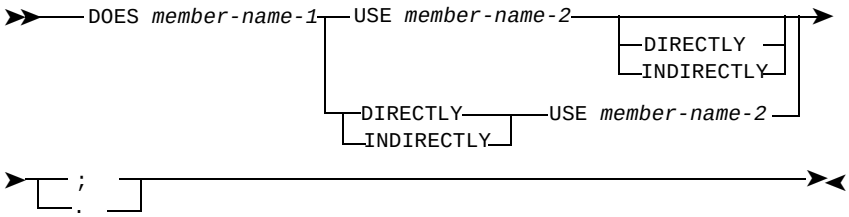
The answer is NO if *member-name-1* directly or indirectly uses a member higher in the member type hierarchy (for example, via a USER-EXIT clause) which in turn directly or indirectly uses member-name-2.

If neither DIRECTLY nor INDIRECTLY is specified in the command then the default is determined by the current DIRECTLY-ASSUMPTION setting.

The security facility controls access to information held in the repository. For details of how security affects interrogations refer to "The Security System" on page 17.

Refer to *ASG-ControlManager User's Guide* for details of the SET DIRECTLY-ASSUMPTION command.

DOES Syntax



where *member-name-1* and *member-name-2* are the names of repository members.

DROP (DMR-DD1)

The DROP command drops specific entries from an existing KEPT-DATA list.

To drop index-names from the unnamed KEPT-DATA list, enter:

```
DROP command ;
```

To drop index-names from a named KEPT-DATA list, enter:

```
DROP FROM list-name command ;
```

where:

list-name is the name you have given to a KEPT-DATA list.

command can be any of the following:

BULK ENCODE	GLOSSARY	WHAT
BULK PRINT	LIST	WHICH
BULK REPORT	PERFORM	WHOSE

If you specify a list-name, then all occurrences of the index-names selected by the prefixed command are deleted from that KEPT-DATA list (provided it exists).

If you do not specify a list-name, then all occurrences of the index-names selected by the prefixed command are deleted from the unnamed KEPT-DATA list (provided one exists).

For example:

```
KEEP IN MODLIST WHICH MODULES CONSTITUTE SYS-EMP-REC ;
DROP FROM MODLIST LIST MODULES ONLY MASTER ;
LIST KEPT IN MODLIST ;
```

will create a KEPT-DATA list named MODLIST, containing all MODULEs used by system SYS-EMP-REC except those whose member names begin with the string MASTER.

If you specify the keyword AND, then output from the prefixed command displays.

For example:

```
DROP FROM ACCOUNT-LIST AND LIST DUMMIES ;
```

will drop from a KEPT-DATA list named ACCOUNT-LIST all dummy members and list all dummy members on the dictionary.

If you do not specify the keyword AND, then except for a count of the number of index-names dropped from the KEPT-DATA list, output from the prefixed command is suppressed.

The DROP command is processed faster when the AND keyword is not specified. This is because the selection of index-names then applies only to the KEPT-DATA list, not to the entire dictionary.

If you prefix a PERFORM command with a DROP command the index-names specified by the selection keywords following the input-stream will be dropped from the KEPT-DATA list.

For example:

```
DROP PERFORM 'PRINT *' ONLY EMP ;
```

will drop from the unnamed KEPT-DATA any members whose name begins with the string EMP, and will print those members.

If the DROP command is within a PERFORM command input-stream, then the index-names generated by the input-stream are dropped from the KEPT-DATA list.

If the Audit and Security facility (selectable unit CMR-DD3) is installed, acceptance of the DROP command is subject to:

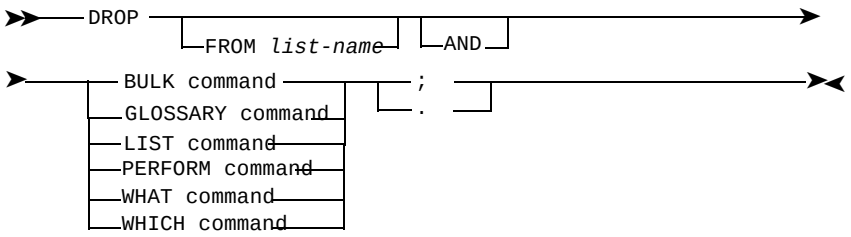
- Access security levels (if the command is BULK PRINT, BULK REPORT, GLOSSARY, LIST, PERFORM, WHAT, WHICH or WHOSE)
- Alter and insertion security levels (if the command is BULK ENCODE).

Refer to "KEEP and ALSO (DMR-DD1)" on page 156 for details of creating a KEPT-DATA list.

Refer to *ASG-Manager Products Status Concepts* manual for details of the effects of status on KEPT-DATA lists.

Refer to "The Security System" on page 17 for details of security.

DROP Syntax



where *list-name* is the name of a KEPT-DATA list.

ENCODE (CMR-CM01)

The ENCODE command generates an encoded record from a member's source record and inserts this encoded record into the dictionary's Data Entries dataset.

Refer to "ENCODE Syntax" on page 131 for the syntax of the ENCODE command.

How to Encode a Member

To generate an encoded record from a member's source record, and to insert this encoded record into the dictionary's Data Entries dataset, enter:

```
ENCODE member-name-list ;
```

where *member-name-list* is a list of one or more member names separated by commas.

For example, to encode three members named EMP-CODE, EMP-NAME and EMP-ADDR, enter:

```
ENCODE EMP-CODE, EMP-NAME, EMP-ADDR ;
```

An encoded record is generated if the source record is syntactically correct.

If a source record for member name does not exist, this message displays:

```
DM01290E member-name NO SOURCE PRESENT
```

and processing continues with the next member name in the list of member names or the next command.

Members can be of any member type available in your dictionary.

Use the SHOW UDS command to find out which member types are available in your dictionary.

You cannot encode a member that is locked by another user. If a member within the member-name-list is locked, the ENCODE command is rejected and processing continues with the next member name or command.

You can only encode a member if your security level is sufficient for you to alter that member and to access all members directly used by that member.

ENCODE is treated as an updating command by the automatic error recovery system, which operates individually in respect of each member specified in the member name list.

You can encode a selection of members in bulk by using the BULK ENCODE command.

Note:

- Details relating to locking are applicable only if the Workstation Interface facility (selectable unit CMR-WS01) is installed.
- Details relating to security levels are applicable only if the Audit and Security facility (selectable unit CMR-DD3) is installed.
- Details relating to statuses are applicable only if either the Basic or Advanced Status facility (selectable unit CMR-DD2 and CMR-AD21 respectively) is installed.

Refer to "LOCK (CMR-WS01)" on page 173 for details of member locking.

Refer to "Security Levels" on page 18 for details of security levels.

Encoding Members in a Status Environment

You can only encode a member if it has a source record in the current status and the status is not read-only.

If you attempt to encode a member in a read-only status, this message displays: `DM00093S COMMAND NOT ALLOWED IN A READ-ONLY STATUS` and the ENCODE command is rejected.

If there is no source record in the update status, this message displays:

```
DM01290E member-name NO SOURCE PRESENT
```

An ENCODE command is rejected even if a source record exists in another status.

If, as a result of a member being encoded, a dummy data entries record is created, it is created in the current status.

Changes to used-by tables are also made in the current status. If there is not a used-by table for the relevant member in the current status, ControlManager looks first in the direct base status and then in each indirect base status in turn, until the root status is reached. The first used-by table it finds is copied to the current status, and the changes made to it.

If the encoding of a member results in a change of its member type in the current status as compared with its member type in the direct base status, this change is reflected in the uses tables in the current status of all the members that use or are used by it.

Refer to *ASG-Manager Products Status Concepts* for details of statuses.

ENCODE Syntax

```
                <<<< , <<<<
>>---ENCODE member-name ---+-;-+---------------------------><
                           '-:-'
```

where *member-name* is the name of a dictionary member.

ERASE KEPT-DATA (DMR-DD1)

The ERASE KEPT-DATA command removes KEPT-DATA lists from virtual storage.

To erase the unnamed KEPT-DATA list and its contents, enter:

```
ERASE KEPT-DATA ;
```

or

```
ERASE KEPT-DATA UNNAMED ;
```

To erase one or more named KEPT-DATA lists and their contents, enter:

```
ERASE KEPT-DATA LISTS list-name ;
```

where *list-name* is the name of one or more KEPT-DATA list, where each name is separated by a comma.

For example, to erase the KEPT-DATA lists named KLIST1 and KLIST2, enter:

```
ERASE KEPT-DATA LISTS KLIST1, KLIST2 ;
```

To erase unnamed and named KEPT-DATA lists, enter:

```
ERASE UNNAMED AND LISTS list-name ;
```

To erase all named and unnamed KEPT-DATA lists and their contents, enter:

```
ERASE KEPT-DATA ALL ;
```

All named and unnamed KEPT-DATA lists and their contents are also erased by the execution of a DICTIONARY command, a CLOSE command or a LOGOFF command.

The unnamed KEPT-DATA list and its contents are also erased by the execution of an AUTHORITY command.

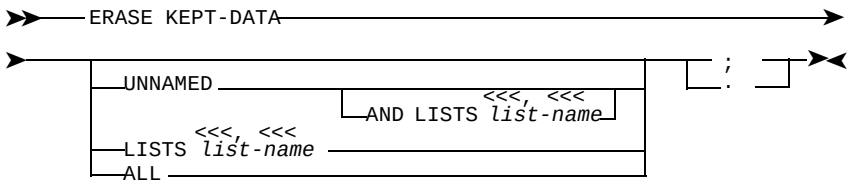
These commands replace the contents of the unnamed KEPT-DATA list or of any existing KEPT-DATA list specified in the command:

- A subsequent KEEP command in which the keyword ALSO has not been specified
- A FETCH KEPT-DATA command.

This command erases the contents of the unnamed KEPT-DATA list or of any existing KEPT-DATA list specified in the command:

- A PERFORM command containing a named or unnamed KEPT-DATA list as the selection and the CLEAR-KEPT-DATA keyword.

ERASE KEPT-DATA Syntax



where *list-name* is the name of a KEPT-DATA list.

FETCH KEPT-DATA (DMR-DD1)

The FETCH KEPT-DATA command retrieves KEPT-DATA lists stored on the MP-AID in KEPT-DATA members.

To fetch a KEPT-DATA-LIST member from the MP-AID into the unnamed KEPT-DATA list, enter:

```
FETCH KEPT-DATA FROM mpaid-name ;
```

where *mpaid-name* is the name of the MP-AID KEPT-DATA member in which the KEPT-DATA list is stored. The contents of an existing unnamed KEPT-DATA list are replaced by the contents of *mpaid-name*.

To retrieve a KEPT-DATA list from the MP-AID into a named KEPT-DATA list, enter:

```
FETCH KEPT-DATA AS list-name FROM mpaid-name ;
```

where *list-name* is the name of the KEPT-DATA list. If there is already a KEPT-DATA list called *list-name*, then its contents are replaced by the contents of *mpaid-name*.

A retrieved KEPT-DATA list does not include index-names that have been removed from the dictionary since the list was stored.

If you store a KEPT-DATA list in one dictionary and retrieve into another, the retrieved list includes only those index-names that exist in both the stored KEPT-DATA list and the second dictionary.

If the Audit and Security facility (selectable unit CMR-DD3) is installed,

KEPT-DATA lists retrieved from the MP-AID, or another dictionary, will include only those index-names that you can access.

Refer to the STORE KEPT-DATA command for details of storing KEPT-DATA lists.

Refer to *ASG-Manager Products Status Concepts* for details of the effects of status on KEPT-DATA lists.

Refer to "The Security System" on page 17 for details of the effects of security.

FETCH KEPT-DATA Syntax

```
>>──FETCH KEPT-DATA──┬──────────────────┬──FROM mpaid-name──────>
                     └──AS list-name────┘
>──┬─;─┬──────────────────────────────────────────────────────><
  └─.─┘
```

where:

list-name is the name of a KEPT-DATA list

mpaid-name is the name of the KEPT-DATA member on the MP-AID in which the KEPT-DATA is stored.

FILE (CMR-FE01)

The FILE command enters the contents of an Update Buffer into the dictionary and generates an encoded record, or enters the contents of an Edit Buffer onto the MP-AID.

If you have updated or edited an existing member, enter:

```
FILE ;
```

When you FILE an Update Buffer your changes are made to the existing records in the Source dataset of the dictionary. An encoded record is generated from the records in the Source dataset, and added to the Data Entries dataset. This encoded record will replace any existing encoded record for the member (if the encode is successful).

When you FILE an Edit Buffer your changes are made to the existing member on the MP-AID.

If you have created a new member, enter:

```
FILE dictionary-member-name ;
```

or

```
FILE mpaid-user-member-name ;
```

When you FILE an Update Buffer, records for a new member are added to the Index dataset and Source dataset, and an encoded record is generated and added to the Data Entries dataset. The dictionary-member-name can have a maximum of 32 characters.

You cannot FILE a new dictionary member giving it the name of an existing member, but you can RFILE it.

If the Workstation Interface facility is installed, (selectable unit CMR-WS01) you cannot file a dictionary member that is locked by another user. If *`dictionary-member-name`* has been locked since you updated it, the FILE command is rejected.

When you FILE an Edit Buffer this new member is added to the MP-AID. The *`mpaid-user-member-name`* can have a maximum of 10 characters. It must not be the same as that of any other USER-MEMBER owned by your logon ID.

Error and Warning Messages

If you FILE an Update Buffer and an error message is generated, the member will not be encoded. Instead, the source of the member, together with the error message, is displayed in a Lookaside Buffer: you can note the errors, QUIT to return to the Update Buffer, correct the errors, and then re-enter the FILE command.

If you FILE an Update Buffer and a warning message is generated, the member may or may not be encoded.

You can display the source of the member in a Lookaside Buffer as follows, by using the SET UPDATE-OUTPUT command:

- Whenever FILE is executed
- Only when FILE generates warnings or errors
- Only when FILE generates errors.

If you FILE an Edit Buffer and an error or warning message is generated, then this is displayed in the message line.

FILE Syntax

```
►──FILE ──┬──────────────┬──┬─;─┬──────────────────────────►◄
          └─member-name──┘  └─.─┘
```

where *member-name* is the name of either a dictionary or an MP-AID member.

FORMAT AS (CMR-CM01)

The FORMAT AS command generates output from the REPORT and BULK REPORT commands in a preferred alternative layout.

To generate a preferred layout enter:

```
FORMAT AS layout-name report-command ;
```

where:

layout-name is the name of a preferred alternative layout and can be one of these:

PSR-ITEM	PSR-MODULE	PSR-DETAIL
PSR-GROUP	PSR-PROGRAM	PSR-SUMMARY
PSR-FILE	PSR-SYSTEM	

report-command is either the REPORT or the BULK REPORT command.

With the exception of the PSR-DETAIL and PSR-SUMMARY layouts, which can be specified for any member type, the layout must match the member type of the members being reported.

For example:

```
FORMAT AS PSR-ITEM REPORT TRANS-CODE ;
```

generates a report for the ITEM member TRANS-CODE in the preferred alternative layout named PSR-ITEM.

The layout name specified in the FORMAT AS command will override any specified in a prior SET FORMAT command. The QUERY FORMAT command will display what the current alternative layout is.

The User Defined Output facility (selectable unit DYR-UD15) provides the FORMAT VIA command with which alternative layouts can be specified for output from REPORT and BULK REPORT commands.

FORMAT AS Syntax

```
>>─FORMAT AS preferred-set-layout-name report-command ┬─;─┬──><
                                                      └─.─┘
```

where:

preferred-set-layout-name is:

```
>──┬─PSR-ITEM──────────┬──────────────────────────────────>
   ├─PSR-GROUP─────────┤
   ├─PSR-FILE──────────┤
   ├─PSR-MODULE────────┤
   ├─PSR-PROGRAM───────┤
   ├─PSR-SYSTEM────────┤
   ├─PSR-SUMMARY───────┤
   └─PSR-DETAIL────────┘
```

report-command is a REPORT or BULK REPORT command.

GLOSSARY (DMR-DD1)

Refer to "GLOSSARY Syntax" on page 148 for the syntax of the GLOSSARY command.

Introduction to the GLOSSARY Command

The GLOSSARY command displays:

- Details of selected encoded members or dummies
- Selected aliases and details of encoded members that have those aliases.

You specify the information to be output in a GIVING clause. This may be specific common-clauses and/or the relationship keywords USAGES and REFERENCES. You may also specify attributes and clauses peculiar to a certain member type, by stating that member type in a FOR clause.

You select the members and aliases for which you want a glossary, by specifying one or more of these:

- LOCKED: locked members
- KEPT-DATA: members or aliases kept in a KEPT-DATA list
- MEMBERS: a list of specific member names
- One or more specific member types
- EXCEPT: to exclude specific member types
- DUMMY, DUMMYS or DUMMIES: dummy members
- ALIASES: general or specific aliases
- CHECK-OK, CHECK-NEEDED: members that have passed or need integrity checks.

You can further refine this selection by specifying one or more of these secondary selections:

- Name: selecting members and aliases whose names meet given criteria
- Status: selecting members and aliases which exist, or were processed, in specific statuses
- Time: selecting members and aliases which were, or were not, processed at a given time or during a given period of time
- User: selecting members and aliases which were, or were not, processed by a certain user.

The SIGNIFICANT keyword restricts output to those members containing the specified clause(s).

The ALL-STATUSES keyword selects index-names from all the statuses within your Status Window.

You can alter the form of the output with the ALPHABETICALLY and DETAILS-ONLY keywords: the former outputs members and aliases alphabetically, the latter suppresses the output headings and the total-lines.

The security facility controls access to information held in the repository. For details of how security affects interrogations refer to "The Security System" on page 17.

Outputting Common Clauses

To obtain a glossary of all members, outputting specific common clauses, enter:

```
GLOSSARY GIVING extract ;
```

where *extract* is the name of a common clause (optionally preceded by the keywords DIRECT or INDIRECT).

If you specify more than one extract, each must be separated by a comma. The member names are output in alphanumeric order, along with their extracts, which are output in the order in which they are stated in the command.

For example:

```
GLOSSARY GIVING CATALOG, DESCRIPTION ;
```

outputs a glossary of all members, giving their DESCRIPTION and CATALOG clauses.

The ADMINISTRATIVE-DATA, COMMENT, DESCRIPTION, NOTE and QUERY clauses can contain one or more delimited text strings all of which form a single piece of text. If you want to output particular lines within these text extracts, enter:

GLOSSARY GIVING extract LINES *n* TO *m* ;

where *n* and *m* are unsigned integers in the range 1 to 32767, identifying delimited text strings. If m is present it must be equal to or greater than *n*. *m* can be END. TO *m* is optional.

For example:

```
GLOSSARY GIVING COMMENT LINES 1 TO 5 ;
```

outputs a glossary of all members, giving lines 1 to 5 of their COMMENT clause.

You can output any of the other clauses and attributes of members by specifying a FOR clause in the GLOSSARY command.

Outputting Clauses Specific to Given Member Types

To glossary all members of a specified member type and display the clauses specific to that member type, enter:

```
GLOSSARY FOR member-type GIVING detail ;
```

where:

member-type is the interrogate keyword for any member type available in your repository.

Use the SHOW UDS command to find out the available member types.

detail is one or more of the following, separated by commas:

- The name of a common clause
- The name of a clause specific to the member type given in the FOR clause.

Use the SHOW MEMBER-TYPE command to find out the available clauses.

You need not specify the FOR keyword unless you glossary more than one member type or a selection of members using the ALIASES, EXCEPT, KEPT-DATA or MEMBERS keywords.

For example, the two commands:

```
GLOSSARY ITEM GIVING USER-EXIT ;
```

```
GLOSSARY FOR ITEM GIVING USER-EXIT ;
```

display the same output.

You can specify more than one FOR clause, but each must include a GIVING clause. Only one member type can be entered in a FOR clause.

For example:

```
GLOSSARY FOR PROGRAMS GIVING ASSIGNMENT, OBJECT-COMPUTER FOR
CONVENTIONAL-FILES GIVING SIZE ;
```

outputs a glossary of all PROGRAM members, giving their ASSIGNMENT and OBJECT-COMPUTER clauses, and all CONVENTIONAL-FILE members, giving their SIZE clause.

The members selected are output alphanumerically within their member type. Member types and details are output in the order in which they are stated in the command, unless you specify the ALPHABETICALLY keyword.

Selecting or Excluding Specific Member Types

To glossary all members of a specified member type, enter:

```
GLOSSARY member-type-list GIVING extract ;
```

or

```
GLOSSARY member-type-list FOR member-type GIVING detail ;
```

where *member-type-list* is the interrogate keyword for any member type, or any collective member type, available in your repository. Each keyword must be separated by a comma.

You need not specify the FOR keyword if member-type and member-type-list are identical and an ALIASES, EXCEPT, KEPT-DATA or MEMBERS keyword has not been specified.

For example:

```
GLOSSARY GROUPS, CONVENTIONAL-FILES FOR CONVENTIONAL-FILES GIVING
SIZE ;
```

outputs a glossary of all GROUP and CONVENTIONAL-FILE members, outputting the SIZE clause of the latter.

If you want to exclude particular member types from the output of the command, enter:

```
GLOSSARY EXCEPT member-type-list GIVING details ;
```

or

```
GLOSSARY EXCEPT member-type-list FOR member-type GIVING extract ;
```

For example:

```
GLOSSARY EXCEPT MODULES, SYSTEMS FOR PROGRAMS GIVING AUTHOR ;
```

outputs a glossary of all members, except MODULES and SYSTEMS, giving the AUTHOR clause of the PROGRAM members.

Selecting Aliases

To output the name and type of a selection of aliases, enter:

```
GLOSSARY aliases ;
```

aliases is one or more of:

```
ALIASES
ALIAS alias-type
ALIASES number
alias-type ALIASES.
```

where:

number is an integer in the range 1 to the maximum number of general aliases allowed in your repository.

alias-type is one of the specific alias-types available in your repository.

For example:

```
GLOSSARY COBOL ALIASES, ALIASES 1 GIVING CATALOG;
```

outputs a glossary of all COBOL aliases and those aliases occurring first in an ALIAS clause, giving the catalog classifications of the members of which they are aliases.

You can use the SHOW ALIAS-TYPES command to output the specific alias-types and the maximum number of general aliases available in your repository.

Selecting a List of Member Names

To obtain a glossary of named members, enter:

```
GLOSSARY MEMBERS member-name GIVING extract ;
```

or

```
GLOSSARY MEMBERS member-name FOR member-type GIVING detail ;
```

where *member-name* is one or more member names, separated by commas.

For example:

```
GLOSSARY MEMBERS TAX, ADMIN, DEPARTMENT ITEM, GROUP FOR ITEM GIVING
NOTE ;
```

outputs a glossary of those named members that are ITEM or GROUP members, giving the NOTE clauses of the ITEM members.

Selecting Dummy Members

To obtain a glossary of all dummy members, giving the members that use them, enter:

```
GLOSSARY DUMMIES GIVING USAGES ;
```

DUMMIES can be replaced with the alternative keyword DUMMYS.

You can also select dummy members of a specific member type, by entering:

```
GLOSSARY DUMMY member-type GIVING USAGES ;
```

where *member-type* is the interrogate keyword for any member type, or any collective member type, available in your repository.

For example:

```
GLOSSARY DUMMY ITEMS GIVING DIRECT USAGES ;
```

outputs a glossary of all dummy ITEM members, giving the members that directly use them.

If you want to output a glossary of all dummy members, except for those of a given member type, enter:

```
GLOSSARY DUMMIES EXCEPT member-type GIVING USAGES ;
```

where *member-type* is as defined above.

For example:

```
GLOSSARY DUMMIES EXCEPT ITEMS, PROGRAMS GIVING USAGES ;
```

outputs a glossary of all dummy members, except those that are ITEM or PROGRAM members, giving the members that use them.

Refer to "Collective Member Types" on page 316 for details of collective member types.

Selecting Non-Dummy Members

To output a glossary containing only non-dummy members, that is, members which have been defined and encoded, giving the names of members which refer to them, enter:

```
GLOSSARY REAL GIVING USAGES ;
```

The keyword REAL may be used in conjunction with other keywords as the opposite of DUMMY. For example, to output a glossary of all non-dummy MODULE members that are currently locked, giving their DESCRIPTION clause, enter:

GLOSSARY REAL MODULES GIVING DESCRIPTION ;

Selecting Locked Members

You can select members that are currently locked by including the LOCKED keyword in the command. For example, to output a glossary of all PROGRAM members that are currently locked, giving their DESCRIPTION clause, enter:

```
GLOSSARY LOCKED PROGRAMS GIVING DESCRIPTION ;
```

Selecting KEPT-DATA Lists

To output a glossary of members and aliases stored in a KEPT-DATA list, enter:

```
GLOSSARY kept-data GIVING extract ;
```

or

```
GLOSSARY kept-data FOR member-type GIVING detail ;
```

where *kept-data* is one of:

```
KEPT-DATA
KEPT-DATA IN list-name
```

where *list-name* is the name of a KEPT-DATA list.

This can be combined with any of the selections available with the GLOSSARY command.

For example:

```
GLOSSARY KEPT-DATA ITEMS GIVING CATALOG ;
```

outputs a glossary of all ITEM members held in the unnamed KEPT-DATA list, giving their CATALOG clause.

```
GLOSSARY KEPT-DATA IN LIST1 ALIASES ;
```

outputs a glossary of all the aliases held in the KEPT-DATA list named LIST1.

Members that appear in both a list of named members (in a MEMBERS clause) and the KEPT-DATA list are only output once.

Member names and aliases are output in the order that they appear in the KEPT-DATA list. Lists of member names and member types are output in the order in which they appear in the command, unless you specify the ALPHABETICALLY keyword.

Outputting Members that Use or are Used By Selected Members

To obtain a glossary of all members, giving the members that use them, enter:

```
GLOSSARY GIVING USAGES ;
```

The members output depend on the current DIRECTLY-ASSUMPTION setting. You can override this setting by specifying either of the keywords DIRECT or INDIRECT in the command. DIRECT selects only direct references, while INDIRECT selects both direct and indirect references.

For example:

```
GLOSSARY ALIASES COBOL GIVING DIRECT USAGES ;
```

outputs the alias name and alias type of all COBOL aliases, giving the members in whose encoded record they appear, and the members that directly use those members.

```
GLOSSARY DUMMIES GIVING INDIRECT USAGES ;
```

outputs a glossary of all dummy members, giving the members that directly or indirectly use them.

You can also obtain a glossary of members which gives the members they use, by using the keyword REFERENCES in place of the keyword USAGES.

For example:

```
GLOSSARY GROUPS GIVING DIRECT REFERENCES ;
```

outputs a glossary of all GROUP members, giving the members they directly use.

A reference path is not traced beyond a member that is used by a member at a lower level in the member type hierarchy (via, for example, a USER-EXIT clause), or beyond a member already selected for output.

The security facility controls access to information held in the repository. For details of how security affects interrogations refer to "The Security System" on page 17.

Refer to *ASG-ControlManager User's Guide* for details of the SET DIRECTLY-ASSUMPTION command.

Outputting Only Those Members With a Given Clause

You can prevent members that do not contain the given extract from being output, by entering:

```
GLOSSARY SIGNIFICANT GIVING extract ;
```

You can also prevent members from being output that are not of the given member type or do not contain the given detail, by entering:

```
GLOSSARY SIGNIFICANT selection FOR member-type GIVING detail ;
```

If you do not specify the SIGNIFICANT keyword, then the member name and member type of all selected members are output, regardless of their member type or whether their encoded record contains the extract or detail specified in the GIVING clause. The specified extract or detail is output for those members whose encoded record contains it.

For example:

```
GLOSSARY SIGNIFICANT KEPT-DATA FOR ITEMS GIVING ALIASES ;
```

outputs a glossary of all ITEM members stored in the unnamed KEPT-DATA list which contain an ALIAS clause in their encoded record, and outputs this clause, whereas:

```
GLOSSARY KEPT-DATA FOR ITEMS GIVING ALIASES ;
```

outputs a glossary of all the members stored in the unnamed KEPT-DATA list, regardless of whether they contain an ALIAS clause or are ITEM members, and gives the aliases of the ITEM members.

Name, Status, Time and User-Related-Selections

Refer to "Applying Commands to a Selection Of Index-names Based on Name, Status, Time and User" on page 293 for details of name, status, time and user-related-selections.

Displaying Output Alphabetically

To output member names and aliases alphanumerically, regardless of the order in which they appear in a KEPT-DATA list or the sequence in which a member name selection has been specified, enter the keyword ALPHABETICALLY in the command. For example:

```
GLOSSARY KEPT-DATA ITEMS, GROUPS ALPHABETICALLY FOR ITEMS GIVING
NOTE ;
```

outputs a glossary of all ITEM and GROUP members held in the unnamed KEPT-DATA list, alphabetically in the order of their member names, rather than the order in which they were added to the KEPT-DATA list. The NOTE clause of the ITEM members is also output.

Suppressing Headings and Total-Lines

The DETAILS-ONLY keyword suppresses output headings and total-lines. Form feeds before and after the command are also suppressed, but not form feeds within the glossary.

The DETAILS-ONLY keyword must be specified after any selection and before the GIVING or FOR and GIVING clause. For example:

```
GLOSSARY KEPT-DATA DETAILS-ONLY FOR MODULE GIVING AUTHOR ;
```

If you do not specify the DETAILS-ONLY keyword, then an output heading and total lines are output.

Output headings display the selection of members and/or aliases being output.

Total-lines display the number of members and/or aliases output and the total number of members of each member type. With the Audit and Security facility the number of the selected members with an access level higher than the user's security level is also output.

Suppressing Member-Type and Top-Level Clause Names

By default, output from the GLOSSARY command includes the member-type of each member selected, and the names of the top-level clauses/attributes along with their contents. You can suppress member-types by using the NO-TYPE keyword, and top-level clause/attribute keywords by using the NO-KEYWORD keyword.

If used, these keywords must be specified after any selection and before the GIVING or FOR and GIVING clause. For example:

```
GLOSSARY KEPT-DATA NO-TYPE FOR GROUPS GIVING CONTAINS

GLOSSARY KEPT-DATA NO-KEYWORD GIVING NOTE, DESCRIPTION
```

Selecting Members and Aliases from All Statuses

The GLOSSARY command outputs a glossary of members or aliases whose encoded record exists in either the current status or the next visible status. By specifying the ALL-STATUSES keyword you can output a glossary of members or aliases in all the statuses currently available to you. A separate glossary is output for each status. For example:

```
GLOSSARY ALL-STATUSES MEMBER ADMIN GIVING DESCRIPTION ;
```

outputs a glossary of the member ADMIN in all the statuses in which it is encoded.

Selecting Members That Have Passed or Need Integrity Checks

To select check-ok members, members that have passed integrity checks, include the CHECK-OK keyword in the command.

To select check-needed members, members that need integrity checks, include the CHECK-NEEDED keyword in the command.

For further details of check-ok and check-needed members, refer to *ASG-MethodManager Administration*.

GLOSSARY Command Examples

To obtain a glossary of all members, outputting lines 1 and 2 of their DESCRIPTION clause, enter:

```
GLOSSARY GIVING DESCRIPTION LINES 1 TO 2 ;
```

To obtain a glossary of all members, and output the EDIT-INPUT clause of those that are MODULE members, enter:

```
GLOSSARY FOR MODULES GIVING EDIT-INPUT ;
```

To obtain a glossary of all ITEM and GROUP members, outputting their CATALOG clause, enter:

```
GLOSSARY ITEMS, GROUPS GIVING CATALOG ;
```

To obtain a glossary of all dummy members, except FILES, and output the members that directly use them, enter:

```
GLOSSARY DUMMIES EXCEPT FILES GIVING DIRECT USAGES ;
```

To obtain a glossary of all aliases, outputting the member name and member type of the members in whose encoded record they appear, enter:

```
GLOSSARY ALIASES ;
```

To obtain a glossary of the members ADMIN and TEL-NO, and, if either of them is a PROGRAM member that uses other members, output the member name and member type of the used members, enter:

```
GLOSSARY MEMBERS ADMIN, TEL-NO FOR PROGRAM GIVING REFERENCES ;
```

To obtain a glossary of all locked ITEM members held in the unnamed KEPT-DATA list that contain a USAGE clause, and output this clause, enter:

```
GLOSSARY SIGNIFICANT, LOCKED KEPT-DATA ITEMS FOR ITEMS GIVING USAGE ;
```

To obtain a glossary, in alphanumeric order, of the aliases held in the KEPT-DATA list named LIST1, enter:

```
GLOSSARY KEPT-DATA IN LIST1 ALIASES ALPHABETICALLY ;
```

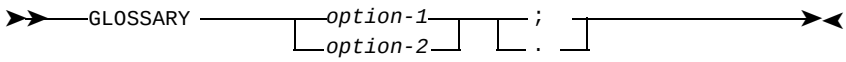
To obtain a glossary of all GROUP members, outputting their SEE clause, whilst suppressing output headings and total-lines, enter:

```
GLOSSARY GROUPS DETAILS-ONLY GIVING SEE ;
```

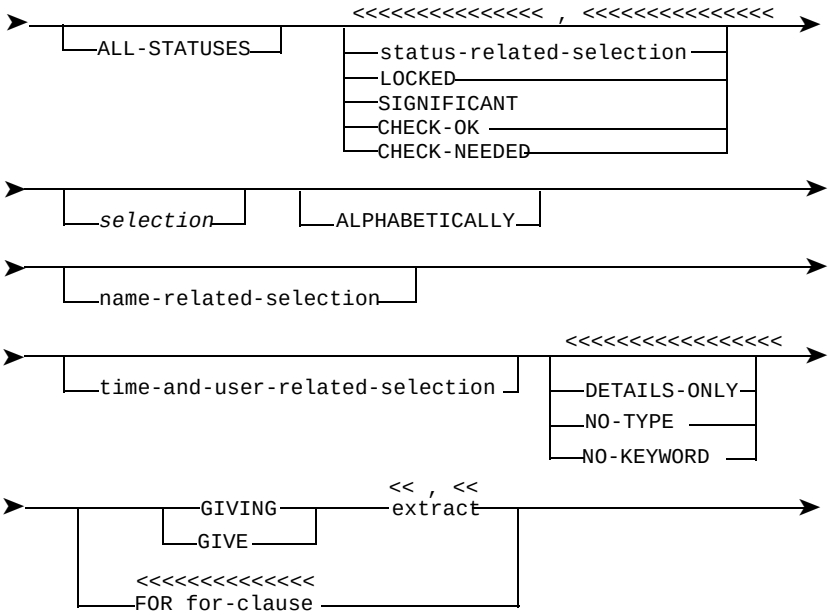
To display all check-ok PROGRAM members, giving their DESCRIPTION clause, enter:

```
GLOSSARY CHECK-OK PROGRAMS GIVING DESCRIPTION ;
```

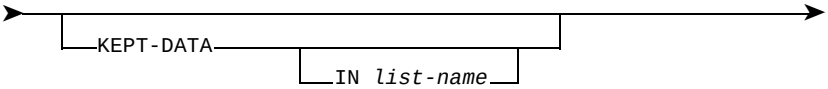
GLOSSARY Syntax



where *option-1* is:



where *selection* is:



```
►──┬──────────────────────────────────────────────────────────────────────►
   │          <<<< , <<<<
   └──MEMBERS member-name ──┘

►──┬──────────────────────────────────────────────────────────────────────►
   │                                  <<< , <<
   │──────────────────────────────── category ──────────────────────────┐
   │   ┌──DUMMY──┐
   │   │──EXCEPT─┤
   │   └──REAL───┘
   │
   │   ┌──DUMMYS──┐                   <<< , <<
   │   │──DUMMIES─┤   ┌──────────────────────┐
   │   └──REAL────┘   └──EXCEPTS-category─────┘
```

where:

list-name is the name of a KEPT-DATA list

member-name is the name of a repository member

category is:

```
►──┬──────────────────────────────────────────────────────────────────────►
   │  member-type-list ──────────────────────────────────────────────────┐
   │  ┌──ALIASES ──┬──────────────┬──────────────────────────────────────┤
   │  │            │──alias-type──┤                                      │
   │  │            └──number──────┘                                      │
   │  └──alias-type ALIASES──────────────────────────────────────────────┘
```

where:

member-type-list is the interrogate keyword for any member type/collective member type in your repository

(Use the SHOW UDS command to list the member type interrogate keywords available in your repository.)

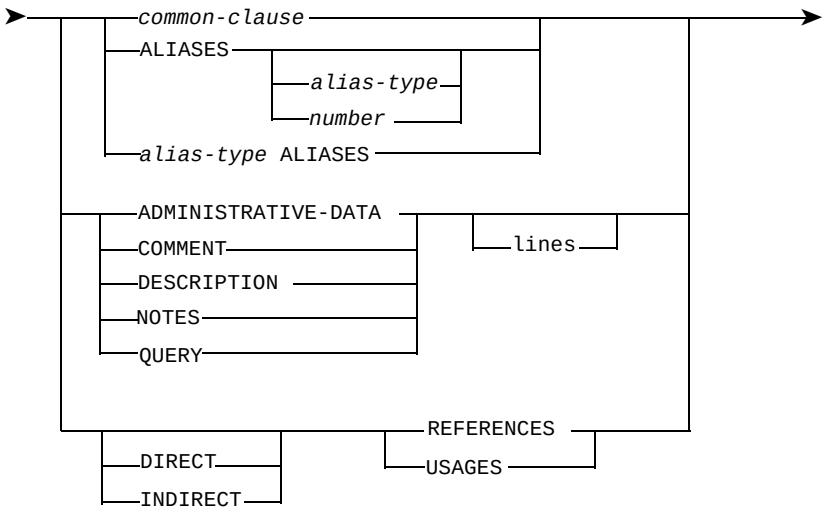
alias-type is one of the specific alias-type keywords available in your repository

number is an integer from 1 to the maximum number of general aliases allowed in your repository.

(Use the SHOW ALIAS-TYPES command to find out the specific alias-types and the number of general aliases available in your repository.)

Status, *name*, *time* and *user-related-selection* are defined in "Applying Commands to a Selection Of Index-names Based on Name, Status, Time and User" on page 293.

extract is:

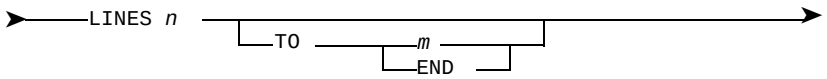


where:

common-clause is any clause available to all the member types present in your repository

alias-type and *number* are as defined above

lines is:

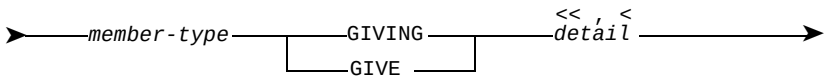


where:

m and *n* are unsigned integers in the range 1 to 32767, identifying delimited character strings.

(Use the SHOW MEMBER-TYPE command to find out which common clauses are available in your repository.)

for-clause is:



where:

member-type is as defined above

detail is *extract* as defined above, or any clause/attribute specific to the member type given in the FOR clause.

(Use the SHOW MEMBER-TYPE command to find out which clauses or attributes are available to a particular member type in your repository.)

option-2 is:

```
                                <<<<<<<<<<<<<<<< , <<<<<<<<<<<<<<<<
 ──┬──────────────────────────┬──┬──status-related-selection──┬──
   └──ALL-STATUSES────────────┘  ├──LOCKED────────────────────┤
                                 ├──SIGNIFICANT───────────────┤
                                 ├──CHECK-OK──────────────────┤
                                 └──CHECK-NEEDED──────────────┘

 ──┬──────────────────────────────────────────────────────────┬──
   └──KEPT-DATA──┬──────────────────────┬──────────────────────┘
                 └──IN list-name────────┘

 ──┬──ALIASES──┬──alias-type──┬──────────────────────────────┬──
   │           │              │                              │
   │           └──number──────┘                              │
   │                                                         │
   └──alias-type ALIASES─────────────────────────────────────┘

 ──┬──ALPHABETICALLY──┬──────────────────────────────────────┬──
   │                  │                                      │
   │                  └──name-related-selection──────────────┘

                                                <<<<<<<<<<<<<<<<<<
 ──┬──────────────────────────────────────────┬──┬──DETAILS-ONLY──┬──
   └──time-and-user-related-selection─────────┘  ├──NO-TYPE────────┤
                                                 └──NO-KEYWORD─────┘

 ──┬──────────────────────────────────────────────────────────┬──
   │                          << , <<                         │
   │   ┌──GIVING──┬──────────extract──────────────────────────┤
   │   │          │                                           │
   │   └──GIVE────┘                                           │
   │                <<<<<<<<<<<<<<<<                          │
   └──FOR for-clause──────────────────────────────────────────┘
```

where:

status-related-selection, *list-name*, *alias-type*, *number*, *name-related-selection*, *time-and-user-related-selection*, *extract* and *for-clause* are as defined above.

INSERT (CMR-CM01)

The INSERT command inserts a member's source record into the dictionary.

Refer to "INSERT Syntax" on page 155 for the syntax of the INSERT command.

How to Insert a Member

To insert a member definition (the member's source record) into the Source dataset in the current status, enter:

```
INSERT member-name ;
member-definition
;
```

where *member-name* is the name of a member that does not already exist in the dictionary, or of a member that exists as a dummy, and member-definition is its source record.

For the syntax of member definition, enter:

```
HELP member-type-name-S ;
```

where *member-type-name* is the name of the type of member you want to enter into the dictionary.

For example:

```
HELP ITEM-S ;
```

Use the SHOW UDS command to find out which member types are available in your dictionary.

For example, to insert a member named TRANS-TOT, enter:

```
INSERT TRANS-TOT ;
ITEM
HELD-AS CHARACTER 6
;
```

If a source record does not already exist for member-name, then a source record (containing the member definition lines) is entered into the Source dataset, and a record for member-name is created in the Index dataset.

If a source record already exists for member-name, the command and member definition are rejected, and processing continues with the next command.

If a dummy record exists in the Data Entries dataset, a source record is entered into the Source dataset and the record in the Index dataset is updated. The dummy encoded record is not changed.

No validation of the member definition statement syntax is carried out when the source record is inserted. Validation is carried out when the member is subsequently encoded.

If the INSERT command is followed immediately by a terminator in the first character position of an input line (which can be regarded as an empty member definition statement), an index record and an empty source record is created for the member. This can be of use, in conjunction with the Automation of Set Up facility's CONVERT command with the MERGE-ITEMS keyword, or MERGE command.

Refer to *ASG-DataManager Automation of Set-up* for details.

Unless you include a NUMBER clause in your INSERT command, lines in the source record are numbered in increments of 100, starting with 100.

If you include the PRINT keyword in your INSERT command, the source record display after it has been inserted.

The INSERT command is treated as an updating command by the automatic recovery system.

You can only INSERT a member if your general security level is equal to or higher than the dictionary's insertion security level.

Note:

- Details relating to security levels are applicable only if the Audit and Security facility (selectable unit CMR-DD3) is installed.
- Details relating to statuses are applicable only if either the Basic or Advanced Status facility (selectable units CMR-DD2 and CMR-AD21 respectively) is installed.

Refer to *ASG-ControlManager User's Guide* for the member naming rules.

Refer to *ASG-Manager Products Status Concepts* manual for details of statuses.

How to Number Lines in a Member's Source Record

To insert a source record with lines numbered in specified increments, enter:

```
INSERT member-name NUMBER n ;
member definition
;
```

n is an unsigned integer in the range 1 and 10000 (inclusive). The lines in the source record are numbered in increments of *n*, the first line being number *n*.

For example, to insert the member named TRANS-TOT with lines numbered in increments of 200, enter:

```
INSERT TRANS-TOT NUMBER 200 ;
ITEM
HELD-AS CHARACTER 6
DESCRIPTION 'EMPLOYEE TRANSACTION CODE'
;
```

If you do not specify the NUMBER clause, lines in the source record are numbered in increments of 100, starting with 100.

If you do not specify an integer this message displays:

```
DM00127W ; INVALID INTEGER
```

and *n* is assumed to be 100.

If you specify an integer of less than 1 or greater than 10000 this message displays:

```
DM01182I n NOT VALID INCREMENT - '100' ASSUMED
```

and *n* is assumed to be 100.

If you specify a signed integer in the range 1 and 10000 (inclusive) this message displays:

```
DM00127W n INVALID INTEGER
```

The sign is automatically removed, and *n* is treated as a valid integer.

If you specify an integer that causes the line numbers to extend into more than 5 digits, then, with one exception, only the last 5 digits are output in any printout or display of the source record. The exception is the response to a DISPLAY amendment line in an ALTER or MODIFY command; then all 8 digits display.

Refer to "ALTER (CMR-CM01)" on page 52 for details of DISPLAY.

Inserting and Displaying a Source Record

To insert and then display a source record, enter:

```
INSERT member-name PRINT ;
```

Line numbers in the source record will also display.

To insert but not display a source record, enter:

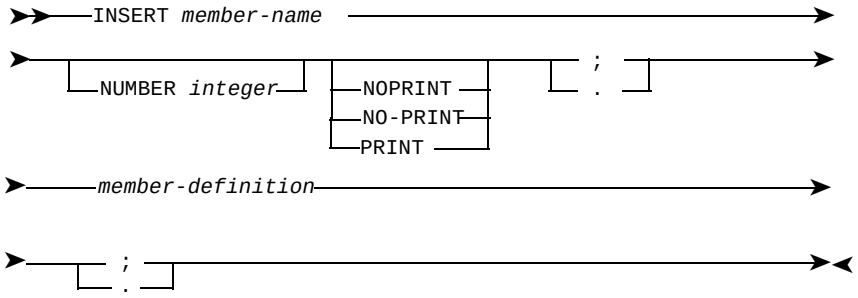
```
INSERT member-name NOPRINT ;
```

NOPRINT can be replaced with the alternative keyword NO-PRINT.

If neither NOPRINT nor PRINT is specified in the command, then:

- In interactive processing, a default of NOPRINT is assumed
- In batch processing, a default of PRINT is assumed.

INSERT Syntax



where:

member-name is the name of the dictionary member.

integer is an unsigned integer in the range of 1 to 10000, being the line number for the first line and the increment for subsequent line numbers of the member's source record.

member-definition is the member's source record.

KEEP and ALSO (DMR-DD1)

The KEEP and ALSO commands create and accumulate, in a KEPT-DATA list, lists of member names, aliases, catalog classifications and/or indexed attributes that can be used to drive further commands.

To create an unnamed KEPT-DATA list, enter:

```
KEEP command ;
```

To create a named KEPT-DATA list, enter:

```
KEEP IN list-name command ;
```

where *list-name* is the name of a list where the KEPT-DATA will be held. It can be a maximum of 79 characters in length. KEPT-DATA list names formed of characters from the standard character set need not be delimited.

ASG recommends that the name of a KEPT-DATA list should reflect its use and content.

command can be any valid variant of the following commands:

BULK ENCODEGLOSSARYWHAT
BULK PRINTLISTWHICH
BULK REPORTPERFORMWHOSE

The KEPT-DATA list contains the index-names selected by the above commands.

For example, to create a KEPT-DATA list named WHAT-USES-ADDRESS containing all members that use ADDRESS, enter:

```
KEEP IN WHAT-USES-ADDRESS WHAT USES ADDRESS ;
```

To add index-names to the existing unnamed KEPT-DATA list, enter:

```
ALSO command ;
```

To add index-names to an existing named KEPT-DATA list, enter:

```
ALSO IN list-name command ;
```

If there is no such named or unnamed KEPT-DATA list, ALSO is treated as a KEEP command.

Any number of index-names can be accumulated in a KEPT-DATA list, subject to the virtual storage available.

To create a KEPT-DATA list and to print or display the output generated, enter:

```
KEEP AND command ;
```

or

```
KEEP IN list-name AND command ;
```

To add index-names to an existing KEPT-DATA list and to print or display the output generated, enter:

```
ALSO AND command ;
```

or

```
ALSO IN list-name AND command ;
```

If you prefix a PERFORM command with KEEP or ALSO, the index-names specified by the selection keywords following the input-stream will be kept in a KEPT-DATA list. For example:

```
KEEP PERFORM 'WHAT DIRECTLY USES *' MEMBER
                ACC-NO, TAX-CODE, EMP-NO ;
```

creates an unnamed KEPT-DATA list containing the three members ACC-NO, TAX-CODE and EMP-NO. The members that directly use ACC-NO, TAX-NO and EMP-NO display.

If KEEP or ALSO is within a PERFORM command input-stream, then the generated index-names are kept.

If the Audit and Security facility is installed (selectable unit CMR-DD3) acceptance of the KEEP and ALSO commands is subject to:

- Access security levels, if the prefixed command is BULK PRINT, BULK REPORT, GLOSSARY, LIST, PERFORM, WHAT, WHICH or WHOSE
- Alter security levels and the dictionary's insertion security level, if the prefixed command is BULK ENCODE

in the manner defined for the prefixed command.

Refer to *ASG-ControlManager User's Guide* for details of the rules governing member names.

Refer to *ASG-Manager Products Status Concepts* for details of renaming members in a multi-status environment.

Refer to "Commands Which Manipulate KEPT-DATA Lists" on page 36 for details of how to use KEPT-DATA lists.

KEEP Syntax

```
>>-┬KEEP──────────────┬┬──────────────┬┬─────┬──>
   └ALSO──┬──────┬────┘└─IN list-name─┘└─AND─┘
          └─KEEP─┘
>──┬BULK command──────┬┬─;─┬──────────────────><
   ├GLOSSARY command───┤└─.─┘
   ├LIST command───────┤
   ├PERFORM command────┤
   ├WHAT command───────┤
   ├WHICH command──────┤
   └WHOSE command──────┘
```

where *list-name* is the name of a KEPT-DATA list.

LIST (CMR-CM01)

The LIST command lists all or a selection of index-names.

Refer to "LIST Syntax" on page 172 for the syntax of the LIST command.

Introduction to the LIST Command

The LIST command is used to list all or a selection of index-names. Use the various keywords with the LIST command to produce specific output.

To list all members in the repository, enter:

```
LIST ;
```

Do not use this form of the command unless you really need to, as processing can be very time-consuming and the output lengthy.

Alternatively, you can list a selection of index-names based on one or more of these:

- LOCKED: currently locked members
- CHECK-OK, CHECK-NEEDED: members that have passed or need integrity checks
- KEPT-DATA: index-names in a KEPT-DATA list
- MEMBERS: a list of named members
- INDEX-NAMES, ALIASES, CATALOG, ATTRIBUTES: all or specific types of index-name
- Including specific member types

- GENERIC: members of a type based on a specified member type
- EXCEPT: excluding specific types of index-name
- SOURCES: members with source record only
- DUMMIES, DUMMYS, DUMMY: dummy members.

You can further refine your selection by including one or more secondary selections:

- Name: selecting index-names whose names meet given criteria
- Status: selecting index-names which exist, or were processed in specific statuses
- Time: selecting index-names which were, or were not, processed at a given time
- User: selecting index-names which were, or were not, processed by a certain user.

The output from the LIST command consists of a list of selected index-names with details of their member type, condition (source or encoded), usage and information about protection.

You can list a member if your security level is insufficient for you to access that member, but only the member name and owner name are output. All other detail is suppressed.

You can include one of the following keywords in your LIST command to define the type of output you require:

- ALL-STATUSES: to apply the command to all the statuses currently available to you
- HISTORY: to output the details of when and by whom the index-name was inserted, encoded and, where applicable, last altered
- LOCK-INFORMATION: to list, by user, members that are locked or whose lock has expired.

You can specify how the information is to be displayed using the keywords:

- LOCKS: to indicate any locked members in the list. This output replaces the usage column
- NO-USAGE: to suppress the details in the usage column
- DETAILS-ONLY: to output only the list of index-names by suppressing headings and totals
- TOTALS-ONLY: to output only the total number of index-names processed
- ALPHABETICALLY: to list index-names in alphabetical order.

Listing Members in a Status Environment

Unless you specify otherwise, when you list an index-name it is listed from the current status. If the index-name does not exist in the current status it is listed from the nearest base status in which it does exist.

If you include the ALL-STATUSES keyword in your command, the index-name is listed from every status in which it exists and that you have access to.

For example, if you have a four-status repository that contains:

- A member named ACC-NAME with a source record in each of the four statuses, and
- A member named ACC-NO with a source record in two statuses

and you want to list each of these members from every status in which they appear, enter:

```
LIST ALL-STATUSES MEMBER ACC-NAME, ACC-NO ;
```

The output generated by the LIST command with the ALL-STATUSES keyword includes columns containing member name, member type, usage, condition, authority to access, alter and remove the member, and status.

For example, the above command generates the following output:

```
LIST OF MEMBERS
  MEMBER NAME             TYPE      USAGE CONDITION AU STATUS
  ACC-NAME                FILE          4  SCE ENC ACC 1987
                          FILE          4  SCE ENC ACC 1988
                          FILE          6  SCE ENC ACC 1989
                          FILE          5  SCE ENC ALT 1990
  ACC-NO                  ITEM          8  SCE ENC ACC 1989
                          ITEM         10  SCE ENC REM 1990
  LIST CONTAINS       2 ITEMS
                      4 FILES
                      2 MEMBERS IN TOTAL
```

If you list aliases, catalogs or indexed attributes with the ALL-STATUSES keyword, the condition and authority columns in the output are blank.

If you do not have access to the member(s) you have specified in the LIST ALL-STATUSES command, output is affected in the following ways:

- If you cannot access a member, only the member name is displayed. All details of TYPE, USAGE, CONDITION and AU (authority) are suppressed
- At the end of the LIST ALL-STATUSES output, the number of members omitted from the list is stated in addition to the total number of members listed. For example, if you cannot access two of the members you have specified, the following output appears in the LIST CONTAINS column:

```
2 MEMBERS INACCESSIBLE TO THIS USER
2 MEMBERS NOT SELECTED FOR OUTPUT
```

Listing Index-names With Details of When They Were Inserted and Encoded

If you specify the keyword HISTORY in the command then the history of each listed index-name is displayed, including details of when and by whom it was created and, where applicable, last encoded or altered.

To list all the members in the repository giving the history of each, enter:

```
LIST HISTORY ;
```

Do not use this form of the command unless you really need to, as processing can be very time consuming and the output lengthy.

You can further refine the output from LIST HISTORY by including additional selections.

For example:

```
LIST HISTORY KEPT-DATA ;
```

lists the history of index-names kept in the unnamed KEPT-DATA list.

The output produced by LIST HISTORY has a different format to that produced by other forms of the LIST command. For example: `LIST HISTORY MEMBER TAX-CODE ;` gives the following output:

```
HISTORY LIST OF MEMBERS
  TAX-CODE
    ACC YES  ALT YES  REM YES
    STATUS  UPDATE  DEV1
    MEMBER TYPE ITEM
       INSERTED  12 FEB 1990  11.23.04  BY PERSONNEL
       ENCODED  20 FEB 1990  16.48.52  BY MASTER
```

If you do not have the status facilities, the output displays STATUS UNNAMED.

If you do not have access to the member(s) you have specified in the LIST HISTORY command, output is affected in the following ways:

- If you cannot access a member, the member name is displayed, the ACC ALT and REM columns all contain NO and additionally the OWNER name appears. All other output is suppressed
- If the LIST HISTORY command is issued by the Controller, each ACC ALT and REM line will contain a security level.

Listing Members With Details of When and By Whom They Were Locked

To find out who has locked any members and when the locks are due to expire, enter:

```
LIST LOCK-INFORMATION ;
```

The output from this command includes all members that are currently locked and all members whose lock has not been released but has expired. The output is grouped by user; that is, all members locked by a particular user are listed under the name of that user, with the most recently locked member given first. The date and time that each lock is due to expire also displays.

For example, the above command generates the following output:

```
MEMBER LOCKED BY USER ACCOUNTS
  EXPIRY DATE            EXPIRY TIME                MEMBER NAME
  30 JUN 1990            15.25.42                   ACC-NO
  12 FEB 1990            12.20.35         EXPIRED   TAX-NO
```

Listing Currently Locked Members

To list all members which are currently locked, enter:

```
LIST LOCKED ;
```

For example, the command:

```
LIST LOCKED IF INTRODUCED BY ACCOUNTS ON '31-MAY-89';
```

lists all locked members entered in the repository on 31st MAY 1989, by a user (or department) whose password in the AUTHORITY command is ACCOUNTS.

Listing Named Members

To list one or more named members, enter:

```
LIST MEMBERS member-name-list ;
```

where *member-name-list* is one or more member names separated by commas.

Member names are listed in the order they are specified in the member-name-list, unless the keyword ALPHABETICALLY is used. If you specify the KEPT-DATA and MEMBERS keywords in your LIST command, then any members that fall within either category are listed. Members that appear in both categories are listed only once.

Listing Index-Names in a KEPT-DATA List

To list index-names in an unnamed KEPT-DATA list, enter:

```
LIST KEPT-DATA ;
```

To list index-names in a named KEPT-DATA list, enter:

```
LIST KEPT-DATA IN list-name ;
```

where *list-name* is the name of a KEPT-DATA list.

Index-names are listed in the order in which they appear in the KEPT-DATA list, unless the keyword ALPHABETICALLY is used.

If you specify the KEPT-DATA and MEMBERS keywords in your LIST command, then any members that fall within either category are listed. Members that appear in both categories are listed only once.

For example:

```
LIST KEPT-DATA IN ADDRESS MEMBERS RPT-PAGE, RPT-TITLE ;
```

lists the index-names in the KEPT-DATA list named ADDRESS followed by the members RPT-PAGE and RPT-TITLE. If either of the members RPT-PAGE and RPT-TITLE also appear in the KEPT-DATA list named ADDRESS, then they will be listed only once.

Listing Index-Names Alphabetically

If you specify the keyword ALPHABETICALLY in the LIST command, index-names will be listed alphanumerically.

For example:

```
LIST MEMBERS S-CODE, RPT-PAGE, RPT-TITLE ALPHABETICALLY ;
```

lists RPT-PAGE then RPT-TITLE followed by S-CODE.

Note:

The keyword ALPHABETICALLY only has an effect when used with the selection keywords MEMBERS and KEPT-DATA. With other selections, for example INDEX-NAMES, the list is automatically compiled in alphanumeric order. Although ALPHABETICALLY is accepted it has no effect.

Listing Source Members

To list all members with source but no encoded record, enter:

```
LIST SOURCES ;
```

For example:

```
LIST SOURCES ONLY ACC ;
```

lists only those source members whose name begins with ACC.

Listing Dummy Members

To list members which exist as dummy records in the repository, enter:

```
LIST DUMMIES ;
```

The category DUMMY includes all members with dummy encoded records. Some dummy members may also have an unencoded source record.

Note:

The keywords DUMMY, DUMMIES, and DUMMYS are all accepted, but cannot be truncated.

The keyword DUMMY has no effect in the processing of the categories ALIASES, CATALOGUES and ATTRIBUTES although it is accepted with them.

To exclude specific member types using the EXCEPT keyword, enter:

```
LIST DUMMIES EXCEPT member-type ;
```

For example:

```
LIST DUMMIES EXCEPT ITEMS;
```

lists all dummy members other than ITEMs.

Listing Real, that is, Non-dummy Members

To list all non-dummy members, that is, those which have been defined and encoded, enter:

```
LIST REAL ;
```

The keyword REAL may be used in conjunction with other keywords. For example, to list the history of all ITEM members in the repository, enter:

```
LIST HISTORY REAL ITEMS ;
```

Listing or Excluding Index-names and Member Types

To list all index-names, enter:

```
LIST INDEX-NAMES ;
```

To list specified types of index-names, enter:

```
LIST index-name-type ;
```

where *`index-name-type`* is the interrogate keyword for:

- Any member type or collective member type available in your repository
- The keywords ALIASES, CATALOGS and ATTRIBUTES.

Use the SHOW UDS command to list the member type interrogate keywords available in your repository.

For example, to list the indexed attributes of all encoded members, enter:

```
LIST ATTRIBUTES ;
```

To list all relationship members, enter:

```
LIST RELATIONSHIPS ;
```

To list all entity members, enter:

```
LIST EXCEPT RELATIONSHIPS ;
```

Note:
Your Systems Administrator can choose to rename the RELATIONSHIPS keyword.

When more than one index-name-type is specified each must be separated by a comma.

For example:

LIST ITEMS, GROUPS ;

lists all members with either GROUP or ITEM member type.

For member type and collective member types only members with encoded or dummy records are processed. For ALIASES, CATALOGUES and ATTRIBUTES only members with encoded records are processed.

To exclude specific member types from the LIST, enter:

```
LIST EXCEPT member-type ;
```

To exclude index-name-types from KEPT-DATA lists containing members and index-names, enter:

```
LIST KEPT-DATA EXCEPT index-name-type ;
```

For example:

```
LIST KEPT-DATA EXCEPT ITEMS ;
```

lists everything in the KEPT-DATA list except the ITEMs, and:

```
LIST KEPT-DATA EXCEPT ALIASES;
```

lists the contents of the KEPT-DATA list except the ALIASES.

Name, Status, Time, and User-Related-Selections

Refer to "Applying Commands to a Selection Of Index-names Based on Name, Status, Time and User" on page 293 for details of the name, status, time and user-related-selections.

Example Output

The LIST command lists index-names together with details about their member type, usage, condition and security.

For example, the command:

```
LIST INDEX-NAMES;
```

might generate the following output:

```
LIST OF MEMBERS
MEMBER NAME             TYPE       USAGE CONDITION  AC  ALT REM
ACCOUNT                 ALIAS          1
ACCT-NO                 MODULE         2 *    DUM  YES YES YES
EMP-NO                  ITEM
                                              PERSONNEL
MOD-EMP-HIST            MODULE         3 *SCE ENC  YES YES YES
SORT                    CATALOGUE      5
SYS-EMP-HIS             SOURCE           *SCE      YES YES YES
TEST                    ATTRIBUTE      1
LIST CONTAINS  1 ITEM
               2 MODULES
               1 CATALOGUE
               1 ALIAS
               1 ATTRIBUTE
               1 DUMMY
               1 CONTAINING SOURCE DETAILS ONLY
               7 MEMBERS IN TOTAL
```

Description of the Output

The LIST command lists index-names together with details regarding their type, usage, condition and security.

MEMBER NAME contains the names of members, aliases, catalogue classifications or indexed attributes.

TYPE contains the member type of the member or one of the keywords ALIAS, CATALOGUE or ATTRIBUTE.

USAGE is dependent on the TYPE of MEMBER NAME listed. It contains one of the following:

- For a member with an encoded or dummy record, the number of encoded members that directly use the listed member. For a member with a source record but no encoded record, the column is blank
- For an alias, the number of encoded members having that alias
- For a catalog classification, the number of encoded members catalogued under that classification
- For an indexed attribute, the number of encoded members having that attribute.

CONDITION is dependent on the TYPE of MEMBER NAME listed. It contains one of these:

- For an alias, catalog classification or indexed attribute, the column is blank
- For a member name, the column may contain:

```
SCE if the member has a source record
ENC if the member has an encoded record
DUM if the member has a dummy record
*   if the member has no encoded record or has a source record
       that has been changed since it was last encoded
DMR if the member is of an internal member type generated by a
       database management system interface.
```

LIST CONTAINS lists the total number of each TYPE of MEMBER NAME output and the number of these that are also dummy members.

With the Audit and Security facility, output for all listed member names includes:

- AC if you can access the member
- ALT if you can alter the member
- REM if you can remove the member
- The owner-name of the owner of the member

If you have sufficient authority to ACCESS, ALTER, or REMOVE a member each column will contain YES. If you have insufficient authority the column will contain NO.

The columns AC, ALT, REM, and OWNER are blank if you list aliases, catalog classifications or indexed attributes.

If you have insufficient authority to access a member only the member-name is given in the list. All other details are suppressed although owner-name is given if the member is owned.

At the end of the LIST output the total number of members accessible to the user and the total number of members inaccessible to the user are given in addition to the usual counts.

If the LIST command is issued by the Controller the AC, ALT, and REM columns contain the member's access, alteration and removal security levels.

Modifying the Output

You can:

- Exclude those parts of the output that you do not require by using the keywords NO-USAGE, DETAILS-ONLY and TOTALS-ONLY
- See which members are locked, using the LOCKS keyword
- See which members are check-needed, using the SET CHECK-CHARACTER command.

If you specify the keyword NO-USAGE in the command the USAGE column will be omitted from the output.

For example:

```
LIST ITEM NO-USAGE ;
```

lists all members of the member type ITEM, but will not give the number of members which use them. That is, the USAGE column will be blank.

If you specify the keyword LOCKS the usage column is replaced by the LOCKS column, which contains the word LOCK if a member in the list is currently locked.

For example:

```
LIST KEPT-DATA LOCKS ;
```

lists the index-names in the KEPT-DATA list indicating any locked members.

If you specify the keyword TOTALS-ONLY in the command, all information other than that displayed under LIST CONTAINS will be omitted from the output.

For example:

```
LIST ALIASES TOTALS-ONLY ;
```

outputs the total number of aliases of encoded members.

The NO-USAGE and TOTALS-ONLY keywords improve response time as they remove the requirement for the data entries dataset to be processed.

If you specify the keyword DETAILS-ONLY then headings, form feeds and LIST CONTAINS are omitted.

For example:

```
LIST KEPT-DATA DETAILS-ONLY ;
```

lists the index-names in the unnamed KEPT-DATA list but will not display the heading, line-spaces, or totals.

If you wish to see which members are check-needed then set the check-character using the SET CHECK-CHARACTER command. For example, the output from a LIST command when the check-character is + might be as follows:

```
LIST OF SELECTED MEMBERS
  MEMBER NAME                 TYPE       USAGE CONDITION AC ALT REM
  MOD2                         MODULE         3  SCE ENC+  0   0   0
  MOD1                         MODULE         1  SCE ENC   0   0   0

  DUM                          MODULE         1 *    DUM+  0   0   0
  LIST CONTAINS         3 MODULES
                        1 DUMMY
                        3 MEMBERS IN TOTAL
```

The module MOD1 has passed the checks and the module MOD2 and the dummy module DUM need checking.

Selecting Members That Have Passed or Need Integrity Checks

To select check-ok members, members that have passed integrity checks, include the CHECK-OK keyword in the command.

To select check-needed members, members that need integrity checks, include the CHECK-NEEDED keyword in the command.

For further details of check-ok and check-needed members, refer to *ASG-MethodManager Administration*.

Listing Members Based on a Specific Member Type

To list members of types based on a specific member type, enter:

```
LIST GENERIC member-type ;
```

where *member-type* is the interrogate keyword of any base member type.

LIST Command Examples

To list all FILEs with dummy records, enter:

```
LIST DUMMY FILES ;
```

To list all members whose member name begins with A, enter:

```
LIST ONLY A ;
```

To list all members except those with a member type of ITEM, that have a source record but no encoded record, enter:

```
LIST SOURCES EXCEPT ITEM ;
```

To list index-names whose names include the string EMPLOYEE, enter:

```
LIST INDEX-NAMES WHEN ANY = EMPLOYEE DETAILS-ONLY ;
```

The keyword DETAILS-ONLY suppresses headings, totals and form feeds.

To list the members ACCOUNT and MOD-EMP-HIST, enter:

```
LIST MEMBERS ACCOUNT, MOD-EMP-HIST ;
```

To list all members and display their history, enter:

```
LIST HISTORY ;
```

The output includes when and by whom the members were inserted and, where applicable, last encoded or altered.

To list index-names in a KEPT-DATA list named ADDRESS, enter:

```
LIST KEPT IN ADDRESS ;
```

To list the index-names from both the unnamed KEPT-DATA list and a list of named members, in alphabetical order, enter:

```
LIST KEPT-DATA MEMBERS TAX-NO, EMP-NO, REF-NO ALPHABETICALLY ;
```

To list check-ok relationship members, enter:

```
LIST CHECK-OK RELATIONSHIPS ;
```

To list members of type PROGRAM, or of any type based on type PROGRAM, enter:

```
LIST GENERIC PROGRAMS ;
```

LIST Syntax

```
>>---LIST ---+---list-clause ------------------------------+---+---;---+---><
             |   LOCK-INFORMATION                           |   |   .   |
             |                  <<< , <<                    |
             |                  MEMBERS mem-name            |
```

where *list-clause* is:

```
>---+---------------------+---<<<<<<<<<< , <<<<<<<<<<<---+---------------------+---><
    |  HISTORY            |                              |  status-related-selection  |
    |  ALL-STATUSES       |                              |  LOCKED                    |
    |                     |                              |  CHECK-OK                  |
    |                     |                              |  CHECK-NEEDED              |

>---+---------------+---+---------------------+---><
    |  selection    |   |  ALPHABETICALLY     |

>---+---------------------------+---><
    |  name-related-selection   |

>---+---------------------------------------+---><
    |  time-and-user-related-selection      |

>---+---+---------------+---+---------+---+---+---><
    |   |  DETAILS-ONLY |   |  NO-USAGE |   |  LOCKS |
    |   |               |   |         |   |         |
    |   |  TOTALS-ONLY  |
```

where:

status-related-selection is defined in "Applying Commands to a Selection Of Index-names Based on Name, Status, Time and User" on page 293

selection is:

```
>---+---+---------------------+---><
    |   |  KEPT-DATA          |
    |   |                     |
    |   |     IN list-name    |

>---+---+---------------+---><
    |   |  <<< , <<     |
    |   |  MEMBERS mem-name |

>---+---+---+---+---+---+---+---+---+---><
    |   |   |   |   |   |   |   |   |
    |   |   |   |   |   |   |   |   |
    |   |   |   |   |   |   |   |   |
```

where:

list-name is the name of a KEPT-DATA list

mem-name is the name of a repository member

index-name-type is:

```
➤──┬──┬──────────┬──────────────
```

If the Basic or Advanced Status facility is installed (selectable units CMR-DD2 and CMR-AD21 respectively) the new member must not already have a source record in any status in the dictionary. The member is created and locked in the current status.

Refer to *ASG-ControlManager User's Guide* for member naming rules.

How to Lock an Existing Member

To lock an existing dictionary member to prevent updating of that member by other users, enter:

```
LOCK EXISTING member-name ;
```

member-name is an existing dictionary member which has a source record.

If you wish to lock a member which has no source record, use the LOCK NEW command.

If the Basic or Advanced Status facility is installed (selectable units CMR-DD2 and CMR-AD21 respectively) the member to be locked must have a source record in the current status or in a direct or indirect base status. If the member does not exist in the current status, the LOCK EXISTING command copies the member's source record from the nearest direct or indirect base status in which it does exist, into the current status, and then locks it.

The LOCK EXISTING command does not encode source records which are copied into the current status from a base status. If you wish to encode the member in the current status, use the ENCODE command or the UPDATE and FILE/RFILE commands.

A LOCK EXISTING command is rejected if:

- The member to be locked is already locked in the current status

or

- The member does not exist in the current status and is locked, by another user, in the nearest direct or indirect base status from the current status, in which it does exist.

How to Renew a Lock

To renew a lock which has expired, or to extend the period of a current lock so as to prevent updating of the locked member by other users, enter:

```
LOCK RENEW member-name ;
```

When extending a current lock, the period of the latest lock replaces that of the existing lock: the two lock periods are not added together.

The advantage of using the LOCK RENEW command instead of the LOCK EXISTING command when you wish to renew an expired lock is that a LOCK RENEW command cannot be accepted if the member has been updated since its lock expired. Therefore, if your LOCK RENEW command is accepted, you can be certain that the member in question has not changed since you last updated it.

If the Basic or Advanced Status facilities are installed (selectable units

CMR-DD2 and CMR-AD21 respectively) the member whose lock is to be renewed, must exist in the current status. The member is only locked in the current status.

The LOCK RENEW command is only accepted if:

- Entered by the last user to lock the member with a LOCK NEW, LOCK EXISTING or LOCK RENEW command

and

- The last lock was not released with a LOCK RELEASE command

and

- The source or encoded record of the member has not been changed by any user since the lock expired.

How to Release a Lock on a Member

To release a lock on a member and make the member available to all users for updating, enter:

```
LOCK RELEASE member-name ;
```

The LOCK RELEASE command is only accepted if entered by the user who last locked the member with a LOCK NEW, LOCK EXISTING or LOCK RENEW command, or by the dictionary Controller, who can release any locks at any time.

If the LOCK RELEASE command is used to release either a current or an expired lock, that lock cannot be renewed with a LOCK RENEW command.

If a lock is not released with the LOCK RELEASE command, it will eventually expire when the lock period specified for the user, or for the dictionary, expires. The member will continue to be included in lists of locked members and the lock can be renewed (using the LOCK RENEW command) by the user who issued the lock which has expired.

It is recommended that the dictionary Controller checks that expired locks, which are no longer required, are released (using the LOCK RELEASE command) on a regular basis. All members locked by a particular user can be released simultaneously with the PERFORM command. For example, the command:

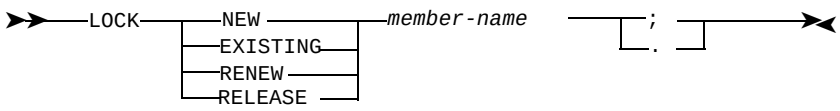
```
PERFORM 'LOCK RELEASE "*"' LOCKED IF INTRODUCED BY
                           ACCOUNTS ON '31-MAY-89' ;
```

releases the locks on all locked members entered in the dictionary by the user (or department) whose password in the AUTHORITY command is ACCOUNTS on the 31st May, 1989.

If the Basic or Advanced Status facilities are installed (selectable units CMR-DD2 and CMR-AD21 respectively) the member whose lock is to be released, must exist in the current status. If the member is currently locked in more than one status, only the lock in the current status is released.

Refer to *ASG-Manager Products Controller's Manual* for details of how the dictionary Controller can control member locking.

LOCK Syntax



where *member-name* is the name of a dictionary member.

MODE, QUERY MODE, DMS, WBDA (CMR-CM01)

The MODE, QUERY MODE, DMS, or the WBDA command sets the environment to which a command should be routed, where cross-product command ambiguity exists.

The MODE command defines the Targeting Mode during the current session or (when it is placed in Executive Routines or in Logon, Global or User Defined Profiles) as a default.

To define Dictionary Management Software as the Targeting Mode, enter:

```
MODE DMS ;
```

or

```
MODE DICTIONARY-MANAGEMENT-SOFTWARE ;
```

To define the Workbench Design Area as the Targeting Mode, enter:

```
MODE WBDA ;
```

or

```
MODE DESIGN-AREA ;
```

To find out the current Targeting Mode, enter:

```
QUERY MODE ;
```

If, for a particular command, you want to override the current Targeting Mode without actually changing it, simply prefix the command with one of the following prefix commands:

- DMS (to apply the command to the current dictionary)
- WBDA (to apply the command to the Workbench Design Area).

If a Targeting Prefix is used with a command that does not require one or actually duplicates the current Targeting Mode, it will be ignored and will not prevent the command input being executed.

The default Targeting Mode is DICTIONARY-MANAGEMENT-SOFTWARE.

Refer to "Targeting Commands" on page 40 for a description of the use of Targeting commands.

MODE Syntax

```
>>---MODE---+-DICTIONARY-MANAGEGEMENT-SOFTWARE-+---+---;---+---><
            |-DMS                              |   |   .   |
            |-DESIGN-AREA                      |
            |-WBDA                             |

>>---QUERY MODE---+---;---+---><
                  |   .   |

>>---DMS command---+---;---+---><
                   |   .   |
```

where *command* is the name of a Manager Products command.

```
>>---WBDA command---+---;---+---><
                    |   .   |
```

where *command* is LIST, MERGE or REPORT when DesignManager is installed.

MODIFY (CMR-CM01)

The MODIFY command modifies the source record of an individual member and encodes the modified source record.

Refer to "MODIFY Syntax" on page 193 for the syntax of the MODIFY command.

Introduction to the MODIFY Command

The MODIFY command enables you to modify the source record of a member, generate an encoded record from the modified source record, and insert this encoded record into the dictionary's Data Entries dataset.

The MODIFY command is entered in two stages.

The first stage identifies the member to be modified. You can also specify one of the following optional keywords:

- WIPE-OUT deletes a member's existing source record
- RENUMBER renumbers the lines within a member's source record

For example:

```
MODIFY PROJECT-NO WIPE-OUT ;
```

deletes PROJECT-NO's source record.

The second stage comprises one or more amendment lines.

An amendment line can be:

- An insertion line or replacement line. These lines either have no initial keyword, but commence with a line number, or commence with the keyword LINE.
- An editing instruction, for example, BLANK, CHANGE, COPY, DELETE, FILL or MOVE
- A control instruction, for example, ACCEPT, DISPLAY, END, LOCATE, QUIT, RENUMBER, a semi-colon or a full stop.

You can enter amendment lines in any order, but each amendment line must be wholly contained on one input-line.

You must enter the first element of each amendment line (that is, a line number, LINE or any of the editing or control instructions listed above) in the column 1. You must also enter the final terminator in column 1.

For example:

```
10 ITEM
20 HELD-AS NUMERIC 5
;
```

creates a new source record for PROJECT-NO with line numbers increasing in steps of ten and encodes it.

When you enter the final terminator of the MODIFY command, the following information is always displayed:

- The member’s new source record
- A message indicating wether or not the member has successfully encoded.

If you enter the syntax of the first stage incorrectly, the second stage is not processed.

If you enter the syntax of the first stage correctly, the member you are modifying is still available to other users for updating purposes while you enter the amendment lines in the second stage.

If another user does update the member between acceptance of the first stage and completion of the second stage of your MODIFY command, the amendment lines are abandoned and the entire MODIFY command is rejected. Processing continues with the next command.

Use the LOCK or PROTECT commands before the MODIFY command to prevent other users concurrently updating a member that you want to MODIFY.

You can only MODIFY a member if your security level is equal to or higher than the dictionary's insertion security level and is sufficient for you to alter that member and access any members which the modified member uses.

If you cannot access any member to which the MODIFYed member directly refers, the source of the modified member is filed but not encoded.

MODIFY is treated as an updating command by the automatic recovery system.

Note:

- Details relating to locking only apply if the Workstation Interface facility (selectable unit CMR-WS01) is installed.
- Details relating to security levels only apply if the Audit and Security facility (selectable unit CMR-DD3) is installed.
- Details relating to statuses only apply if either the Basic or Advanced Status facilities (selectable units CMR-DD2 and CMR-AD21 respectively) are installed.

Refer to "ENCODE (CMR-CM01)" on page 129 for details on what happens when a member is encoded.

Refer to *ASG-Manager Products Status Concepts* for details of updating members in a multi-status environment.

Refer to "The Security System" on page 17 for details of the Security System.

Renumbering Lines (RENUMBER)

To renumber all the lines of a member's source record and change each line number by a specific integer, commencing with that integer, enter:

```
MODIFY member-name RENUMBER n ;
```

where:

member-name is the name of the dictionary member to be altered.

n is an unsigned integer with a value of 10000 or less.

For example, if *n* is 50, the lines of a member's new source record are renumbered in steps of 50, commencing at 50.

If you do not include an integer in your command, 100 is assumed.

You can also include a RENUMBER amendment line in the second stage of your command to renumber specific lines within the member's source record.

If you want to enter more than one of the optional keywords RENUMBER or WIPE-OUT, you can include them in the first stage of the command in any order.

Deleting a Member's Current Source Record (WIPE-OUT)

To delete a member's current source record so that you can redefine the entire member, enter:

```
MODIFY member-name WIPE-OUT ;
```

where *member-name* is the name of the dictionary member to be altered.

The optional keyword WIPE-OUT is useful if, for example, you want to rewrite a member's entire definition and reorganize its clauses so that each clause appears on a specific line number.

If you want to enter more than one of the optional keywords RENUMBER or WIPE-OUT, you can include them in the first stage of the command in any order.

Inserting a New Line

To insert a new line, include the amendment line:

```
m source-line
```

where:

m is the number of the new line, followed by a space.

source-line is the line to be inserted. It is assumed to begin immediately after the space.

If *m* identifies an existing line, that line is replaced by the new line (source-line).

m must be followed by a space. *source-line* is assumed to begin immediately after the space.

To insert a line of spaces, include:

```
m
```

Deleting a Line (DELETE)

To delete a line, include the amendment line:

```
DELETE line-number
```

To delete lines with numbers in the range *m* to *n* include:

```
DELETE m THRU n
```

THRU can be replaced with the alternative keyword TO.

To delete the first line in the member containing a given string, include:

```
LOCATE 'string'
DELETE LINE
```

Replacing a Line

To replace a line, include the amendment line:

```
m source-line
```

where:

m is the number of the line you want to replace.

source-line is the new replacement line.

m must be followed by a space. *source-line* is assumed to begin immediately after the space.

If the replacement line (source-line) is shorter than the line it replaces, the rest of the line is filled with spaces.

To replace a line with spaces, include:

```
m
```

To replace each line numbered in the range m through n with a character string, include:

```
FILL m THRU n 'string'
```

To replace a line numbered *n* by a line numbered *m*, include:

```
MOVE m TO n
```

Line *m* is moved to line *n* and is deleted from its original position.

To replace a line with a copy of another line, include:

```
COPY m TO n
```

The line numbered *n* is replaced by a copy of the line numbered *m*.

To replace the first line in the member containing a given string of characters, include:

```
LOCATE 'string'
LINE source-line
```

You cannot alter the whole of a line which is already completely full by a replacement line (unless a different input device which has a longer record length than that used for the original insertion of the line is used).

The reason for this is that the replacement line must include the line number, or the keyword LINE, together with a space before the replacement text can commence. Thus, if you want to replace text on line 200 (in which every character space is currently used), you would enter:

```
200 source-line
```

As 200 takes up four character spaces, you cannot specify new text for the last four character spaces of the line. These four characters will appear as blank spaces.

If you want to change these spaces to text you should use either the CHANGE or the FILL amendment lines.

Copying a Line (COPY)

To copy a line, include the amendment line:

```
COPY line-number BEFORE r
```

where:

> *line-number* is the number of the line to be copied.
>
> *r* is a line number identifying the position before which the copied line is to be inserted. The line which is copied is left unchanged in its original position.

To copy lines with numbers in the range *m* to *n* inclusive, include:

```
COPY m THRU n BEFORE r
```

To copy the first line in the member containing a given string, include:

```
LOCATE 'string'
COPY LINE BEFORE r
```

To insert a line before the first line containing a given string, include:

```
LOCATE 'string'
COPY line-number BEFORE LINE
```

To insert the line(s) after *r* instead of before, replace BEFORE with AFTER in the formats given above.

To replace a line numbered *r* with a copy of another line numbered *m* include:

COPY *m* TO *r*

The maximum number of lines that can be inserted by a COPY amendment line is 32,767.

The line numbering of the inserted lines is controlled automatically. If necessary, existing lines may also be renumbered.

Moving a Line (MOVE)

To move a line, include the amendment line:

MOVE *line-number* BEFORE *r*

where:

line-number is the number of the line to be moved.

r is a line number identifying the position before which the line is to be inserted.

It does not need to be the number of an existing line.

The line which is moved is deleted from its original position.

To move lines with numbers in the range *m* to *n* inclusive, include:

MOVE *m* THRU *n* BEFORE *r*

where *r* must not be within the range of lines to be moved.

To move the first line in the member containing a given string, include:

```
LOCATE 'string'
MOVE LINE BEFORE r
```

To move a line before the first line containing a given string, include:

```
LOCATE 'string'
MOVE line-number BEFORE LINE
```

To insert the line(s) after *r* instead of before, replace BEFORE with AFTER in the formats given above.

The numbering of inserted lines is controlled by default. If necessary existing lines may also be renumbered.

To move a line numbered *m* to replace a line numbered *r*, include:

```
MOVE m TO r
```

The maximum number of lines that can be inserted by a MOVE amendment line is 32,767.

Replacing a String (CHANGE)

To change all occurrences of a string to another string, include the amendment line:

```
CHANGE ALL 'string-1' TO 'string-2'
```

where:

string-1 is the string to be changed.

string-2 is the new string and may be shorter or longer than *string-1*.

For example, to delete all occurrences of the string APRIL from a member, include:

```
CHANGE ALL 'APRIL' TO ' '
```

To change all occurrences of a string in a given line, include:

```
CHANGE line-number ALL 'string-1' TO 'string-2'
```

To change all occurrences of a string in lines with numbers in the range *m* to *n* inclusive, include:

```
CHANGE m THRU n ALL 'string-1' TO 'string-2'
```

If you do not include the keyword ALL in the above amendment line, only the first occurrence of the string in each line is changed.

For example, to change the first occurrence of the string JAN to the string FEB, for each line in which it occurs, enter:

```
CHANGE 'JAN' TO 'FEB'
```

To change the first occurrence of a string in the member, include:

```
LOCATE 'string-1'
CHANGE LINE 'string-1' TO 'string-2'
```

Filling a Line or Column with Spaces (BLANK)

To fill the whole of a line with spaces, include the amendment line:

```
BLANK line-number
```

To blank out every line with numbers in the range *m* to *n* inclusive, include:

```
BLANK m THRU n
```

To blank out columns a to b of every line, include:

```
BLANK COLUMNS a THRU b
```

If THRU b is omitted, only column *a* is blanked out.

To blank out columns a to b in a block of lines, include:

```
BLANK m THRU n COLUMNS a THRU b
```

If THRU *n* is omitted, only the specified columns in the line numbered *m* are blanked out.

To blank out the first line in the member containing a given string, include:

```
LOCATE 'string'
BLANK LINE
```

Filling Columns of Text with a Specified String (FILL)

To replace columns a to b of every line with a specified string, include the amendment line:

```
FILL COLUMNS a THRU b 'string'
```

If string is shorter than the columns to be filled, the string is extended with spaces. If string is longer, the amendment-line fails. If you omit THRU b, only column a is replaced and string must be a single character.

To replace columns a to b in a given line with a string, include:

```
FILL line-number COLUMNS a THRU b 'string'
```

To replace columns in lines with numbers in the range *m* to *n* inclusive with a string, include:

```
FILL m THRU n COLUMNS a THRU b 'string'
```

To fill columns in the first line of the member containing a given string, include:

```
LOCATE 'string-1'
FILL LINE COLUMNS a THRU b 'string-2'
```

Checking Changes Before they are Implemented (DISPLAY)

To check any changes that you have made using amendment lines before the member is copied into the source dataset, include one of the following amendment lines.

To print or display the whole of the working storage record, include the amendment line:

```
DISPLAY
```

To print or display lines with numbers in the range *m* to *n* inclusive, include:

```
DISPLAY m THRU n
```

If you omit THRU *n*, only the line numbered *m* is output.

To output the whole of the record from a given line onwards, include:

```
DISPLAY m THRU END
```

To output the first line in the member containing a given string, include:

```
LOCATE 'string'
DISPLAY LINE
```

The CHANGE, FILL or BLANK amendment lines can be followed by PRINT to display the changes made by the lines, for example:

```
CHANGE ALL 'string-1' TO 'string-2' PRINT
```

In interactive environments you can correct a mistake, or abandon the amendment by including QUIT. In batch environments DISPLAY followed by QUIT allows you to check the effect of the amendments without them being executed.

For example:

```
MODIFY EMP-IDENT ;
CHANGE 'ABC' TO 'DEF'
DISPLAY
QUIT
```

displays:

- The amendment line, followed by the number of changes made
- The member, with all instances of ABC changed to DEF

without altering the source record of EMP-IDENT.

If you are happy with the result, you can run the amendments again without the DISPLAY and the QUIT.

The LOCATE Feature

LOCATE scans the whole or part of the member for a character string. If the string is found, the number of the line containing it is stored in memory and is printed or displayed. If an amendment line subsequently refers to LINE, the amendment is made to the LOCATEd line whose number is stored in memory.

For example:

```
LOCATE 'ABC'
DELETE LINE
```

deletes the first line in the member containing the character string, ABC.

To scan the whole of the member and to output every line in which the specified string is detected, include the amendment line:

```
LOCATE ALL 'string'
```

No line number is stored in memory.

To locate the first occurrence of a string in lines with numbers in the range *m* to *n* inclusive, include:

```
LOCATE m THRU n 'string'
```

To locate the first occurrence of a string from line *m*, include:

```
LOCATE m THRU END 'string'
```

To locate the first occurrence of a string from the line located by the last LOCATE command, include:

```
LOCATE NEXT 'string'
```

To locate the next occurrence of the string specified in the last LOCATE command, include:

```
LOCATE NEXT
```

The number of the line stored in memory by LOCATE is updated if the line is renumbered by a RENUMBER, MOVE or COPY amendment line. If the line is MOVEd, the line number is deleted from memory.

If the string is not found, an error message is output in batch mode and a warning message in online mode. The member will not be filed or encoded unless the MODIFY command is terminated by the ACCEPT keyword.

If you use the ACCEPT keyword, the syntax of the altered member is not checked for errors; its use may result in a corrupted source member.

Renumbering Lines (RENUMBER FROM, THRU, TO)

To renumber lines with numbers in the range *n1* to *n2* inclusive, in steps of *n3*, include the amendment line:

RENUMBER *n1* THRU *n2* IN *n3*

where:

> *n1* and *n2* are line numbers.
>
> *n3* is an unsigned integer defining the increment by which line numbers are increased.

The number of the last renumbered line must not be greater than the number of the line following *n2*.

For example, to renumber the lines numbered 1600 to 3800 in increments of 200, include the amendment line:

```
RENUMBER 1600 THRU 3800 IN 200
```

To renumber all lines in the range *n1* to *n2* commencing with the integer *n3* in increments of *n4*, enter:

```
RENUMBER n1 THRU n2 FROM n3 IN n4
```

where:

n1 and *n2* are line numbers.

n3 is an unsigned integer defining the line number of the first of the newly renumbered lines.

n4 is an unsigned integer defining the increment by which line numbers are increased.

When renumbered, *n2* must be less than the number of the following line.

The integer *n3* must be greater than the number of the line preceding *n1*.

For example, to renumber ten lines numbered 10 to 100 in increments of 2, commencing with the number 50, include the amendment line:

```
RENUMBER 10 THRU 100 FROM 50 IN 2
```

If, when the member's new source record is renumbered, the increment is such that the new line numbers are greater than five digits long, only the last five digits of each line display.

If you need to display the full line numbers use a DISPLAY amendment line, whereupon all eight digits display on any output.

If you renumber a member's line numbers, any amendment lines that you enter subsequently must take account of the new numbering.

You can optionally include leading zeros in your RENUMBER amendment line, provided that the number you specify does not exceed eight digits.

If you do not include RENUMBER in your command, the line numbering is controlled automatically.

The following is an example of how lines are renumbered when you enter a COPY amendment line without also entering a RENUMBER amendment line.

If you enter:

```
COPY 200 THRU 400 AFTER 600
```

the lines inserted are numbered as follows:

```
100 XXX     ====>     100 XXX
200 CCC               200 CCC
300 CCC               300 CCC
400 CCC               400 CCC
500 XXX               500 XXX
600 XXX               600 XXX
700 XXX               610 CCC
                      620 CCC
                      630 CCC
                      700 XXX
```

If the new lines cannot be contained within the existing line numbering, subsequent lines are also renumbered in steps of 100.

If you do not want to adopt the automatically inserted line numbers, you can use a RENUMBER amendment line to define your own line numbering system for the affected lines.

For example, if you want to move ten lines numbered 10 to 100 after the line currently numbered 800, enter:

```
MOVE 10 THRU 100 AFTER 800
```

The inserted lines are automatically renumbered 900 to 1800.

To specify that you want the steps between the inserted lines to remain as 10, enter:

```
RENUMBER 900 THRU 1800 IN 10
```

Terminating a MODIFY Command

To terminate your MODIFY command and implement the changes specified in the amendment lines included in the command, enter one of the following amendment lines:

```
ACCEPT
END
;
.
```

In this context, END is interpreted as an amendment terminator, not as a LOGOFF command.

To terminate your MODIFY command and abandon any changes specified in the amendment lines included in the command, enter:

```
QUIT
```

This message is output on acceptance of the QUIT amendment line:

```
DM01165E AMENDMENT ABORTED
```

You should enter the END, ACCEPT and QUIT amendment lines in upper-case.

Examples of the MODIFY Command

To insert a new line in EMP-SAL-AFT-UPD and then renumber the member's lines in steps of 100, enter:

```
MODIFY EMP-SAL-AFT-UPD RENUMBER;
1150 'AUTHOR DBA'
;
```

To make and encode the following changes to EMP-DEPT-INFO,

- Replace every occurrence of the word EMPLOYEE in lines 900 to 1200 by EMP
- Display the modified lines on the primary output device
- Delete the first line in which the word COBOL appears

enter:

```
MODIFY EMP-DEPT-INFO ;
CHANGE 900 THRU 1200 ALL 'EMPLOYEE' TO 'EMP'
DISPLAY 900 THRU 1200
LOCATE 'COBOL'
DELETE LINE
;
```

To renumber all the lines of a member named ENTRY-MGT in steps of 50, with the first line being numbered 50, enter:

```
MODIFY ENTRY-MGT RENUMBER 50 ;
```

To erase the contents of the current source record of a member named SALE-STAT-INST, enter:

```
MODIFY SALE-STAT-INST WIPE-OUT ;
```

MODIFY Syntax

```
>>─┬─MODIFY─┬─member-name──────────────────────────────────>
   └─MOD────┘
>──┬──────────────────────────────────┬──┬─;─┬──────────────><
   │  ┌──────────────┐                │  └─.─┘
   ├──RENUMBER──┬──────────┬──────────┤
   │            └─integer──┘          │
   └──WIPE-OUT──────────────────────┘
>──┬─<<<<<<<<<<<<<<<──────────────────────────────────────>
   └─ammendment-line──┘
>──┬─ACCEPT─┬──────────────────────────────────────────><
   ├─QUIT───┤
   ├─END────┤
   ├─;──────┤
   └─.──────┘
```

where:

member-name is the name of a dictionary member

integer is an unsigned integer in the range 1 to 10000

amendment-line is:

```
>──┬─┬─m────┬──┬──────────────┬──────────────────────────>
   │ └─LINE─┘  └─source-line──┘
   ├─DELETE──┬─LINE──────────────────────────────────────┤
   │         └─m──┬──────────┬──────────┬─n──┬──┤
   │              ├─THRU────┤          │    │
   │              └─TO──────┘          │    │
   ├─MOVE──┬─┬─LINE──────────────┬──┬─BEFORE─┬──┬─r────┬──┤
   │ └─COPY─┘ └─m──┬──────────┬──┘  ├─TO─────┤  └─LINE─┘
   │               └─THRU n───┘     └─AFTER──┘
   ├─CHANGE change-amendment-line──────────────────────┤
   ├─FILL fill-amendment-line──────────────────────────┤
   ├─BLANK blank-amendment-line────────────────────────┤
   ├─DISPLAY──┬──────────────────────────────────────┤
   │          ├─m──┬──────────┬──────────┬──┤
   │          │    └─THRU──┬─n──┬──┤    │
   │          │            └─END──┘       │
   │          └─LINE──────────────────────┘
   ├─LOCATE──┬──┬──────────────────────────────────┬─'string'──┬──┤
   │         │  └─m──┬──────────┬──────────┬──┤  │           │
   │         │       └─THRU──┬─n──┬──┤    │  │           │
   │         │               └─END──┘       │  │           │
   │         ├─NEXT──┬──────────┬──────────┤  │           │
   │         │       └─'string'─┘          │  │           │
   │         └─ALL 'string'────────────────┘  │           │
   ├─RENUMBER m THRU n──┬──────────┬──IN i──────────────┤
   │                    └─FROM v───┘
```

where:

m, *n*, *r*, and *v* are unsigned integers in the range of 1 to 16777215, being line numbers.

source-line is a member definition statement line appropriate to the member-type.

change-amendment-line is:

```
 m ──┬──────────┬──┬──────────┬──────────────────────────────────────────
     │ THRU n   │  │ ALL      │ 'string-1'  ┬──────┬
     │          │  │          │             │ TO   │
     ├──────────┤  │          │             │      │
     │ LIE      │  │          │             │      │
     │          │  │          │             │      │
     │          │  │          │             │      │
     │          
```

PERFORM (DMR-DD1)

The PERFORM command applies commands to a selection of index-names.

Refer to "PERFORM Syntax" on page 207 for the syntax of the PERFORM command.

Introduction to the PERFORM Command

The PERFORM command executes one or more commands defined in an input-stream. An input-stream consists of either a series of PERFORM input-lines, or commands contained in a COMMAND-STREAM member.

The PERFORM command can include any Manager Products command. PERFORM commands can be nested to any depth with the User Defined Commands facility.

By specifying a Perform Character, you can apply the PERFORM command to a selection of index-names by using one or more of these keywords:

- LOCKED: currently locked members
- KEPT-DATA: index-names in a KEPT-DATA list
- MEMBERS: a list of named members
- Specific types of member
- INDEX-NAMES, ALIASES, CATALOGS and ATTRIBUTES: all or specific index-names
- EXCEPT: excludes specific types of member
- SOURCES: members with source record only
- DUMMIES/DUMMY/DUMMYS: dummy members
- CHECK-OK, CHECK-NEEDED: members that have passed or need integrity checks
- GENERIC: members based on a member type.

Use the keyword ALPHABETICALLY to process index-names in alphanumeric order.

You can further refine your selection by including one or more secondary selections:

- Name: selecting index-names whose names meet given criteria
- Status: selecting index-names that exist, or were processed in specific statuses
- Time: selecting index-names that were, or were not, processed at a given time or during a given period of time
- User: selecting index-names that were, or were not, processed by a certain user.

The PERFORM command applies the selection criteria to all statuses, including those not visible from the current status. However, PROTECT and COPY FROM are the only commands applied to the members in all statuses. All other commands are only applied to members visible from the current status.

If a member exists, but is not visible from the current status, then error messages are output.

You can display the specified commands to be performed for each index-name, before they are executed, using the following keywords:

- PRINT: to display each command before the output it generates
- PRINT FIRST: to display all the commands before the output they generate
- DISPLAY-ONLY: to display all the commands without performing them.

If a Perform Character ('*' by default) and selection are specified, then the PERFORM command produces separate output for each index-name selected.

For example:

```
PERFORM 'LIST MEMBER *' DUMMIES;
```

outputs several lists, each containing one dummy member.

Performing Commands in a COMMAND-STREAM Member

To perform a COMMAND-STREAM member, enter:

```
PERFORM COMMAND-STREAM member-name ;
```

where *member-name* is the name of a COMMAND-STREAM member whose encoded record contains the commands to be executed.

For example:

```
PERFORM COMMAND-STREAM COMX KEPT;
```

executes the COMMAND-STREAM COMX for index-names in the unnamed KEPT-DATA list.

Note:

Each command specified in a COMMAND-STREAM member must have a terminator.

Performing Commands in Input-Lines

An input-line is any Manager Products command, and may contain an asterisk where a member-name, alias, catalog classification or indexed attribute would normally occur.

To execute a PERFORM input-line, enter:

```
PERFORM 'input-line' ;
```

You can specify several input-lines. Each input-line must be within delimiters, and separated by commas.

You can apply a PERFORM command to a selection of index-names by using a Perform Character in the input-line. For example:

```
PERFORM 'LIST MEMBER * ', 'WHAT DIRECTLY USES * ' KEPT-DATA ;
```

displays a list of all the members in the unnamed KEPT-DATA list and then displays those members that directly use each member in the KEPT-DATA list.

If a character string cannot be wholly contained on one line of the command area, you can divide it into two or more delimited strings. These will be concatenated to form a single character string when the command is performed. To prevent words from being merged by concatenation, insert a space between the last character of the string and the delimiter.

For example:

```
PERFORM
    'GLOSSARY MEMBER * GIVING '
    ,'ALIAS'
    KEPT-DATA ;
```

You do not need to include terminators at the end of input-lines as they are added automatically, unless the input-lines contain an ADD, INSERT, REPLACE, ALTER or MODIFY command, in which case the following special rules apply:

- Each element of an ADD, INSERT, REPLACE, ALTER or MODIFY command that would normally appear on a separate line must appear in a separate input-line
- Terminators must be included in the usual places. In particular, the terminator which usually appears on a separate line at the very end of the command must appear in the first character position of a separate input-line
- NO-TERMINATOR must be inserted after the selection.

For example:

```
PERFORM
    'MODIFY * ;'
    ,'LOCATE "ABC"'
    ,'CHANGE LINE "ABC" TO "DEF"'
    ,';'
    KEPT-DATA NO-TERMINATOR ;
```

modifies the source record of the members in the unnamed KEPT-DATA list, changes the first occurrence of the character string ABC to DEF and if no syntax errors are discovered, encodes the new record.

The systems administrator can specify a character in addition to ' and " to act as a delimiter. The QUERY STRING-DELIMITER command displays the current delimiters.

Replacing Index-Names with the Perform Character

The Perform Character indicates where an index-name selected by the PERFORM command is to be inserted.

The default Perform Character is an asterisk (*).

For example:

```
PERFORM
    'WHAT CONSTITUTES * '
    MEMBERS ORDER-CODE, CUST-CODE ;
```

displays the members that constitute ORDER-CODE and CUST-CODE.

The Perform Character must be within delimiters if the index-name it represents contains characters from the extended character set.

Delimiters around a Perform Character must be different from those surrounding the input-line.

For example:

```
PERFORM
    'WHAT DIRECTLY USES "*"'
    ,'WHAT DIRECTLY CONSTITUTES "*"'
    KEPT-DATA ;
```

selects the members that directly use and are directly used by each member in the unnamed KEPT-DATA list, regardless of whether the member name of the members held in the KEPT DATA list are made up of characters from the standard or extended character set.

The QUERY PERFORM-CHARACTER command displays the current Perform Character.

The SET PERFORM-CHARACTER command allows you to set a different Perform Character.

Refer to *ASG-ControlManager User's Guide* for details of:

- The standard and extended character set
- The QUERY STRING-DELIMITER command
- The SET and QUERY PERFORM CHARACTER commands.

Selecting Members that are Locked

To select all members that are currently locked, enter:

```
PERFORM input-stream LOCKED ;
```

where *`input-stream`* is a sequence of PERFORM input-lines.

For example:

```
PERFORM
    'WHAT DIRECTLY CONSTITUTES * '
LOCKED ;
```

lists all the members directly used by LOCKED members.

Note:

If you locked the member all commands can be executed. If another user locked the member, updating commands are inhibited.

Selecting Members with Source Record

To select all members with a source record but no encoded (non-dummy) record, enter:

```
PERFORM input-stream SOURCES ;
```

For example:

```
PERFORM 'BULK PRINT * ' SOURCES ;
```

prints all members with only a source record.

Selecting Dummy Members

To select all dummy members, enter:

```
PERFORM input-stream DUMMIES ;
```

The category DUMMIES includes all members that have dummy data entries records. Dummies can have unencoded source records.

For example:

```
PERFORM
    'WHAT INDIRECTLY USES "*"'
    DUMMIES ;
```

selects members that directly and indirectly use all dummy members.

You can select dummy members of a particular type, for example:

```
PERFORM
    'WHAT DIRECTLY USES "*"'
    DUMMY FILES ;
```

lists those members that use dummy FILEs.

The keyword DUMMY is accepted with the index-names ALIAS, CATALOG and ATTRIBUTE but has no effect on the selecting of these index-names.

Selecting a List of Non-Dummy Members

To select all non-dummy members, that is, members which have been defined and encoded, enter:

```
PERFORM input-stream REAL ;
```

For example:

```
PERFORM
    'WHAT DIRECTLY USES "*"'
    REAL ;
```

selects members that directly use all real members.

Selecting a List of Named Members

To select a list of one or more named members, enter:

```
PERFORM input-stream MEMBERS member-name-list ;
```

where *member-name-list* is one or more member names separated by commas and identifies the members to be processed. For example:

```
PERFORM
    'KEEP WHAT DIRECTLY USES *'
    MEMBERS REC-RPT-HEAD, REC-RPT-HEAD2 ;
```

keeps the members that directly use the members REC-RPT-HEAD and REC-RPT-HEAD2.

Member names are processed in the order they are specified in the member-name-list unless the keyword ALPHABETICALLY is used.

If you specify both the KEPT-DATA and MEMBERS keywords in your PERFORM command, then any members that fall within either category are listed. Members that appear in both categories are listed only once.

Selecting Index-Names in a KEPT-DATA List

To select index-names in an unnamed KEPT-DATA list, enter:

```
PERFORM input-stream KEPT-DATA ;
```

For example:

```
PERFORM
    'WHAT DIRECTLY USES *'
    KEPT-DATA;
```

lists those members that directly use the members kept in the unnamed KEPT-DATA list.

To select index-names in a named KEPT-DATA list, enter: `PERFORM input-stream KEPT-DATA IN list-name ;`. Where *list-name* identifies a particular named KEPT-DATA list.

For example:

```
PERFORM
    'WHAT FORMS *'
    KEPT-DATA IN CATALOG ;
```

selects, from a KEPT-DATA list named CATALOG, all members whose encoded record contains a catalog classification.

Index-names are processed in the order they appear in the KEPT-DATA list unless the keyword ALPHABETICALLY is used.

If you specify both KEPT-DATA and MEMBERS keywords in your PERFORM command then any members that fall within either category are listed. Members that appear in both categories are listed only once.

To clear the entries in a KEPT-DATA list before the PERFORM command is executed, enter:

```
PERFORM input-stream CLEAR-KEPT-DATA ;
```

For example:

```
KEEP WHICH MEMBERS DIRECTLY CONSTITUTE DATE ;
```

keeps a list of all members used by the member DATE, and:

```
PERFORM
    'ALSO KEEP WHAT DIRECTLY USES *'
    KEPT-DATA CLEAR-KEPT-DATA ;
```

clears from the unnamed KEPT-DATA list the members that directly constitute DATE and adds those members that directly use each of them.

The CLEAR-KEPT-DATA keyword is always actioned, even when you specify DISPLAY-ONLY.

Selecting a List of Index-Names ALPHABETICALLY

Index-names are processed in the order they appear in the KEPT-DATA list. Member names are processed in the order they are specified in the list of named members.

You can process KEPT-DATA lists and lists of named members in alphanumeric order by specifying the keyword ALPHABETICALLY.

For example:

```
PERFORM
    'REPORT * '
    MEMBERS EMP-IDENT, TAX-CODE, ACC-NO ALPHABETICALLY ;
```

reports the selected members in the order ACC-NO, EMP-IDENT, TAX-CODE.

Selecting or Excluding Index-Names and Member Types

To select all index-names, enter:

```
PERFORM input-stream INDEX-NAMES ;
```

To select particular types of index-names, enter:

```
PERFORM input-stream index-name-type ;
```

where *index-name-type* is the interrogate keyword for:

- Any member type or collective member type available in your repository
- The keywords ALIASES, CATALOGS and, with the User Defined Syntax facility, ATTRIBUTES.

When more than one index-name-type is specified each must be separated by a comma.

To select all index-names except those of particular types, enter:

```
PERFORM input-stream INDEX-NAMES EXCEPT index-name-type ;
```

For example:

PERFORM COMMAND-STREAM COMX INDEX-NAMES EXCEPT ALIASES, CATALOGS, ATTRIBUTES ;

applies the COMMAND-STREAM named COMX to all encoded members of any member type, but will not include the index-names ALIASES, CATALOGS or ATTRIBUTES.

Refer to "Applying Commands to a Selection Of Index-names Based on Name, Status, Time and User" on page 293 for details of selecting index-names by name, status, time and user.

Refer to "Collective Member Types" on page 316 for details of collective member-types.

Name, Status, Time and User-Related-Selections

Refer to "Applying Commands to a Selection Of Index-names Based on Name, Status, Time and User" on page 293 for details of the name, status, time and user-related-selections.

The PRINT, PRINT FIRST and DISPLAY-ONLY Keywords

Before the PERFORM command is executed, a list consisting of each command to be performed on each index-name specified in the selection is generated in memory.

For example:

```
PERFORM
    'WHAT DIRECTLY USES *'
    ,'PRINT *'
    MEMBERS TAX-CODE, ACC-NO ;
```

generates the following list of commands:

```
WHAT DIRECTLY USES TAX-CODE ;
PRINT TAX-CODE ;
WHAT DIRECTLY USES ACC-NO ;
PRINT ACC-NO ;
```

Once the list is complete, execution of the commands begins and the resulting output displays.

The PRINT, PRINT FIRST and DISPLAY-ONLY keywords allow you to display the list of generated commands.

To display the generated commands but not execute them, enter:

```
PERFORM input-stream DISPLAY-ONLY ;
```

This allows you to verify that the commands to be executed are generated as intended and, if not, to correct the PERFORM command before reentering it without the DISPLAY-ONLY keyword.

To display each command before the resulting output, enter:

```
PERFORM input-stream PRINT ;
```

This allows you to see each element of output in relation to the command that generated it.

To display all the generated commands before the first one is executed, enter:

```
PERFORM input-stream PRINT FIRST ;
```

This allows you to see a complete list of commands followed by all the output.

The NO-TERMINATOR, CLEAR-KEPT-DATA, PRINT (FIRST) or

DISPLAY-ONLY keywords can be specified in any order at the end of the command.

Selecting Members That Have Passed or Need Integrity Checks

To select check-ok members, members that have passed integrity checks, include the CHECK-OK keyword in the command.

To select check-needed members, members that need integrity checks, include the CHECK-NEEDED keyword in the command.

For further details of check-ok and check-needed members, refer to *ASG-MethodManager Administration*.

Selecting Members Based on a Specific Member Type

To select members based on a specific member type include the GENERIC keyword in the command. For example:

```
PERFORM 'input-line' GENERIC FILES ;
```

PERFORM Examples

To produce COBOL data descriptions for all the FILEs used by the PROGRAM PROG-RPT-MASTER, enter:

```
KEEP WHICH FILES CONSTITUTE PROG-RPT-MASTER ;
PERFORM
    'PRODUCE COBOL FROM "*"'
    KEPT-DATA ;
```

The Perform Character (here '*') is enclosed in quotes because the member names that replace it may contain characters from the extended character set.

To change all members with the type GROUP that are catalogued as RECORD, from GROUP to RECORD member type, enter:

```
KEEP WHICH GROUPS FORM RECORD ;
PERFORM
    'MODIFY *;'
    ,'CHANGE 100 "GROUP" TO "RECORD"'
    ,';'
    KEPT-DATA NO-TERMINATOR ;
```

The keyword NO-TERMINATOR and the presence of terminators is required because the PERFORM input-lines include the MODIFY command. This example illustrates the applications possible with the User Defined Syntax facility. Preparatory work would have included cataloging members under classifications corresponding to the member type to be established. It is assumed that the default line number (100) for the first line applies to all records.

To report the FILEs in the repository, giving the PROGRAMs that use each FILE and the FILE's COBOL data descriptions, enter:

```
PERFORM
    'REPORT *'
    ,'TEXT "WHERE FILE * IS USED"'
    ,'WHICH PROGRAMS USE *'
    ,'SKIP'
    ,'TEXT "COBOL DESCRIPTION OF FILE *"'
    ,'PRODUCE COBOL FROM * NOGENERATION PRINT'
    FILES ;
```

If printed, each report, each list of PROGRAMs and each COBOL data description begins on a new page.

To copy each GROUP used by any FILE owned by PERSONNEL, naming the new GROUP after the member it was copied from, but prefixed by PAY-, and make PERSONNEL the owner of the new members, enter:

```
KEEP WHICH FILES BELONG-TO PERSONNEL ;
PERFORM
    'ALSO KEEP WHICH GROUPS CONSTITUTE *'
    KEPT-DATA CLEAR-KEPT-DATA ;
PERFORM
    'COPY * TO PAY-*'
   ,'PROTECT PAY-*'
   ,'OWNER PERSONNEL' KEPT-DATA ;
```

The KEPT-DATA list for the first PERFORM command contains a list of programs and the KEPT-DATA list for the second PERFORM command contains a list of groups. CLEAR-KEPT-DATA is included so that the two lists do not overlap.

To process only check-ok members in the unnamed KEPT-DATA list, enter:

```
PERFORM input-stream CHECK-OK KEPT-DATA ;
```

where *`input-stream`* is a sequence of PERFORM input-lines.

PERFORM Syntax

```
>>──PERFORM ──┬──COMMAND-STREAM member-name ──────────────────────────>
              │  <<<<<< ,<<<<
              └──'input-line' ────────────────────────────────────────┘

>──┬──status-related-selection ──┬──────────────────────────────────>
   ├──LOCKED ────────────────────┤
   ├──CHECK-OK ──────────────────┤
   └──CHECK-NEEDED ──────────────┘

>──┬──ALPHABETICALLY ──┬──┬──name-related-selection ──┬──────────────>
   └──────────────────┘  └──────────────────────────┘

>──┬──time-and-user-related-selection ──┬──┬──NO-TERMINATOR ──┬──────>
   └────────────────────────────────────┘  └──TERMINATOR ─────┘

>──┬──CLEAR-KEPT-DATA ──┬──┬──DISPLAY-ONLY ──────────┬──┬──;──┬──>><
   └───────────────────┘  ├──PRINT ──┬──────────────┤  └──.──┘
                          │          └──FIRST ──────┘
```

where:

member-name is the name of an encoded COMMAND-STREAM member

input-line is any Manager Products command, and may contain an asterisk where a member-name, alias, catalog classification or indexed attribute would normally occur

status-related-selection is defined in "Applying Commands to a Selection Of Index-names Based on Name, Status, Time and User" on page 293

selection is:

```
>──┬──KEPT-DATA ──┬──────────────┬──┬──────────────────────────────┬──>
   │              └──IN list-name┘  │  <<<< , <<<<                 │
   │                                └──MEMBERS member-name ────────┘
   │
>──┬──────────────────────────────────────────────────────────────────>
   │  <<<<<< , <<<<<<
   ├──────────────────────────────────────────────────────────────────
   │  ┬──DUMMY ──┬
   │  ├──EXCEPT──┤
   │  └──REAL ───┘
   │
   ├──┬──DUMMYS──────────┬──┬──────────────────────────────────────┬
   │  ├──DUMMIES ────────┤  │  <<<<<< , <<<<<<                     │
   │  ├──INDEX-NAMES ────┤  └──EXCEPT index-name-type ─────────────┘
   │  └──REAL ───────────┘
   │
   └──SOURCES ──────────────────────────────────────────────────────
```

where:

list-name is the name of a KEPT-DATA list

member-name is the name of a repository member

index-name-type is:

```
➤──┬──┬──────────────┬──────────────────────────────────────────➤
   │  │  GENERIC     │  member-type
   │  └──────────────┘
   │
   ├──ALIASES ──────────────────────────────────────────────────
   ├──CATALOGS──────────────────────────────────────────────────
   ├──CATALOGUES────────────────────────────────────────────────
   └──ATTRIBUTES────────────────────────────────────────────────
```

where *member-type* is the interrogate keyword for any member type, or any collective member type, available in your repository.

name, *time* and *user-related-selection* are defined in "Applying Commands to a Selection Of Index-names Based on Name, Status, Time and User" on page 293.

PRINT (CMR-CM01)

The PRINT command outputs the source records, or selected parts of the source records, of selected dictionary members.

Refer to "PRINT Syntax" on page 213 for the syntax of the PRINT command.

How to Print a Member

To output the current source record of a member, enter:

```
PRINT member-name-list ;
```

where *member-name-list* is a list of one or more member names separated by commas.

The command is accepted or rejected individually for each member name you specify.

For example, to print three members named, ACC-OCT, ACC-NOV and ACC-DEC, enter:

```
PRINT ACC-OCT, ACC-NOV, ACC-DEC ;
```

If you attempt to print a member which does not have a source record in the dictionary, the message:

```
DM01240E member-name NO SOURCE PRESENT
```

is output, the member is not printed and processing continues with the next member name or command.

You can only print a member if your security level is sufficient for you to access that member.

Unless you specify otherwise, when you use the PRINT command, only the source records of members in the current status or, if the member does not have a source record in the current status, from the nearest base status which does contain a source record of that member, are printed.

If you want to output specific information, you can include one or more of the following in your PRINT command:

- ALL-STATUSES: to apply the command to all the statuses currently available to you
- LINES: to print specific lines of a member's source record
- ENCODED-SOURCE: to print a member's source record as it stood at the time the member was last successfully encoded.

You can print a selection of members in bulk by using the BULK PRINT command.

Note:

- Details relating to statuses apply only if either the Basic or the Advanced Status facility (selectable units CMR-DD2 or CMR-AD21) is installed
- Details relating to members' protection apply only if the Audit and Security facility (selectable unit CMR-DD3) is installed.

Refer to "Security Levels" on page 18 for details of security levels.

Refer to *ASG-Manager Products Status Concepts* for details of the Status facilities.

Printing Specific Lines of a Member

If you specify LINES, in your PRINT command, you can specify only one member name in the command.

To output the first *n* lines of a member's source record, enter:

```
PRINT member-name LINES n ;
```

where *n* is an integer of no more than seven digits, representing the number of the lines of the member's source record that you want to print, starting from the first line.

For example, to output the first 20 lines of the encoded source record of a member named PROG-RPT-MASTER, enter:

```
PRINT PROG-RPT-MASTER LINES 20 ENCODED-SOURCE ;
```

To output lines with numbers in the range *n1* to *n2*, inclusive, enter:

```
PRINT member-name LINES n1 TO n2 ;
```

where:

> *n1* is an integer of no more than seven digits representing the number of the first line of the source record that you want to print.
>
> *n2* is an integer of no more than seven digits representing the number of the last line of the source record that you want to print.

For example, if a member named PROD-LIST contains 50 lines of data and its lines are numbered from 100 in increments of 100 (that is 100 to 5000), to output the source record of PROD-LIST excluding the first 10 lines and only up to the line numbered 3600, enter:

```
PRINT PROD-LIST LINES 1100 TO 3600 ;
```

To output a member's source record from line number n to the end, enter:

```
PRINT member-name LINES n TO END ;
```

For example, to output the source record of a member named EMP-ADDR from line number 1200 onwards, enter:

```
PRINT EMP-ADDR LINES 1200 TO END ;
```

Printing Members in a Status Environment

Unless you specify otherwise, when you print a member, the member is printed from the current status or, if the member does not have a source record in the current status, from the nearest base status that contains a source record of the member.

If you include the ALL-STATUSES keyword in your command, the member's source record is printed from every status that contains a source record of the member and that you have access to.

For example, if you have a four-status dictionary which contains:

- A member named ACC-NO with a source record in two statuses
- A member named ACC-NAME with a source record in each of the four statuses

and you want to output a copy of each of these two members from every status in which they appear, enter:

```
PRINT ACC-NO, ACC-NAME ALL-STATUSES ;
```

In this example two separate versions of ACC-NO and four separate versions of ACC-NAME would be output (assuming you have access to all four statuses in your dictionary).

Refer to *ASG-Manager Products Status Concepts* for details of the Status facilities.

Printing a Member's Source Record as it was at the Time it was Last Successfully Encoded

To output a member's source record as it was when the member was last successfully encoded, include the ENCODED-SOURCE keyword in your print command.

To output the current source record of a member and its source record as it was when the member was last successfully encoded, include the keywords

ENCODED-SOURCE ALSO in your print command.

For example, if the source records of two members named EMP-LIST and

EMP-CODE have been updated since they were last encoded, you can compare the two different versions of each of the members' source records by entering:

```
PRINT EMP-LIST, EMP-CODE ENCODED-SOURCE ALSO;
```

Print Output

By default, print output is preceded by a statement that identifies the member being printed, whether the member is encoded or unverified, and from which status the member's source record has been printed.

An encoded member is preceded by:

```
PRINT OF member-name ENCODED IN STATUS status-name
```

A member with a source record but no encoded record is preceded by:

```
PRINT OF member-name UNVERIFIED IN STATUS status-name
```

For an unverified member, the current source record is preceded by:

```
PRINT OF member-name UNVERIFIED IN STATUS status-name
```

and the source record as it was when the member was last successfully encoded is preceded by:

```
PRINT OF member-name ENCODED IN STATUS status-name
```

Each source record of the member being printed is followed by:

```
END OF PRINT
```

You can suppress these print delimiters by using the SET PRINT-DELIMITERS OFF command.

By default, print output contains leading spaces and source line numbers. However, these can be removed using variants of the SET PRINT-SHIFT command.

If you try to print a member that has no source record, then the message:

```
DM01250E member-name NOT PRESENT ON DICTIONARY
```

is output, and processing continues with the next member name or the next command.

If you try to print the encoded record of a member with a source record but no encoded record, then the message:

```
DM01601E member-name NOT ENCODED
```

is output, and processing continues with the next member name or the next command.

PRINT Syntax

```
                <<<< , <<<<
>>──PRINT member-name ──┬──────────┬──┬──;──┬──────────────────────>><
                        └─options──┘  └──.──┘
```

where:

member-name is the name of a dictionary member

options is:

```
>──┬──────────────────────────────────────────────────────────────────┬──>
  │                                                                   │
  └─LINES ──┬─integer──────────────────────────────────────────────┬─┘
            │                                                      │
            └─line-number TO ──┬─line-number 2──┬──────────────────┘
                               │                │
                               └─END────────────┘

>──┬──────────────────────┬──┬──────────────────────────────────────┬──>
  │                      │  │                                      │
  └─ALL-STATUSES─────────┘  └─ENDODE-SOURCE──┬──────────────┬───────┘
                                              │              │
                                              └─ALSO─────────┘
```

where:

integer is a number of lines in the source record

line-number and *line-number-2* are source line numbers.

PROTECT (CMR-DD3)

The PROTECT command protects members of the dictionary against unauthorized access, alteration or removal.

Refer to "PROTECT Syntax" on page 216 for the syntax of the PROTECT command.

Protecting Individual Members of the Dictionary

To protect individual members of the dictionary, enter:

```
PROTECT member-name-list protection ;
```

where:

member-name-list is one or more member names each separated by a comma.

protection is one or more of these clauses, in any order:

```
ACCESS access-security-level
ALTER alter-security-level
REMOVE remove-security-level
OWNER owner-name
```

where:

access-security-level, *alter-security-level* and remove-security-level are integers in the range 0 to 254.

owner-name is the name of an owner previously established by the Controller.

For example:

```
PROTECT EMP-NO ACCESS 25 ALTER 50 REMOVE 100 OWNER ACCOUNTS ;
```

will give EMP-NO an access-security-level of 25, an alter-security-level of 50 and a remove-security-level of 100, and assign it to the owner ACCOUNTS.

You can only access, alter or remove a member whose relevant security level is less than or equal to your general security level. With the example above, if your general security level were 50, you could access and alter the member EMP-NO, but not remove it.

The *access-security-level* prevents unauthorized:

- Extraction of any information about a member, apart from its existence
- Alteration or removal of the member.

The *alter-security-level* prevents unauthorized:

- Alteration of a member's source or encoded records
- Locking of the member
- Protection of the member
- Removal of the member.

The *remove-security-level* prevents unauthorized removal of a member.

If the member has no alter-security-level, or if the access-security-level is above the alter-security-level, then the alter-security-level is made equal to the access-security-level.

If the member has no remove-security-level, or if the alter-security-level is above the remove-security-level, then the remove-security-level is made equal to the alter-security-level.

Protection of a member also affects any members which directly use it. For example, you cannot encode a member which directly uses another member to which you are denied access.

The PROTECT command is accepted only if your general security level is greater than or equal to the protection security level set for the dictionary by the Controller.

If the Basic or Advanced Status facilities are installed (selectable units CMR-DD2 and CMR-AD21 respectively) and a dictionary has several statuses, the PROTECT command will protect every copy of the named member in all the statuses, including read-only statuses.

Assigning Ownership to Members

To assign ownership to one or more members, enter:

```
PROTECT member-name-list OWNER owner-name ;
```

where:

member-name-list is one or more member names each separated by a comma.

owner-name is a string of up to 32 printable characters, being the name of an owner as previously established by the Controller.

For example:

```
PROTECT OFF-TEL-NO, OFF-NO OWNER PERSONNEL ;
```

assigns the members OFF-TEL-NO and OFF-NO to the owner PERSONNEL.

Owners can be individuals, departments, project teams or any other body.

The Controller can assign a different security level for each owner registered in the dictionary. Each such security level is known as a specific security level. For you to access, alter or remove an owned member, your specific security level for that owner must be greater than or equal to the access, alter or remove security level of the member.

If you are not assigned a specific security level for an owner, you are not allowed to access or process that owner's members at all. For example, by declaring members developed by a project to be owned by that project, you can prevent users from outside the project from accessing or processing the members.

Once an owner has been declared for a member, the ownership continues until either a new owner is declared for the member, or ownership is removed by giving the following command:

```
PROTECT member-name OWNER NONE ;
```

Note:

To specify an owner-name NONE, enclose it in delimiters.

If the Basic or Advanced Status facilities are installed (selectable units CMR-DD2 and CMR-AD21 respectively) and a dictionary has several statuses, ownership applies to every copy of an owned member in all the statuses, including read-only statuses.

PROTECT Syntax

```
                <<<< , <<<<
>>---PROTECT member-name ---+---ACCESS access-level ---+---;---+---><
                            |---ALTER alter-level   ---|   |   |
                            |---REMOVE remove-level ---|   |---.---|
                            |---OWNER ---+---owner-name---|
                                         |---NONE -------|
```

where:

member-name is the name of a dictionary member

access-level, *alter-level* and *remove-level* are integers in the range 0 to 254, specifying the security levels

owner-name is a string of up to 32 printable characters, being the name of an owner as previously established by the Controller.

QUERY KEPT-DATA (CMR-CM01)

The QUERY KEPT-DATA command finds out what KEPT-DATA lists, if any, you have created.

To list all KEPT-DATA lists, enter:

```
QUERY KEPT-DATA ALL ;
```

To list an unnamed KEPT-DATA list, enter:

```
QUERY KEPT-DATA ;
```

or

```
QUERY KEPT-DATA UNNAMED ;
```

To list any named KEPT-DATA lists, enter:

QUERY KEPT-DATA LIST *list-name* ;

where *name-list* is the name of one or more KEPT-DATA lists, separated by commas.

To list named and unnamed KEPT-DATA lists, enter:

```
QUERY KEPT-DATA UNNAMED AND LISTS list-name ;
```

To list the KEPT-DATA lists which contain a specific member, you can include the keyword CONTAINING as part of any of the above commands.

For example:

QUERY KEPT-DATA ALL CONTAINING *member-name* ;

or

QUERY KEPT-DATA UNNAMED AND LIST list-name CONTAINING *member-name* ;

where:

member-name is the name of a specific member.

list-name is the name of a KEPT-DATA list.

Note:

You can have a maximum of 32,768 named KEPT-DATA lists.

QUERY KEPT-DATA Syntax

```
>>──QUERY KEPT-DATA──┬──────────────────────────────────────────┬──>
                     │──UNNAMED──┬──────────────────────────────┤
                     │           └──AND LISTS list-name─────────┤
                     │       <<<<,<<<<                          │
                     └──LISTS list-name─────────────────────────┘
```

where:

list-name is the name of a KEPT-DATA list.

member-name is the name of a specific member.

QUERY MODE

Refer to "MODE, QUERY MODE, DMS, WBDA (CMR-CM01)" on page 177 for details of QUERY MODE.

READ and WRITE (CMR-FE01, CMR-TP7, CMR-TP8)

The READ and WRITE commands:

- Read the contents of an MVS dataset or a CMS file into an Edit or Update buffer
- Write the contents of any buffer to an MVS dataset or CMS file.

How to Read From, and Write To, an MVS Dataset

Under MVS the READ and WRITE commands can be used to read from and write to either a sequential dataset or a member of a partitioned dataset (PDS). In addition, for a PDS the directory can also be read.

Fixed (FB) and variable (VB) record formats are supported for both input and output. All valid blocksizes are supported.

In order to read from, or write to, a dataset, it must be allocated prior to executing the READ or WRITE command.

When WRITE is issued against a non-formatted screen, the entire content from the current line to the end of the buffer is written to the dataset. However, when WRITE is issued against a MethodManager formatted screen, only the lines which are currently displayed are written; those sections accessible through scrolling are not written.

The READ command supports datasets with any logical record length, but since the maximum width of a CONTROLMANAGER buffer is 255 bytes, any records exceeding that will be truncated.

The features described here are available if the MVS/TSO Interface is installed (selectable unit CMR-TP7).

To read the contents of a sequential dataset into an Edit or Update buffer, enter:

```
READ ddname ;
```

where *ddname* is the ddname specified in a previously issued TSO ALLOCATE command.

The contents of the dataset are inserted after the current line of the Edit or Update buffer.

To write the contents of any buffer (Command, Update, Edit, Lookaside or InfoBank Panel) to a dataset, enter:

```
WRITE ddname ;
```

where *ddname* is the ddname specified in a previously issued TSO ALLOCATE command.

If the ddname is omitted, a default ddname, MPRTSO, is used.

When using the WRITE command, any existing data in the dataset is overwritten. The output dataset is closed on completion of the command. New or existing datasets may be written to, but must have been defined previously.

For output, if the dataset attributes are missing, the following default values are assumed:

Record format FB
LRECL 255
BLKSIZE25500

To read a member or the directory of a PDS into an Edit or Update buffer, enter:

```
READ ddname name ;
```

where:

> *ddname* is the *ddname* specified in a previously issued TSO ALLOCATE command and is mandatory, even when the default ddname of MPRTSO is used.
>
> *name* is mandatory and is either the name of a PDS member or the keyword DIRECTORY if the directory is to be read. The keyword DIRECTORY may not be truncated.

The contents of the member or directory are inserted after the current line of the Edit or Update buffer.

To write the contents of any buffer (Command, Update, Edit, Lookaside or InfoBank panel) to a member of a PDS, enter:

```
WRITE ddname member-name REPLACE ;
```

where:

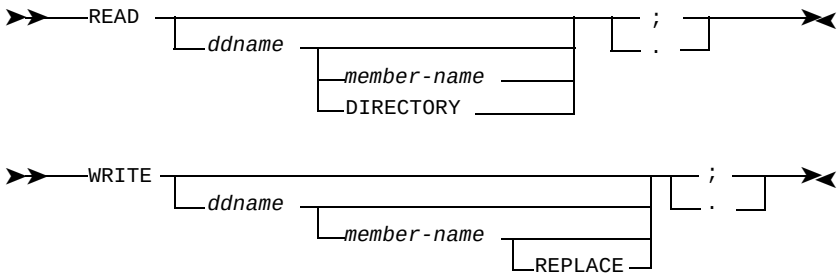
ddname is the ddname specified in a previously issued TSO ALLOCATE command and is mandatory, even when the default ddname of MPRTSO is used.

member-name is mandatory and is the name of the member to be written.

REPLACE is optional. It indicates that the member being written is to replace any existing member having the same name.

For output, the dataset attributes must be available from the dataset DSCB; default values are not assumed.

READ and WRITE Syntax



where:

ddname is the ddname specified in a previously issued TSO ALLOCATE command.

member-name is the name of a PDS member to be read or written.

How to Read From, and Write To, CMS files

To read the contents of a CMS file into an Edit or Update buffer, enter:

```
READ fn ft fm ;
```

where:

fn is the name of the CMS file (the default is CMR).

ft and *fm* are the filetype and filemode of the CMS file (the defaults are DATA for the filetype, A1 for the filemode).

The file is inserted after the current line of the current buffer.

To write the contents of any buffer (Command, Update, Edit, Lookaside or InfoBank Panel) to a CMS file, enter:

```
WRITE fn ft fm ;
```

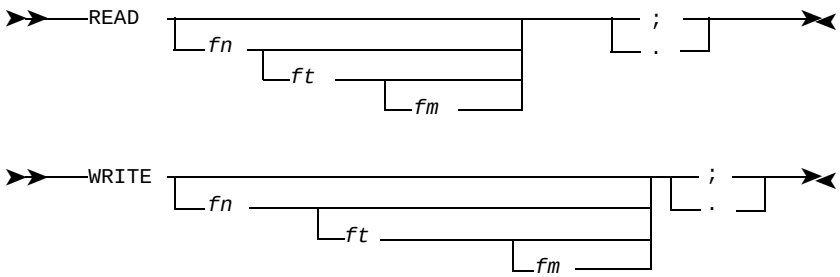
fn, *ft* and *fm* are as defined above.

When WRITE is issued against a non-formatted screen, the entire content from the current line to the end of the buffer is written to the CMS file. However, when WRITE is issued against a MethodManager formatted screen, only the lines which are currently displayed are written; those sections accessible through scrolling are not written.

The CMS file thus created has a variable record format.

The features described here are available if the VM/CMS Interface is installed (selectable unit CMR-TP8).

READ and WRITE Syntax



where *fn*, *ft* and *fm* are the filename, filetype and filemode of a CMS file

REINSTATE (CMR-CM01)

The REINSTATE command removes an unverified source record, thus reinstating a member to the condition in which it was last successfully encoded.

To reinstate a member or members, enter:

```
REINSTATE member-name-list ;
```

where *member-name-list* is a list of one or more member names separated by commas.

The command is accepted or rejected independently for each member you specify. If a member does not have an encoded record, the REINSTATE command has the same effect as a REMOVE command; it removes the source record.

A member is unverified:

- If it has a source record but no encoded record
- If its source record differs from its encoded record; that is, the source record has been altered since the member was last successfully encoded.

An unverified member is created when:

- You ALTER, INSERT or SFILE a member, or
- You COPY one member TO another, or
- A member fails to encode (because of syntax errors).

For example, consider the encoded members EMP-CODE and EMP-FILE. If you ALTER these members, they are then unverified; their source records have been changed since their encoded records were generated. To reinstate the members, enter:

```
REINSTATE EMP-CODE, EMP-FILE ;
```

The ALTERed source records are then replaced with the source records from which the most recent encoded records were generated.

If the Workstation Interface facility (selectable unit CMR-WS01) is installed, you cannot reinstate a member that is locked by another user. If a member within the member-name-list is locked, the REINSTATE command is rejected and processing continues with the next member name or command.

Refer to "LOCK (CMR-WS01)" on page 173 for details of member locking.

REINSTATE Syntax

```
                <<<< , <<<<
>>──REINSTATE member-name ──────────┬─;─┬──────────><
                                    └─.─┘
```

where *member-name* is the name of a dictionary member.

REMOVE (CMR-CM01)

The REMOVE command removes members from the repository.

To remove one or more members from the repository, enter:

```
REMOVE member-name-list ;
```

where *member-name-list* is a list of one or more member names separated by commas.

The command is accepted or rejected independently for each member name you specify.

You cannot remove a member if:

- It is directly used by an encoded member
- A member of the same name also exists in a base status
- It is locked by another user in the current status
- Your general security level is lower than the repository's insertion security level
- Your (general or specific) security level is lower than the member's remove security level.

Refer to the CHECK prefix command for details of how integrity checking affects removal of members.

If you specify a member name for which there is no record in the repository, this message is output:

```
DM01250E member-name NOT PRESENT ON DICTIONARY
```

Processing continues with the next member name or the next command.

The REMOVE command is treated as an updating command by the automatic recovery system, which operates individually in respect of each member name specified in the command.

REMOVE Syntax

```
                <<<< , <<<<
>>---REMOVE member-name ---+---;---+---------------------------><
                           +---.---+
```

where *member-name* is the name of a repository member.

RENAME (DMR-DD1)

The RENAME command renames an existing member of the dictionary.

Refer "RENAME (DMR-DD1)" on page 224 for the syntax of the RENAME command.

Renaming an Existing Member

To rename an existing member of the dictionary, enter:

RENAME *member-name-1* AS *member-name-2* ;

where:

member-name-1 is the name of the member to be renamed.

member-name-2 is the new name, which must obey the rules governing member names and must not be the name of a member that already exists in the dictionary.

You cannot rename a member if:

- It is directly used by an encoded member
- It exists in another status
- It is locked by another user in any status
- Your general security level is lower than the dictionary's insertion security level
- Your alter security level is lower than the member's alter security level.

If an encoded member directly refers to the member you want to rename, the message:

DM01633E member-name IS USED BY OTHER MEMBERS - CANNOT BE RENAMED

is output, and the member is not renamed. Processing continues with the next command.

If you try to rename a member with the name of an existing member, the message:

```
DM01630E member-name-2 ALREADY PRESENT ON THE DICTIONARY
```

is output, and the member is not renamed. Processing continues with the next command.

If you rename a member held in a KEPT-DATA list, the name in the KEPT-DATA list is automatically changed to the new name.

The RENAME command is treated as an updating command by the automatic recovery system.

Note:

- Details relating to locking only apply if the Workstation Interface facility (selectable unit CMR-WS01) is installed
- Details relating to security levels only apply if the Audit and Security facility (selectable unit CMR-DD3) is installed
- Details relating to statuses only apply if either the Basic or Advanced Status facilities (selectable units CMR-DD2 and CMR-AD21 respectively) are installed.

Refer to *ASG-ControlManager User's Guide* for details of the rules governing member names.

Refer to *ASG-Manager Products Status Concepts* for details of renaming members in a multi-status environment.

Refer to "The Security System" on page 17 for details of the security system.

How to Rename a Member Directly Used By Another Member

You cannot use the RENAME command to change the name of a member that is directly used by another encoded member.

The following instructions explain how you can rename a member that is directly used by one or more encoded members.

Note:

`Member-name-1` is the name of the member to be renamed and *`member-name-2`* is the new member name.

To rename the member:

- Interrogate the dictionary to find out WHAT DIRECTLY USES member-name-1
- Use UPDATE and FILE, or MODIFY (or any other Manager Products command or combination of commands that enables you to update and encode a member) to update each member that directly uses member-name-1
- Change any references to *member-name-1* to refer to *member-name-2*
- COPY *member-name-1* TO *member-name-2*
- ENCODE *member-name-2*
- REMOVE *member-name-1*.

Refer to *ASG-ControlManager User's Guide* for details of the UPDATE command.

RENAME Syntax

```
>>──RENAME member-name-1 AS member-name-2 ┬─ ; ─┬──────────><
                                          └─ . ─┘
```

where:

member-name-1 is the name of the existing member to be renamed

member-name-2 is the name by which the member is to be known.

REPLACE (CMR-CM01)

The REPLACE command:

- Replaces the source record of a member with the new source record of that member
- Generates an encoded record from the new source record
- Replaces the Data Entries record of that member with the newly encoded record.

Refer to "REPLACE Syntax" on page 230 for the syntax of the REPLACE command.

How to Replace the Source and Encoded Record of a Member

To replace a member, enter:

```
REPLACE member-name ;
 member-definition
;
```

where:

member-name is the name of a member with a source record or dummy record in the dictionary.

member-definition is the member's new source record.

For the syntax of member definition, enter:

```
HELP member-type-name-S ;
```

where *member-type-name* is the name of the type of member you want to enter.

For example:

```
HELP GROUP-S ;
```

Use the SHOW UDS command to find out which member types are available in your dictionary.

For example, to replace the member EMP-IDENT, enter:

```
REPLACE EMP-IDENT ;
GROUP
CONTAINS S-S-CODE, TEL-NO
DESCRIPTION "THIS GROUP CONTAINS INFORMATION ABOUT"
            "EMPLOYEE'S SOCIAL SECURITY AND TELEPHONE"
            "NUMBER"
;
```

The new source record is inserted in the Source dataset. An encoded record is generated from the source record and inserted in the Data Entries dataset. The Index dataset is updated accordingly.

If the source is syntactically correct and a source record already exists for member-name, the same process is followed except that the new source record replaces the existing source record.

If the source is syntactically correct and an encoded record already exists for member-name, the encoded record generated from the new source record replaces the existing record in the Data Entries dataset. The used-by pointers in the existing encoded record are copied to the newly encoded record before the existing record is deleted.

If there are any errors in the syntax of the source record of the member you are replacing, the source record of the member is inserted or replaced (as appropriate); diagnostic messages are output when the member is encoded; and no new encoded record is generated.

If the member you are replacing uses any other members via a CONTAINS clause, the replaced member cannot be encoded if you have changed the member's member type to one that is lower in the member type hierarchy than the member types of the members which it uses.

Unless you include a NUMBER clause in your REPLACE command, lines in the new source record are numbered in increments of 100, starting with 100.

You can only REPLACE a member if your general security level is equal to or higher than the dictionary's insertion security level, and is sufficient for you to alter the member to be replaced and to access any members directly used by that member.

You cannot replace a member that is locked by another user. If a member is locked in the current status, the REPLACE command is rejected and processing continues with the next command.

The REPLACE command is treated as an updating command by the automatic recovery system.

Note:

- Details relating to locking are applicable only if the Workstation Interface facility (selectable unit CMR-WS01) is installed.
- Details relating to security levels are applicable only if the Audit and Security facility (selectable unit CMR-DD3) is installed.
- Details relating to statuses are applicable only if either the Basic or Advanced Status facility (selectable units CMR-DD2 and CMR-AD21 respectively) is installed.

Refer to "LOCK (CMR-WS01)" on page 173 for details of member locking.

Refer to "Security Levels" on page 18 for details of security levels.

Refer to *ASG-Manager Products Status Concepts* for details of statuses.

How to Number Lines in a Member's Source Record

To replace an existing source record with a new source record containing lines numbered in specified increments, and then encode the new source, enter:

```
REPLACE member-name NUMBER n ;
member-definition
;
```

where *n* is an unsigned integer in the range 1 to 10000 (inclusive). The lines in the source record are numbered in increments of *n*, the first line being number *n*.

For example, to replace the source record of EMP-IDENT with a new source record containing lines numbered in increments of 200, enter:

```
REPLACE EMP-IDENT NUMBER 200 ;
GROUP
CONTAINS S-S-CODE, TEL-NO
;
```

If you specify a signed integer in the range 1 to 10000 (inclusive) this message displays:

```
DM00127W n INVALID INTEGER
```

the sign is then automatically removed, and *n* is treated as a valid integer.

If you specify an integer less than 1 or greater than 10000 this message displays:

```
DM01182W n NOT VALID INCREMENT - '100' ASSUMED
```

and *n* is assumed to be 100.

If you do not specify an integer this message displays:

```
DM00127W ; INVALID INTEGER
```

and *n* is assumed to be 100.

If you specify an integer that causes the line numbers to extend into more than five digits, then, with one exception, only the last five digits are output in any printout or display of the source record. The exception is the response to a DISPLAY amendment line in an ALTER or MODIFY command; then all eight digits display.

Refer to "ALTER (CMR-CM01)" on page 52 for details of DISPLAY.

REPLACE Syntax

```
►──REPLACE member-name ──┬──────────────────┬──┬──;──┬──►
                         └─NUMBER integer───┘  └──.──┘
►──────member-definition──────────────────────────────►
►──┬──;──┬──────────────────────────────────────────►◄
   └──.──┘
```

where:

member-name is the name of a dictionary member

integer is an unsigned integer in the range of 1 to 10000, being the line number for the first line and the increment for subsequent line numbers of the member's source record

member-definition is the member's source record.

REPORT

The REPORT command outputs details of any encoded or dummy member, optionally with the members they use.

Refer to "REPORT Syntax" on page 236 for the syntax of the REPORT command.

Reporting One or a List of Members

You can target your REPORT command at the current repository or the Workbench Design Area (WBDA), by using the MODE command with the DMS or WBDA keywords. The following is concerned only with reporting repository members.

The REPORT command is used to output various details of an encoded or dummy member from its data entries record, including who created it, when it was created, and which members use or are used by it.

To report one or more encoded or dummy members, enter:

REPORT *member-name-list* ;

where *member-name-list* is a list of one or more member names separated by commas.

You can specify how much information you require in the report of a member by including the following keywords in your command:

- ALL-STATUSES to produce a report of the member from every status currently available to you; and/or
- HIERARCHY to report a member and also the members that it uses; or
- DOWN-TO to report a member and list, down to a specific level in the member type hierarchy, the name and member type of each of the members that the reported member uses.

Having decided how much information you want to report, you can define the way the output is displayed by including one of the following keywords in your command:

- SKIP to insert a page break between each report
- NO-SKIP to insert a blank line between reports.

An alternative layout for the output from a REPORT command can be specified in the FORMAT command.

If you include HIERARCHY, ALL-STATUSES, and SKIP or NO-SKIP in your REPORT command, you can enter them in any order but, in relation to the rest of the command, they should be entered after the member name to be reported.

If you specify more than one member name, the ALL-STATUSES and HIERARCHY keywords and the DOWN-TO clause only apply to the member name they immediately follow.

The security facility controls access to information held in the repository. For details of how security affects interrogations refer to "The Security System" on page 17.

Reporting a Member in a Status Environment

Unless you specify otherwise, when you use the REPORT command, members are reported from the current status or, if they do not have a data entries record in the current status, from the nearest base status that does contain a data entries record of any of the members being reported.

To report encoded and dummy members in all the statuses currently available to you, include the ALL-STATUSES keyword in your command. A separate report is created for each status.

In the output of a report command, the STATUS line in the report relates to the name of the status from which the report is drawn, not the current status.

If a member exists as a dummy member in the status in which you are working, details are output only if the member has different used-by pointers in that status when compared with the nearest of the base statuses in which it exists (if any).

If a member does not have a data entries record in any status, one of these messages display:

```
DM01601E member-name NOT ENCODED
```

or

```
DM01250E member-name NO SOURCE PRESENT
```

and processing continues with the next member name or the next command.

Reporting a Member and the Members That it Uses Via an EA Relationship Type CONTAINS

To report a member and the members it directly and indirectly uses via an EA relationship type CONTAINS, include the HIERARCHY keyword.

The CONTAINS clause must be specified in a member of one of these types:

- ITEM
- GROUP
- FILE
- TOTAL FILE
- ADABAS FILE
- TOTAL DATABASE.

Reporting a Member and Listing the Members That it Uses Via Any Relationship Type

To report a member and list the names and member types of the members it directly and indirectly uses via any relationship type, down to a specific level in the member type hierarchy, include the DOWN-TO keyword.

The member type stated in the DOWN-TO clause must be at an equal or lower level in the hierarchy than the member type of the member you are reporting.

To list the member-type keywords used in the DOWN-TO clause, enter:

```
SHOW UDS MEMBER-TYPES REPORT-DOWN-TO KEYWORDS ;
```

A reference path is not traced beyond a member used by a member at a lower level in the member type hierarchy.

Inserting a Blank Line or Page Break Between Reports

The first report always commences on a new page. By default, when several members are to be reported, each report is followed by a page break.

To insert a blank line, rather than a page break, after a report of a member, include the NO-SKIP keyword in your command.

To commence a report on a new page and also insert a page break after the report, in the same manner as the default, include the SKIP keyword in your command.

Note:

If you specify more than one member name, the NO-SKIP or SKIP keywords only apply to the member name they immediately follow.

Output for an Encoded Member

The following details of an encoded member are reported:

- Its member type
- Its edition number (the number of times it has been encoded)
- The name of the user who last encoded it
- The date and time it was last encoded
- The name of the status from which the report is drawn
- With the Audit and Security facility, the type of protection the member has and the name of its owner
- Information from its clauses
- The number of direct references to it
- The number of direct references from it
- Details appropriate to its member type
- The member type and member name of each member directly used by it via a CONTAINS clause.

For example:

```
REPORT OF GROUP EMP-IDENT
EDITION 2 ENCODED BY MASTER AT 13.09.03 ON 06 MAR 1988
STATUS DEV1
PROTECTION: OWNER,REMOVE,ALTER
OWNED BY PERSONNEL
ALIAS
    EMPLOYEE-IDENTIFICATION-RECORD
CATALOGUED AS
EMPLOYEE
SEE DEPARTMENT
KEYS EMP-NO
USER-EXIT PROJ
DESCRIPTION
    THIS GROUP CONTAINS INFORMATION ABOUT EMPLOYEE'S NAME
THIS MEMBER IS DIRECTLY REFERRED TO ONCE
THIS MEMBER CONTAINS 4 DIRECT REFERENCES
GROUP EMP-IDENT
FORM  HELD-AS
CONTAINS
    ITEM EMP-NAME
    ITEM EMP-ADDR
    GROUP TELEPHONE
```

Output for a Dummy Member

These details of a dummy member are reported:

- The name of the status from which the report is drawn
- Its member type
- The line: EDITION **DUMMY**
- The name of the user who encoded the member that created it
- The time and date it was created
- The number of direct references to it
- The number of direct references from it.

For example:

```
REPORT OF MODULE DMC28
  EDITION **DUMMY** CREATED BY MASTER AT 12.15.36 ON 09 JUL 1992
  STATUS PROG
  PROTECTION: NONE
  THIS MEMBER IS DIRECTLY REFERRED TO ONCE
  THIS MEMBER CONTAINS 1 DIRECT REFERENCE
```

REPORT Examples

To report the member EMP-CODE, enter:

```
REPORT EMP-CODE ;
```

To report:

- The member PRODUCTION from every status to which you have access
- The members EMP-CODE and EMP-NAME from the current status only

enter:

```
REPORT PRODUCTION ALL-STATUSES, EMP-CODE, EMP-NAME ;
```

To report the member CLIENT-ADDR and also any members that CLIENT-ADDR uses via an EA relationship type CONTAINS, enter:

```
REPORT CLIENT-ADDR HIERARCHY ;
```

To report the member EMP-NAME, then insert a blank line before reporting the member EMP-CODE, then insert a page break before reporting the member EMP-ADDR, enter:

```
REPORT EMP-NAME NOSKIP, EMP-CODE SKIP, EMP-ADDR ;
```

To report the member PROD-DATA and list the names and types of those members which PROD-DATA uses via any relationship type, down to those of the member type GROUP in the member type hierarchy, enter:

```
REPORT PROD-DATA DOWN-TO GROUPS ;
```

REPORT Syntax

```
        <<<<<<<<<<<<<<<<<<<<<<< , <<<<<<<<<<<<<<<<
➤──────REPORT member-name ──┬──────────────────┬──┬──┬──;──┬──➤◄
                            └──report-clause───┘  └──.──┘
```

where:

member-name is the name of an encoded member.

report-clause is:

```
➤──┬──────────────┬──┬──────────────────────┬──────────────────────────────➤
   └──ALL-STATUSES──┘  ├──HIERARCHY ──────────┤
                       └──DOWN-TO member-type ──┘
➤──┬──────────┬──────────────────────────────────────────────────────────➤
   ├──SKIP ───┤
   ├──NO-SKIP ──┤
   └──NOSKIP ───┘
```

where *member-type* is the interrogate keyword for any member type or collective member type available in your repository.

Use the SHOW UDS command to find out which member types are available in your repository.

RFILE (CMR-FE01)

The RFILE command replaces the source record of an existing member with the contents of the Update Buffer, and attempts to encode it.

RFILE is useful in circumstances where the contents of an Update Buffer cannot be successfully FILEd because the member name being used is the name of a member that already exists in the dictionary. To file a member in this circumstance, enter:

```
RFILE member-name ;
```

The RFILE command provides the same facility, in Full Screen Interactive environments where the Extended Interactive facility is installed, as is provided by the REPLACE command in other interactive and batch environments.

You can tailor the environment so that the source of the member is displayed in a Lookaside Buffer when you successfully enter the RFILE command by using the SET UPDATE-OUTPUT command.

If the Workstation Interface facility is installed (selectable unit CMR-WS01), you cannot RFILE a dictionary member that is locked by another user. If member-name has been locked since you updated it, the RFILE command is rejected.

Refer to *ASG-ControlManager User's Guide* for details of member naming rules and the SET and QUERY UPDATE-OUTPUT commands.

Refer to "LOCK (CMR-WS01)" on page 173 for details of member locking.

RFILE Syntax

```
>>──RFILE ┬──────────────────┬──┬──;──┬──────────────>>
          └─member-name──────┘  └──.──┘
```

where `member-name` is the name of a dictionary member.

SHOW ALIAS-TYPES (CMR-CM01)

The SHOW ALIAS-TYPES command displays information about general aliases and specific alias-type keywords.

Alias-types and alias rules may be defined by the Controller using the CONTROL NEW-ALIASES command. This is described in *ASG-Manager Products Controller's Manual*.

The SHOW ALIAS-TYPES command is used to find out:

- The specific alias-type keywords that are available for use in the dictionary
- The rules which apply to aliases

enter:

```
SHOW ALIAS-TYPES ;
```

Alias details display in the order they were specified by the Controller.

To display the alias details in the form of a CONTROL NEW-ALIAS command whose output can be used to transfer the details to another dictionary, enter:

```
SHOW ALIAS FOR INPUT ;
```

To display internal reference numbers which you may need if you refer a problem to MSP, enter:

```
SHOW ALIAS FOR MSP ;
```

SHOW ALIAS-TYPES Syntax

```
>>──SHOW ALIAS-TYPES ──┬──────────────┬──┬─ ; ─┬──>
                       ├─FOR INPUT────┤  └─ . ─┘
                       └─FOR MPS──────┘
```

SHOW MEMBER-TYPE (CMR-CM01)

The SHOW MEMBER-TYPE command displays the variables for and structure of specified member types.

Refer to "SHOW MEMBER-TYPE Syntax" on page 241 for syntax of the SHOW MEMBER-TYPE command.

The SHOW MEMBER-TYPE Command

The SHOW MEMBER-TYPE command is used to output the variable names or structure for a particular member type, for use with Executive Commands.

To output the variable names of the member type, ITEM, enter:

```
SHOW MEMBER-TYPE VARIABLES FOR MEMBER-TYPE ITEM ;
```

You must use a member-type's interrogate keyword, as output by the SHOW UDS MEMBER-TYPES command, when entering this command.

To output the variables names for COMMON-CLAUSES, enter:

```
SHOW MEMBER-TYPE VARIABLES FOR COMMON-CLAUSES ;
```

In the SHOW MEMBER-TYPE command you can optionally omit the keyword, VARIABLES, for example, to show details of the variables available for a PROGRAM member type you need to specify:

```
SHOW MEMBER-TYPE FOR MEMBER-TYPE PROGRAM;
```

To output the structure of a GROUP member type, enter:

```
SHOW MEMBER-TYPE STRUCTURE FOR MEMBER-TYPE GROUP ;
```

To output details of the names of a member type enter:

```
SHOW MEMBER-TYPE NAMES FOR MEMBER-TYPE GROUP;
```

To select a UDS table (other than the one in control of the dictionary), enter:

```
SHOW MEMBER-TYPE TABLE Dunnn VARIABLES FOR selection ;
```

To obtain details of the base member-type (i.e. excluding any UDA), enter:

```
SHOW MEMBER-TYPE VARIABLES BASED selection ;
```

This gives the ASG defined DictionaryManager keywords, but excludes any UDA details.

Output from the SHOW MEMBER-TYPE Command

The command:

```
SHOW MEMBER-TYPE VARIABLES for-clause ;
```

generates two columns of output. The left-hand column, headed DATA DEFINITION, lists components of a member type's definition statement. The right-hand column, headed DICTIONARYMANAGER NAMES, lists the corresponding variable names to be used in Executive Routines. Variables are shown in lower case, keywords in upper case.

Note:

Where the data definition name includes a hyphen the equivalent Executive Command Variable must use an underscore.

For example:

```
SHOW MEMBER-TYPE FOR MEMBER-TYPE GROUP ;
```

generates this output:

```
DATA DEFINITIONDICTIONARYMANAGER NAMES
--------------------------------------
NOT-ALIGNED  ALIGNMENT
         R-->CONTAINS
CONTAINS
  integer bound   INTEGER_BOUND
  name bound   NAME_BOUND
  member-name   CONTAINS_NAME
  version   CONTAINS_VERSION
  KNOWN-AS name   KNOWN_AS
  ALIGNED/
  UNALIGNED/
  NOT-ALIGNED   CONTAINS_ALIGNMENT
          R-->INDEXED-BY
  INDEXED-BY
    name       INDEXED_BY_NAME
          R-->CONDITION
  IF/
```

```
  AND/
  OR        CONDITION
    member-nameCOMPARATOR
    versionCOMPARATOR_VERSION
    EQ etcOPERATOR
    member-nameCOMPARAND_NAME
    signed integerCOMPARAND_VALUE
    numberCOMPARAND_VALUE
    'string' COMPARAND_VALUE
    versionCOMPARAND_VERSION
                              R-->ELSE
  ELSE
    integer boundINTEGER_BOUND
    name boundNAME_BOUND
    member-nameCONTAINS_NAME
    integerCONTAINS_VERSION
    KNOWN-AS nameKNOWN_AS
    ALIGNED/
    UNALIGNED/
    NOT-ALIGNEDCONTAINS_ALIGNMENT
           R-->INDEXED-BY
    INDEXED-BY
      name  INDEXED_BY_NAME
           R-->CONDITION


IF/
 AND/
 OR      CONDITION
  member-nameCOMPARATOR
  version COMPARATOR_VERSION
 EQ etc   OPERATOR
 member-nameCOMPARAND_NAME
 signed integerCOMPARAND_VALUE
 number   COMPARAND_VALUE
 'string' COMPARAND_VALUE
 version  COMPARAND_VERSION
```

where:

`T-->` represents the name of a text attribute to be DRETRIEVEd.

`R-->` represents the name of a repeating clause to be DRETRIEVEd.

The entries under the name of a clause to be DRETRIEVEd represent the names of the variables generated for entries in that clause.

`etc` is used where more than three keywords exist for an attribute. For example, the clause 'ENTERED-AS etc' represents ENTERED-AS, HELD-AS,

REPORTED-AS and DEFAULTED-AS. The name of the variable generated by the Executive Command is 'FORM'.

Indentation is used to indicate subsequent occurrences of variables in repeating clauses.

To output the STRUCTURE of the member type GROUP, enter:

```
SHOW MEMBER-TYPE STRUCTURE FOR MEMBER-TYPE GROUP ;
```

This is the output produced:

```
R-->KEYS
R-->CONTAINS
   R-->INDEXED-BY
   R-->CONDITION
   R-->ELSE
      R-->INDEXED-BY
      R-->CONDITION
```

To output the variable NAMES of the member type GROUP, enter:

```
SHOW MEMBER-TYPE NAMES FOR MEMBER-TYPE GROUP ;
```

This is the output produced:

```
NAME           CLAUSE/ATTR        TYPE
----------------------------------------------
ENTERED-AS     ATTRIBUTE          KEYWORD
HELD-AS        ATTRIBUTE          KEYWORD
REPORTED-AS    ATTRIBUTE          KEYWORD
DEFAULTED-AS   ATTRIBUTE          KEYWORD
USER-EXIT      ATTRIBUTE          MEMBER-NAME
KEYS           CLAUSE             LIST
ALIGNED        ATTRIBUTE          KEYWORD
UNALIGNED      ATTRIBUTE          KEYWORD
NOT-ALIGNED    ATTRIBUTE          KEYWORD
CONTAINS       CLAUSE             LIST
```

SHOW MEMBER-TYPE Syntax

```
>>──SHOW MEMBER-TYPE ──┬──────────┬──┬──────────┬──────────────────────────>
                       └─TABLE x──┘  ├─VARIABLE─┤
                                     ├─STRUCTURE─┤
                                     └─NAMES─────┘

>──┬─FOR───┬──┬─MEMBER-TYPE member-type──┬──────┬─;─┬──><
   └─BASED─┘  └─COMMON-CLAUSES────────────┘      └─.─┘
```

SHOW PRODUCE-OPTIONS (CMR-CM01)

The SHOW PRODUCE-OPTIONS command displays the currently active settings for each control option of the PRODUCE command.

The command has the form:

```
SHOW PRODUCE-OPTIONS FOR language ;
```

where *language* is any one of the programming languages, database management system languages or file management system languages supported by Manager Products' source language generation capabilities.

For example:

```
SHOW PRODUCE-OPTIONS FOR COBOL ;
```

or:

```
SHOW PRODUCE-OPTIONS FOR IMS PSBGEN ;
```

The settings displayed are those currently specified in the installation macro relevant to language (DGCOB, DGPLI or DGBAL for programming languages, DGREC for language-independent record layouts, or the corresponding macros for database or file management system languages). These settings may be overridden by control options in the PRODUCE command.

If language is a programming language or is RECORD-LAYOUTS, it may be qualified by SEGMENT-OPTIONS or IOGEN. Either of these keywords relates to the tailoring of data descriptions output for segment input/output areas for IMS (DL/I). The settings displayed are then those currently specified in the macro DGSCOB, DGSPLI, DGSBAL or DGSREC, depending on language. For example:

```
SHOW PRODUCE-OPTIONS FOR PL/I IOGEN ;
```

displays the settings of the parameters in DGSPLI.

SHOW PRODUCE-OPTIONS Syntax

```
>>──SHOW PRODUCE-OPTIONS FOR──────────────────────────────────────>><
>──┬─ TOTAL ──────────────────────────────────────────────┬──┬─ ; ─┬──>><
  ├─ADABAS ──────────────────────────────────────────────┤  └─ . ─┘
  ├─MARKIV ──────────────────────────────────────────────┤
  ├─MARK-IV ─────────────────────────────────────────────┤
  ├─SYSTEM2000 ──────────────────────────────────────────┤
  ├─S2K                                                  │
  │                                                      │
  ├─┬─COBOL ──────────────┬──┬─SEGMENT-OPTIONS─┬──────────┤
  │ ├─PL/I ───────────────┤  └─IOGEN ──────────┘          │
  │ ├─PL/1 ───────────────┤                               │
  │ ├─PLI ────────────────┤                               │
  │ ├─PL1 ────────────────┤                               │
  │ ├─BAL ────────────────┤                               │
  │ ├─ASSEMBLER ──────────┤                               │
  │ ├─ASSEMBLY ───────────┤                               │
  │ ├─ALC ────────────────┤                               │
  │ ├─RECORD-LAYOUTS ─────┤                               │
  │ │                     │                               │
  ├─┬─IMS ──────┬──┬─DATABASE-DESCRIPTIONS ──────────────┤
  │ ├─DL/I ─────┤  ├─DBDGEN ──────────────────────────────┤
  │ ├─DL/1 ─────┤  ├─PROGRAM-SPECIFICATIONS ──────────────┤
  │ ├─DLI ──────┤  └─PSBGEN ──────────────────────────────┤
  │ └─DL1 ──────┘                                         │
```

SHOW UDS (CMR-CM01)

The SHOW UDS command displays all or part of the repository's UDS table.

Refer to "SHOW UDS Syntax" on page 247 for the syntax of the SHOW UDS command.

Introduction to the SHOW UDS Command

A UDS table defines the members that can be entered into the repository. It includes the following information:

- The keywords and literals for each member type
- The hierarchical positions of the member types
- Entity-association (EA) relationship types for each member type and the member types to which they can refer
- Entity-relationship (ER) relationship types (including properties)
- Alternative relationship types
- User-defined attribute types.

Refer to *ASG-MethodManager Administration* for details of relationship types.

Use the SHOW UDS command to display all or part of the UDS table.

Displaying the Full UDS Table

To display the full UDS table, enter:

```
SHOW UDS FULL ;
```

You should use this variant of the SHOW command with caution, as it may generate a large volume of output.

Displaying Member Types

To display:

- The keywords and literals for each member type
- The member-type hierarchy
- EA relationship types for each member type and the member types to which they can refer

enter:

```
SHOW UDS MEMBER-TYPES ;
```

To display only the keywords associated with each member type, enter:

```
SHOW UDS MEMBER-TYPES KEYWORDS ;
```

To select the type of keywords to be displayed, enter:

```
SHOW UDS MEMBER-TYPES keyword-type KEYWORDS ;
```

where *keyword-type* is one or more of the following separated by commas:

- REPORT-DOWN-TO
- INTERROGATE
- ENCODE.

To display only the literals associated with each member type, enter:

```
SHOW UDS MEMBER-TYPES LITERALS ;
```

To select the types of literals to be displayed, enter:

```
SHOW UDS MEMBER-TYPES literal-type LITERALS ;
```

where *literal-type* is one or more of the following, separated by commas:

- STANDARD
- SHORT
- LONG
- PLURAL.

To display the member-type hierarchy, enter:

```
SHOW UDS MEMBER-TYPES HIERARCHY ;
```

The members composing the hierarchy are listed in ascending order. For each member type the following displays:

- The standard literal
- The base member type
- The level number
- Whether it is recursive.

To list all member types, with their EA relationship types, enter:

```
SHOW UDS MEMBER-TYPES RELATIONSHIPS ;
```

The order of member types in the list is as follows:

- ER relationship types
- Entity types.

To display selected member types, with their EA relationship types, enter:

```
SHOW UDS MEMBER-TYPES RELATIONSHIPS FOR member-type-list ;
```

where *member-type-list* is one or more member types, separated by commas.

The above commands may be combined. For example:

```
SHOW UDS MEMBER-TYPES ENCODE KEYWORDS SHORT LITERALS HIERARCHY ;
```

displays a list of all the member types, giving their ENCODE keywords and SHORT literals, followed by the member-type hierarchy. The keywords and literals are both listed together. The member-type hierarchy and the relationship types for each member type are listed separately.

Displaying User-Defined Attributes Types

To list the member types that have user-defined attribute types, enter:

```
SHOW UDS USER-DEFINED ATTRIBUTES ;
```

The information given for each attribute type is as follows:

- The attribute type name
- The permitted values of attributes of that type
- The attribute type's optionality.

To find the user-defined attribute types associated with given member types, enter:

```
SHOW UDS USER-DEFINED ATTRIBUTES FOR member-type-list ;
```

where *member-type-list* is one or more member types, separated by commas.

To provide more detailed information about user-defined attribute types, enter:

```
SHOW UDS USER-DEFINED ATTRIBUTES GIVING DETAILS ;
```

Particular member types may also be specified.

To list member types that have given user-defined attribute types, enter:

```
SHOW UDS USER-DEFINED ATTRIBUTES IDENTIFIED-BY attribute-id ;
```

where *attribute-id* is one or more attribute types, separated by commas.

If both FOR and IDENTIFIED-BY options are used, both conditions must be satisfied for a given member to be listed.

Displaying ER Relationship Types

To display:

- ER relationship types (including properties)
- Alternative relationship types

enter:

```
SHOW UDS USER-DEFINED RELATIONSHIPS ;
```

For full details of defining ER relationship types, refer to *ASG-MethodManager Administration*.

Displaying UDS Tables Other Than the Repository's Current One

By default the SHOW UDS command displays information from the repository's current UDS table. To display information from other UDS tables, enter:

```
SHOW UDS TABLE table-name options ;
```

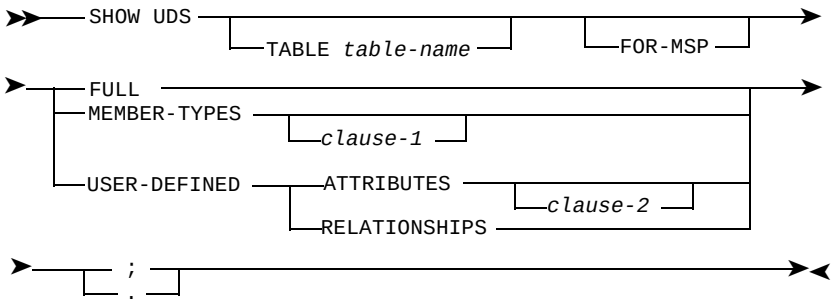
where:

table-name is the name of the UDS table that you want information about.

options specifies the information you require.

The TABLE keyword is restricted to the repository Controller.

SHOW UDS Syntax



where *table-name* is the name of a UDS table

Note:

The TABLE keyword is restricted to the repository Controller.

clause-1 is:

```
➤──┬──────────────
```

SOURCE-FILE (CMR-FE01)

The SOURCE-FILE command enters the contents of an Update Buffer into the dictionary without generating an encoded record.

If you have updated an existing member, enter:

```
SOURCE-FILE ;
```

or

```
SFILE ;
```

Your changes are then applied to the member's source record in the dictionary. However, the encoded record of the member (if there is one) in the Data Entries dataset is not changed.

If you have created a new member, enter:

```
SOURCE-FILE member-name ;
```

or

SFILE *member-name* ;

where *member-name* is the name of a repository member.

Records for this new member are then added to the source dataset of the dictionary. No encoded record is generated.

The member-name can have a maximum of 32 characters. It must not be the same as that of any other member of the current dictionary.

You can tailor the environment so that the source of the member displays in a Lookaside Buffer when you successfully enter the SFILE command by using the SET UPDATE-OUTPUT command.

You cannot source-file a dictionary member that is locked by another user. If the member has been locked since you updated it, the SOURCE-FILE command is rejected.

Refer to the *ASG-ControlManager User's Guide* for details of member naming rules and the SET and QUERY UPDATE-OUTPUT commands.

Refer to "LOCK (CMR-WS01)" on page 173 for details of member locking.

SOURCE-FILE Syntax

```
>>─┬─SFILE      ──┬──┬──────────────┬──┬──;──┬──────────────><
   └─SOURCE-FILE─┘  └─member-name──┘  └──.──┘
```

where *member-name* is the name of a dictionary member.

STORE KEPT-DATA (DMR-DD1)

The STORE KEPT-DATE command stores named and unnamed KEPT-DATA lists on the MP-AID.

To store the unnamed KEPT-DATA list on the MP-AID, enter:

```
STORE KEPT-DATA ONTO mpaid-name ;
```

where *mpaid-name* is the name of the MP-AID KEPT-DATA member in which the list is to be stored. It must not be the same as an existing KEPT-DATA member.

To store a named KEPT-DATA list on the MP-AID, enter:

```
STORE KEPT-DATA IN list-name ONTO mpaid-name ;
```

where *list-name* is the name of a KEPT-DATA list.

To store a KEPT-DATA list onto a KEPT-DATA member that already exists on the MP-AID, replacing the existing contents of the KEPT-DATA member, enter:

```
STORE KEPT-DATA ONTO mpaid-name REPLACE ;
```

or

```
STORE KEPT-DATA IN list-name ONTO mpaid-name REPLACE ;
```

You can fetch a KEPT-DATA member from the MP-AID into a KEPT-DATA list using the FETCH KEPT-DATA command.

If you STORE a KEPT-DATA list in one dictionary and retrieve into another, the retrieved list includes only those index-names that exist in the stored KEPT-DATA list and the second dictionary.

If the Audit and Security facility (selectable unit CMR-DD3) is installed,

KEPT-DATA lists retrieved from the MP-AID, or another dictionary, include only those index-names that you can access.

Refer to "FETCH KEPT-DATA (DMR-DD1)" on page 132 for details of fetching stored KEPT-DATA lists.

Refer to *ASG-Manager Products Status Concepts* for the effects of status on KEPT-DATA lists.

Refer to "The Security System" on page 17 for details of security.

STORE KEPT-DATA Syntax

```
>>──STORE KEPT-DATA ┬──────────────┬─ONTO mpaid-name──────>
                    └─IN list-name─┘

>──┬─────────┬┬─;─┬──────────────────────────────────────><
  └─REPLACE─┘└───┘
```

where:

list-name is the name of a KEPT-DATA list

mpaid-name is the name of the MP-AID KEPT-DATA member in which the KEPT-DATA is to be stored.

UPDATE (CMR-FE01)

The UPDATE command creates new dictionary members or updates existing ones.

To create a new dictionary member, enter:

```
UPDATE ;
```

This opens a buffer in Update Mode into which you insert text.

To update an existing dictionary member, enter:

```
UPDATE member-name ;
```

This opens a buffer in Update Mode containing a copy of the source record of the named dictionary member; you can then update the source record.

If LOOKASIDE-RETENTION is set off and you enter an UPDATE command, any Lookaside Buffers currently open will be closed.

If LOOKASIDE-RETENTION is set on and you enter an UPDATE command, any Lookaside Buffers currently open will be retained (up to the maximum buffer count).

To add the contents of the buffer to the dictionary use FILE, RFILE or SFILE. To abandon the update without adding the contents to the dictionary use QUIT or XQUIT.

If the Workstation Interface facility is installed, (selectable unit CMR-WS01) you cannot update a member that is locked by another user. If member-name is locked, the UPDATE command is rejected and processing continues with the next command.

Refer to *ASG-ControlManager User's Guide* for details of the SET LOOKASIDE-RETENTION command.

Refer to "LOCK (CMR-WS01)" on page 173 for details of member locking.

UPDATE Syntax

```
>>──UPDATE ──┬──────────────┬──┬──;──┬──────────────────────────><
             └─member-name──┘  └──.──┘
```

where *member-name* is the name of an existing dictionary member.

VALIDATE (MMR-PF61)

The VALIDATE command checks the integrity of a member and sets its integrity state.

To check the integrity of a member and set its integrity state, enter:

```
VALIDATE MEMBER member-name ;
```

where *member-name* is the name of a repository member.

If the member passes the checks it is marked as check-ok.

If it fails the checks, it is marked as check-needed and messages are output saying why it failed.

For further details of integrity refer to *ASG-MethodManager Administration*.

VALIDATE Syntax

```
>>──VALIDATE MEMBER member-name──┬─;─┬──────────────────────><
                                 └─.─┘
```

where `member-name` is the name of a repository member.

WBDA

Refer to "MODE, QUERY MODE, DMS, WBDA (CMR-CM01)" on page 177 for details of the WBDA command.

WHAT (DMR-DD1)

The WHAT command outputs index-names that meet specific conditions, or identifies index-names.

Refer to "WHAT Syntax" on page 260 for the syntax of the WHAT command.

Introduction to the WHAT Command

The WHAT command outputs index-names which meet specific conditions.

Using the IS keyword, you can display the identity of a particular index-name; for example, whether it is a member, an ALIAS, a CATALOG classification, or an indexed attribute.

The USES and CONSTITUTES keywords allow you to find out which encoded members use or are used by a specified encoded member, via any relationship type.

For further details of relationship types refer to *ASG-MethodManager Administration*.

If you want to interrogate via only:

- A specific relationship type
- A specific ER relationship type
- A specific EA relationship type

then use the WHICH command.

Normally, the current DIRECTLY-ASSUMPTION setting (changed by the SET DIRECTLY-ASSUMPTION command) determines whether only direct references, or both direct and indirect references, are selected. However, you can specify either DIRECTLY or INDIRECTLY in the WHAT command to override this default.

You can display the encoded members which have the specified catalog classification(s), by using the FORMS keyword. You can combine this with further conditions with the keywords AND, OR, and NOT. These conditions can be any available with the WHICH command, a larger set than that normally available with the WHAT command.

By specifying the BELONGS-TO keyword, you can find out which encoded members belong to (are owned by) a specified person or department.

The LOCALLY keyword allows you to restrict the WHAT command selection to those encoded members within your Status Window for selection.

Refer to "Definition Of Index-names" on page 316 for a definition of index-names.

Refer to *ASG-ControlManager User's Guide* for details of the SET DIRECTLY-ASSUMPTION command.

Selecting Encoded Members that Use a Given Member

To select encoded members that use a given member, enter:

```
WHAT USES member-name ;
```

where *member-name* is the name of a repository member.

The members selected depend on the current DIRECTLY-ASSUMPTION setting.

To select encoded members that directly use the specified member, regardless of the current DIRECTLY-ASSUMPTION setting, enter:

```
WHAT DIRECTLY USES member-name ;
```

or

```
WHAT USES member-name DIRECTLY ;
```

If you want to select encoded members that directly and indirectly use the specified member, regardless of the current DIRECTLY-ASSUMPTION setting, enter:

```
WHAT INDIRECTLY USES member-name ;
```

or

```
WHAT USES member-name INDIRECTLY ;
```

A reference path is not traced beyond a member that is used by a member at a lower level in the member type hierarchy (via, for example, a USER-EXIT clause), or beyond a member already selected for output. Use the SHOW UDS MEMBER-TYPES HIERARCHY command to list the hierarchy.

The list of member names which is output is indented to show the reference paths to the specified member.

The security facility controls access to information held in the repository. For details of how security affects interrogations refer to "The Security System" on page 17.

Selecting Members Used By a Specified Encoded Member

To select members used by a specified encoded member, enter:

```
WHAT CONSTITUTES member-name ;
```

where *member-name* is the name of an encoded repository member.

The members selected depend on the current DIRECTLY-ASSUMPTION setting.

If you want to select members directly used by the specified member, regardless of the current DIRECTLY-ASSUMPTION setting, enter:

```
WHAT DIRECTLY CONSTITUTES member-name ;
```

or

```
WHAT CONSTITUTES member-name DIRECTLY ;
```

If you want to select members directly and indirectly used by the specified member, regardless of the current DIRECTLY-ASSUMPTION setting, enter:

```
WHAT INDIRECTLY CONSTITUTES member-name ;
```

or

```
WHAT CONSTITUTES member-name INDIRECTLY ;
```

A reference path is not traced beyond a member that is used by a member at a lower level in the member type hierarchy (via, for example, a USER-EXIT clause), or beyond a member already selected for output.

The list of member names output is indented to show the reference paths from the specified member.

The security facility controls access to information held in the repository. For details of how security affects interrogations refer to "The Security System" on page 17.

Selecting Only Members in Your Status Window for Selection

To limit a WHAT USES or WHAT CONSTITUTES interrogation to members within your Status Window for Selection, enter:

```
WHAT LOCALLY USES member-name ;
```

or

```
WHAT LOCALLY CONSTITUTES member-name ;
```

The LOCALLY keyword may alternatively come immediately after the USES or CONSTITUTES keyword.

If any members are found that are not within your Status Window for Selection, they, and any members in the reference path beyond them, are ignored, and this message displays on completion of the command:

```
DM01538I nnn ENTRIES NOT SELECTED
```

where *nnn* is the number of times non-local members were found which used/constituted the given member.

Note:
Members in the reference paths beyond non-local members are not included in this count.

If the keywords DIRECTLY or INDIRECTLY are used, they must come immediately before or after the LOCALLY keyword. For example:

```
WHAT DIRECTLY LOCALLY USES EMP-CODE ;
WHAT CONSTITUTES LOCALLY INDIRECTLY DEPT-NAME ;
```

Refer to *ASG-Manager Products Status Concepts* for details of Status Windows.

Showing Relationships

To show the relationships between members, enter:

```
WHAT USES member-name PATH ;
```

or

```
WHAT CONSTITUTES member-name PATH ;
```

where *member-name* is the name of a repository member.

The PATH keyword shows relationships used (both E/A and E/R) and E/R relationship paths passed through.

Selecting Members by Catalog Classification

To select members by catalog classification, enter:

```
WHAT FORMS classification-list ;
```

where *classification-list* is one or more catalog classifications separated by commas.

If two or more classifications are specified, only members having all the specified classifications are selected.

Members with an access security level higher than the user's security level are not listed but are included in the final count of inaccessible members.

A WHAT FORMS command can be combined with other conditions.

Combining Catalog Selection with Further Conditions

To select members with the specified catalog classification(s), which also meet all of the specified conditions, enter:

```
WHAT FORMS classification-list AND condition ;
```

where:

> *classification-list* is one or more catalog classifications separated by commas.
>
> *condition* can be any of the conditions available with the WHAT command or any of the larger set of conditions available with the WHICH command.

If you want to select members that have the specified catalog classification and/or satisfy the specified conditions, enter:

```
WHAT FORMS classification-list OR condition ;
```

A condition can be preceded by the keyword NOT, in which case members that are catalogued under all the specified classifications and/or do not meet the condition are selected.

The keyword DOES (DO) can be included in the command for readability, but it has no effect.

For example:

```
WHAT FORMS MONTHLY AND DOES NOT USE ADMIN ;
```

selects members that have a catalog classification of MONTHLY and do not refer to the member ADMIN.

AND and OR conditions can be combined. Processing, unless overridden by parentheses, goes from left to right.

For example:

```
WHAT FORMS MONTHLY AND USES ADMIN OR FORMS WEEKLY AND USES PERSONNEL ;
```

which is as if you could enter:

```
WHAT ((FORMS MONTHLY AND USES ADMIN) OR FORMS WEEKLY) AND USES
PERSONNEL ;
```

selects out of members that:

- Have both a classification of MONTHLY and use ADMIN, or
- Have a classification of WEEKLY

only those that use the member PERSONNEL.

Parentheses can be used to establish an alternative sequence of evaluation. The number of open parentheses, (, must equal the number of closed parentheses,). There is no restriction on the depth of nesting of parentheses.

For example:

```
WHAT FORMS MONTHLY AND USES ADMIN OR (FORMS WEEKLY AND USES
PERSONNEL) ;
```

selects the members cataloged under the classification MONTHLY that use ADMIN and/or those with a classification of WEEKLY that use PERSONNEL.

Selecting Members that Belong to a Specified Owner

To select members that belong to a specified owner, enter:

```
WHAT BELONGS-TO owner-name ;
```

where *owner-name* must be the name of an owner recorded in the repository.

Members with an access security level higher than the user's security level are not listed in the output of the command but are included in the final count of inaccessible members.

Finding Out What an Index-Name Is

To find out whether an index-name is a member name, alias, catalog classification or user-defined indexed attribute, enter:

```
WHAT IS index-name ;
```

If *index-name* is an alias, you will also be told the number or alias-type of the alias and the member name and member type of each member for which it is an alias.

If the index-name in the command is the member name or alias of a protected member with an access security level higher than the user's security level, one of these messages is output:

```
member-name IS AN INACCESSIBLE MEMBER
```

or

```
alias IS THE ALIAS OF n INACCESSIBLE MEMBER
```

WHAT Examples

To select encoded members that, depending on the current DIRECTLY-ASSUMPTION setting, directly, or both directly and indirectly, use EMP-CODE, enter:

```
WHAT USES EMP-CODE ;
```

To select encoded members that directly use EMP-CODE, enter:

```
WHAT DIRECTLY USES EMP-CODE ;
```

To select encoded members that directly and indirectly use EMP-CODE, enter:

```
WHAT USES EMP-CODE INDIRECTLY ;
```

To select members that, depending on the current DIRECTLY-ASSUMPTION setting, are directly, or both directly and indirectly, used by DEPT-NAME, enter:

```
WHAT CONSTITUTES DEPT-NAME ;
```

To select members that are directly used by DEPT-NAME, and are within your Status Window for selection, enter:

```
WHAT DIRECTLY LOCALLY CONSTITUTES DEPT-NAME ;
```

To select members that are directly and indirectly used by DEPT-NAME, enter:

```
WHAT INDIRECTLY CONSTITUTES DEPT-NAME;
```

To show the relationships used or passed through by ENT-SOURCE, enter:

```
WHAT CONST ENT-SOURCE PATH ;
```

The results returned might be as follows:

```
ENTITY    ENT-SOURCE REFERS TO
  ENTITY  ENT-TARGET VIA BUSINESS RELATIONSHIP BUSREL
  BUSINESS RELATIONSHIPBUSREL-2   VIA SEE
                       THRU BUSINESS RELATIONSHIP BUSREL
ENTITY    ENT-TARGET REFERS TO NO MEMBERS
          BUSREL-2 REFERS TO NO MEMBERS
```

To select members that are catalogued under both the classifications BATCH and WEEKLY, enter:

```
WHAT FORMS BATCH, WEEKLY ;
```

To select members that belong to owner PERSONNEL, enter:

```
WHAT BELONGS-TO PERSONNEL ;
```

To select members that are catalogued under the classification WEEKLY and directly use EMP-CODE, enter:

```
WHAT FORMS WEEKLY AND DIRECTLY USES EMP-CODE ;
```

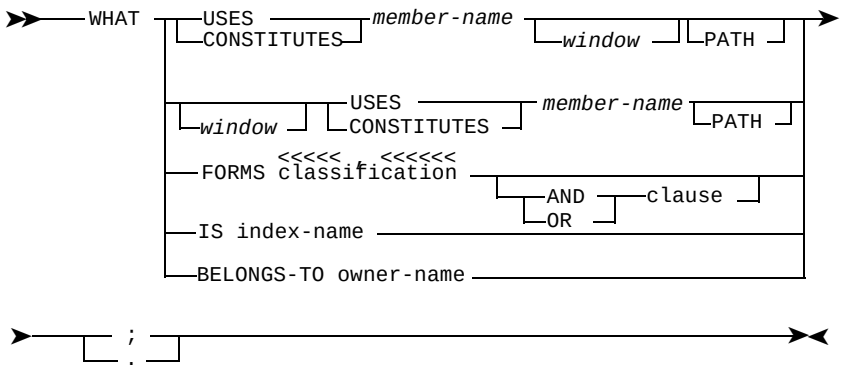
To find out what type of index-name S-S-CODE is, enter:

```
WHAT IS S-S-CODE ;
```

To select members that are catalogued under the classification COBOL and directly constitute DMC28 via a relationship type or a relationship-type class CALLS, enter:

```
WHAT FORMS COBOL AND DIRECTLY CONSTITUTES DMC28 VIA CALLS ;
```

WHAT Syntax



where:

member-name is the name of a repository member

window is:

```
>>-+-------------+-DIRECTLY-+-+-----------+-+-------------+-+-------------+-+-------
```

references-clause is:

```
➤┬─USES        ──┬──name ──┬──────────┬──────────────────────────────────────➤
 │ CONSTITUTE ────┘         └─form ────┘└─VIA rel──┘                          │
 │                                                                             │
 └─INPUTS      ──┬──name ──┬──────────┬──────────────────────────────────────┘
   OUTPUTS       │         └─form ────┘
   UPDATES       │
   CONTAINS      │
   CALLS         │
   PASSES        │
   FATHERS       │
   GENERATES     │
   GENERATES     │
   INPUT-BY      │
   OUTPUT-BY     │
   UPDATED-BY    │
   CONTAINED-BY  │
   CALLED-BY     │
   PASSED-BY     │
   FATHERED-BY   │
   GENERATED-BY ─┘
➤──┬─DIRECTLY   ──┬──────────────────────────────────────────────────────────➤
   └─INDIRECTLY ──┘
```

where:

name is the name of a repository member

form is:

```
➤──┬─DEFAULTED-AS ──────────────────────────────────────────────────────────➤
   │                                                                          │
   └─┬─ENTERED-AS ──┬──────────┬──────────────────────────────────────────┬──┘
     │ HELD-AS    ──┘          └─VERSION ──┘                version ──┘
     └─REPORTED-AS ──┘
```

where:

version is an unsigned integer in the range 1 to 15

rel is a relationship type or relationship-type class.

classification is a catalog classification

attribute-specification is:

```
                  <<<
```

where:

operator is:

```
➤──┬─EQUALS──────────────────────────────────────────────────────➤
   ├─=─────────┤
   ├─EQ────────┤
   ├─NE────────┤
   ├─GT────────┤
   ├─>─────────┤
   ├─GE────────┤
   ├─LT────────┤
   ├─<─────────┤
   └─LE────────┘
```

string is as defined above.

value-attr is:

```
➤──┬─ALIAS──┬─┬─alias-type─┬─┬──────────────────────────────────➤
   │        │ ├─number─────┤ │
   │        │ │            │ │
   ├─alias-type ALIAS────────┤ │
   ├─CATALOGUE───────────────┤ │
   └─attribute───────────────┘ │
```

where:

alias-type is one of the specific alias-type keywords available in your repository

number is an integer from 1 to the maximum number of general aliases allowed in your repository.

(Use the SHOW ALIAS-TYPES command to find out the specific alias-types and the number of general aliases available in your repository.)

attribute is any attribute in a member definition that contains neither text nor a string (see output of SHOW MEMBER-TYPE command).

v-relation is:

```
➤──operator <value, <──────────────────────────────────────────➤
```

where:

operator is defined above

value is any variable appropriate to the value-attribute.

owner-name is the name of an owner

operator is defined above

n1 is an unsigned integer in the range 0 to 2,147,483,647

list-name is the name of a KEPT-DATA list.

index-name is the name of a repository member, or an alias or a catalog classification, or a user-defined indexed attribute.

WHICH (DMR-DD1)

The WHICH command selects index-names that do or do not meet a condition or set of conditions.

Refer to "WHICH Syntax" on page 285 for the syntax of the WHICH command.

Introduction to the WHICH Command

The WHICH command outputs a selection of index-names that meet or do not meet one or more conditions.

You can specify any of the following selections:

- INDEX-NAMES: all index-names
- ALIASES, ATTRIBUTES, CATALOGS: specific index-names
- MEMBERS: all encoded and dummy members
- LOCKED: locked encoded and dummy members
- CHECK-OK, CHECK-NEEDED: members that have passed or need integrity checks
- KEPT-DATA (IN): index-names kept in a KEPT-DATA list
- EXCEPT: excluding a category of index-names from output
- GENERIC: members based on a specified member type.

You can also select members by their member type.

You can further restrict this selection by specifying one or more of the following secondary selections:

- Name: selecting index-names whose names meet given criteria
- Status: selecting index-names that exist, or were processed, in specific statuses
- Time: selecting index-names that were, or were not, processed at a given time or during a given period of time
- User: selecting index-names that were, or were not, processed by a certain user.

You can specify these conditions with the above selections:

- USES, VIA: selecting encoded members that use a specified index-name via a specified relationship type or relationship class
- CONSTITUTES, VIA: selecting index-names that are used by a specified encoded member via a specified relationship type or relationship class
- FORMS: selecting members that have a specified catalog classification
- HAVE, SPECIFIED: selecting encoded members containing a specified clause, or a specific clause containing a specified string, value or text
- BELONGS-TO: selecting encoded members belonging to a specified owner
- ARE REFERRING-TO: selecting encoded members that use a specified number of index-names
- ARE USED-BY: selecting index-names that are used by a specified number of encoded members
- ARE KEPT (IN): selecting index-names that are kept in a KEPT-DATA list.

You can specify a combination of the above conditions with the keywords AND, OR, and NOT.

The current DIRECTLY-ASSUMPTION setting (changed by the SET DIRECTLY-ASSUMPTION command) determines whether only direct, or both direct and indirect, references are selected with the USES and CONSTITUTES keywords.

WHICH Command Selection

To select index-names that meet a condition, enter:

```
WHICH selection condition ;
```

where *selection* is one of:

- MEMBERS
- INDEX-NAMES
- KEPT-DATA
- KEPT-DATA IN *list-name*
- *category*
- EXCEPT *category*.

LOCKED, CHECK-NEEDED and CHECK-OK can precede selection. These keywords are explained later in this section.

where:

list-name is the name of a KEPT-DATA list.

category is one or more of the following, separated by commas:

- ALIASES
- *alias-type* ALIASES
- ALIASES *alias-type*
- ATTRIBUTES
- ALIASES *number*
- CATALOGS/CATALOGUES
- member-type
- *collective-member-type*

You can list the member types and collective member types using the SHOW UDS MEMBER-TYPES command.

The category ALIASES covers all aliases of all encoded members. An ALIASES category can optionally be restricted to specific aliases of a stated type, or to general aliases of a stated number. Your selection must not include more than one ALIASES category.

where:

number is an unsigned integer in the range 1 to the maximum number of general aliases allowed in your repository.

alias-type is one of the specific alias-types available in your repository.

You can display the specific alias types and the number of general aliases available in your repository, by using the SHOW ALIAS-TYPES command.

The category ATTRIBUTES covers all indexed attributes of encoded members.

To select all encoded and dummy members that meet a condition, enter:

```
WHICH MEMBERS condition ;
```

To select members of a particular type that meet a condition, enter:

```
WHICH member-type condition ;
```

To select from members based on a specified member type, enter:

```
WHICH GENERIC member-type condition ;
```

where *member-type* is the interrogate keyword of any member type in your repository.

To select all encoded members that are locked and meet a condition, enter:

```
WHICH LOCKED MEMBERS condition ;
```

To select check-ok members, members that have passed integrity checks, include the CHECK-OK keyword in the command.

To select check-needed members, members that need integrity checks, include the CHECK-NEEDED keyword in the command.

For further details of check-ok and check-needed members, refer to *ASG-MethodManager Administration*.

To select index-names that meet a condition, with the exception of those of selected categories, enter:

```
WHICH EXCEPT category condition ;
```

To select index-names that meet a condition, enter:

```
WHICH INDEX-NAMES condition ;
```

If you want to select index-names in a KEPT- DATA list that meet a condition, enter:

```
WHICH KEPT-DATA condition ;
```

or

```
WHICH KEPT-DATA IN list-name condition ;
```

where *list-name* is the name of a KEPT-DATA list.

The index-names listed as a result of an interrogation are normally in the order in which they were added to the repository or KEPT-DATA list. To obtain a list in alphanumeric order, enter:

```
WHICH selection ALPHABETICALLY condition ;
```

To select all the index-names in a KEPT-DATA list that are of a given category and meet a condition, enter:

```
WHICH KEPT-DATA category condition ;
```

or

```
WHICH KEPT-DATA IN list-name category condition ;
```

To select all the index-names in a KEPT-DATA list, except for one or more categories, that meet a condition, enter:

```
WHICH KEPT-DATA EXCEPT category condition ;
```

or

```
WHICH KEPT-DATA IN list-name EXCEPT category condition ;
```

KEPT can be a condition as well as a selection.

Selecting Members That Use a Specified Member

To select encoded members that use a specified member, enter:

```
WHICH selection USES member-name ;
```

selection may be any of the selections available with the WHICH command.

The members selected will depend on the current DIRECTLY-ASSUMPTION setting.

If you want to select encoded members that directly use the specified member, regardless of the current DIRECTLY-ASSUMPTION setting, enter:

```
WHICH selection DIRECTLY USE member-name ;
```

or

```
WHICH selection USES member-name DIRECTLY ;
```

If you want to select encoded members that indirectly and directly use the specified member, regardless of the current DIRECTLY-ASSUMPTION setting, enter:

```
WHICH selection INDIRECTLY USE member-name ;
```

or

```
WHICH selection USES member-name INDIRECTLY ;
```

For example:

```
WHICH GROUPS DIRECTLY USE EMP-CODE ;
```

outputs all encoded GROUP members that directly use the member EMP-CODE.

If member-name is the name of an ITEM, you can select encoded members that use a particular form of the specified member by entering:

```
WHICH selection USE member-name form ;
```

form is any of the following:

ENTERED-AS
REPORTED-AS
HELD-AS
DEFAULTED-AS

To select encoded members that use a particular version of an ITEM, enter:

```
WHICH selection USE member-name VERSION n ;
```

To select encoded members that use a particular form and version of an ITEM, enter:

```
WHICH selection USE member-name form VERSION n ;
```

where *n* is an unsigned integer in the range 1 to 15, specifying the version number of the specified form of the ITEM.

A form keyword is accepted but is not meaningful if member-name is a GROUP or FILE. A version is only accepted if member-name is an ITEM.

A reference path is not traced beyond a member that is used by a member at a lower level in the member type hierarchy (via, for example, a USER-EXIT clause), or beyond a member already selected for output. Use the SHOW UDS MEMBER-TYPES HIERARCHY command to list the hierarchy.

The security facility controls access to information held in the repository. For details of how security affects interrogations refer to "The Security System" on page 17.

Selecting Members That are Used By a Specified Member

To select index-names that are used by a specified encoded member, enter:

```
WHICH selection CONSTITUTES member-name ;
```

selection may be any of the selections available with the WHICH command.

The members selected will depend on the current DIRECTLY-ASSUMPTION setting.

If you want to select index-names that are directly used by the specified encoded member, regardless of the current DIRECTLY-ASSUMPTION setting, enter:

```
WHICH selection DIRECTLY CONSTITUTES member-name ;
```

or

```
WHICH selection CONSTITUTES member-name DIRECTLY ;
```

If you want to select index-names that are indirectly and directly used by the specified member, regardless of the current DIRECTLY-ASSUMPTION setting, enter:

```
WHICH selection INDIRECTLY CONSTITUTES member-name ;
```

or

```
WHICH selection CONSTITUTES member-name INDIRECTLY ;
```

For example:

```
WHICH INDEX-NAMES CONSTITUTE EMP-CODE ;
```

outputs all the index-names used by the encoded member EMP-CODE.

If member-name is the name of an ITEM, you can select index-names that are used by a particular form of the specified member, by entering:

```
WHICH selection CONSTITUTE member-name form ;
```

form is any of the following:

ENTERED-AS
REPORTED-AS
HELD-AS
DEFAULTED-AS

To select index-names that are used by a particular version of an ITEM, enter:

```
WHICH selection CONSTITUTE member-name VERSION n ;
```

To select index-names that are used by a particular form and version of an ITEM, enter:

WHICH selection CONSTITUTE member-name form VERSION *n* ;

where *n* is an unsigned integer in the range 1 to 15, specifying the version number of the specified form of the ITEM.

A form keyword is accepted but is not meaningful if member-name is a GROUP or FILE. A version is only accepted if member-name is an ITEM.

A reference path is not traced beyond a member that is used by a member at a lower level in the member type hierarchy (via, for example, a USER-EXIT clause), or beyond a member already selected for output. Use the SHOW UDS MEMBER-TYPES HIERARCHY command to list the hierarchy.

The security facility controls access to information held in the repository. For details of how security affects interrogations refer to "The Security System" on page 17.

Selecting Members That Use a Specified Member Via a Specified Relationship Type or Relationship Class

To select encoded members that use a specified member via a specified relationship type or relationship class, enter:

```
WHICH selection USE member-name keyword VIA name ;
```

where:

selection is any of the selections available with the WHICH command.

keyword is one of the following keywords:

ASSOCIATED
RELATED
CONNECTED

These keywords have the following meanings:

ASSOCIATED *name* is the name of an EA relationship type

RELATED *name* is the name of an ER relationship type or a relationship class

CONNECTED *name* is the name of a relationship type or a relationship class

If keyword is omitted then the CONNECTED keyword is assumed.

where *name* is defined above.

For example, to select members that use DMC28 via an ER relationship type or relationship class CALLS, enter:

```
WHICH MEMBERS USE DMC28 RELATED VIA CALLS ;
```

To select members that use DMC28 via an EA relationship type CALLS, enter:

```
WHICH MEMBERS USE DMC28 ASSOCIATED VIA CALLS ;
```

For further details of relationship types and relationship classes refer to *ASG-MethodManager Administration*.

The command has this alternative form:

```
WHICH selection reference member-name ;
```

where *reference* is one of these EA relationship types:

```
INPUTS
UPDATES
CONTAINS
OUTPUTS
CALLS
PASSES
FATHERS
GENERATES.
```

The correspondence between the two forms of the command is shown below:

```
reference member-name- condition member-name VIA name
```

INPUTS member-name	-	USES member-name VIA INPUTS
OUTPUTS member-name	-	USES member-name VIA OUTPUTS
UPDATES member-name	-	USES member-name VIA UPDATES
CONTAINS member-name	-	USES member-name VIA CONTAINS
CALLS member-name	-	USES member-name VIA CALLS
PASSES member-name	-	USES member-name VIA PASSING
FATHERS member-name	-	USES member-name VIA FATHERS
GENERATES member-name	-	USES member-name VIA GENERATES

You can use the clause VIA BOUND to interrogate on the array declarations in the CONTAINS clauses of FILE and GROUP members.

Refer to "GROUP Member Type" on page 351 for details of the CONTAINS clause in GROUP members.

Either of the keywords DIRECTLY or INDIRECTLY can be entered in the command. If INDIRECTLY is specified, used-by pointers are followed to members that indirectly or directly use the specified member VIA the specified relationship or a CONTAINS relationship. On each reference path thus followed, only members lying beyond a reference VIA the specified relationship are output.

The security facility controls access to information held in the repository. For details of how security affects interrogations refer to "The Security System" on page 17.

Selecting Index-Names That are Used by a Specified Member Via a Specified Relationship Type or Relationship Class

To select index-names that are used by a specified member via a specified relationship type or relationship class, enter:

```
WHICH selection CONSTITUTE member-name keyword VIA name ;
```

where:

selection is any of the selections available with the WHICH command.

keyword is one of the following keywords:

```
ASSOCIATED
RELATED
CONNECTED
```

These keywords have the following meanings:

ASSOCIATED	*name* is the name of an EA relationship type
RELATED	*name* is the name of an ER relationship type or a relationship class
CONNECTED	*name* is the name of a relationship type or a relationship class

If keyword is omitted then the CONNECTED keyword is assumed.

where *name* is defined above.

For example, to select members that are used by DMC28 via an ER relationship type or relationship class CALLS, enter:

```
WHICH MEMBERS CONSTITUTE DMC28 RELATED VIA CALLS ;
```

For further details of relationship types and relationship classes refer to *ASG-MethodManager Administration*.

The command has the following alternative form:

```
WHICH selection reference member-name ;
```

where *reference* is one of the following EA relationship types:

```
INPUT-BY
UPDATED-BY
CONTAINED-BY
OUTPUT-BY
CALLED-BY
PASSED-BY
FATHERED-BY
GENERATED-BY.
```

The correspondence between the two forms of the command is shown below:

```
reference member-name- condition member-name VIA name
```

```
INPUT-BY member-name   -  CONSTITUTES member-name VIA INPUTS
OUTPUT-BY member-name  -  CONSTITUTES member-name VIA
OUTPUTS
UPDATED-BY member-name -  CONSTITUTES member-name VIA
UPDATES
CONTAINED-BY member-name- CONSTITUTES member-name VIA
CONTAINS
CALLED-BY member-name  -  CONSTITUTES member-name VIA CALLS
PASSED-BY member-name  -  CONSTITUTES member-name VIA
PASSING
FATHERED-BY member-name- CONSTITUTES member-name VIA
FATHERS
GENERATED-BT member-name-  CONSTITUTES member-name VIA
GENERATES
```

You can use the clause VIA BOUND to interrogate on the array declarations in the CONTAINS clauses of FILE and GROUP members.

Refer to "GROUP Member Type" on page 351 for details of the CONTAINS clause in GROUP members.

Either of the keywords DIRECTLY or INDIRECTLY can be entered in the command. If INDIRECTLY is specified, used-by pointers are followed to members that indirectly or directly use the specified member VIA the specified relationship or a CONTAINS relationship. On each reference path thus followed, only members lying beyond a reference VIA the specified relationship are output.

The security facility controls access to information held in the repository. For details of how security affects interrogations refer to "The Security System" on page 17.

Selecting Members by Catalog Classification

To select members by catalog classification, enter:

```
WHICH selection FORM classification-list ;
```

where:

> *selection* may be any of the selections available with the WHICH command. Non member-type selection categories covering aliases, catalog classifications and user-defined indexed attributes are accepted but are not meaningful.
>
> *classification-list* may be one or more catalog classifications separated by commas.

If two or more classifications are specified, only members that have all the specified classifications are selected.

For example:

```
WHICH KEPT-DATA FORM BATCH, WEEKLY ;
```

outputs those encoded members in the unnamed KEPT-DATA list that have both the catalog classifications BATCH and WEEKLY.

Selecting Members With a Given Clause or Attribute

To select members with a given clause or attribute, enter:

```
WHICH selection HAVE attribute SPECIFIED ;
```

HAVE can be replaced with the alternative keyword HAS.

where:

> *selection* can be any of the selections available with the WHICH command.
>
> *attribute* is the name of any clause or attribute within the definition of the specified member types.

You can list the member definition clauses for specific member types, by using the SHOW MEMBER-TYPE command.

For example:

```
WHICH MEMBERS HAVE DESCRIPTION SPECIFIED;
```

selects members with a DESCRIPTION clause in their definition.

```
WHICH ITEMS HAVE FREQ-PROCESS-VALUE SPECIFIED;
```

selects ITEM members having a FREQ-PROCESS VALUE attribute in their definition (an attribute of the FREQUENCY clause).

Selecting Members Having an Attribute Containing a Specific Value

To select members with an attribute or clause containing a specific value, enter:

WHICH *selection* HAVE *attribute relation* ;

where:

selection is any selection available with the WHICH command.

attribute is a text-attribute, value-attribute, or string-attribute.

A text-attribute is an attribute containing text.

A string-attribute is an attribute containing a string.

A value-attribute is ALIAS, CATALOG, or any attribute that does not contain text or strings.

Use the SHOW MEMBER-TYPE command to display the clauses and attributes of a member type and identify whether they contain a string, a value, or text.

relation is a string-relation, text-relation or value-relation. You must specify a string-relation for a string-attribute, a text-relation for a text-attribute, and a value relation for a value-attribute.

A string-relation is:

```
INCLUDES string(s)
```

or

```
operator string(s)
```

operator is one of the following:

```
EQUALS or EQ or =                 (equal to)
NE                                (not equal to)
GT or >                           (greater than)
GE                                (greater than or equal to)
LT or <                           (less than)
LE                                (less than or equal to)
```

INCLUDES can be replaced with the alternative keyword INCLUDING which has the same effect.

string is one or more character strings, separated by commas.

For example:

```
WHICH MEMBERS HAVE VOLUME INCLUDING ’D00106’, ’D00108’ ;
```

selects members with a VOLUME string-attribute containing either of the strings D00106 or D00108.

A text-relation is:

```
INCLUDES text
```

or

LINES *n* TO *m* INCLUDES *text*

where:

> *text* is a text string.
>
> *n* and *m* are unsigned integers in the range 1 to 32767, which identify a character string in a text attribute. m can be END. TO m is optional.

For example:

```
WHICH MEMBERS HAVE DESCRIPTION LINES 1 TO 2 INCLUDING 'TAX CODE FOR EACH
                                                       EMPLOYEE’ ;
```

selects members with a DESCRIPTION text-attribute containing the text string TAX CODE FOR EACH EMPLOYEE in lines 1 or 2 of the DESCRIPTION.

A value-relation is operator followed by value.

where:

operator is as defined above.

value is a single string of characters contained in a value-attribute. Several values, separated by commas, are allowed if the operator is EQUALS or =.

If value-attribute is ALIAS or CATALOG, only the operators EQUALS, = and NE are valid.

For example:

```
WHICH MEMBERS HAVE ALIAS EQ TELEPHONE, ADDRESS ;
```

selects members with an ALIAS of TELEPHONE and/or ADDRESS.

To select several types of member containing the specified attribute or clause, but each with a different relation, enter:

```
WHICH selection HAVE attribute FOR member-type relation FOR member-type
                                                        relation ;
```

For example:

```
WHICH KEPT HAVE ALIAS FOR ITEM = DAILY FOR GROUP = WEEKLY ;
```

selects those members held in the unnamed KEPT- DATA list that are ITEMs having an alias of DAILY, or are GROUPs having an alias of WEEKLY.

Selecting Members Belonging to a Specified Owner

To select members that belong to a specified owner, enter:

WHICH *selection* BELONG-TO *owner-name* ;

where:

selection can be any of the selections available with the WHICH command. Non member-type selection categories covering aliases, catalog classifications and user-defined indexed attributes are accepted but are not meaningful.

If a member-type selection is specified, all encoded or dummy members that belong to owner-name are selected. If the selection is MEMBERS, KEPT-DATA or INDEX-NAMES then all members (including source-only members) that belong to owner-name are selected.

owner-name must be the name of an owner registered in the repository.

Members with access security levels higher than the user's security level are not listed in the output of the command but are included in the final count of inaccessible members.

Selecting Members by a Usage or Reference Count

To select encoded members that directly use a specified number of members, enter:

```
WHICH selection ARE REFERRING-TO operator n TIMES ;
```

To select index-names that are directly used by a specified number of encoded members, enter:

```
WHICH selection ARE USED-BY operator n TIMES ;
```

where:

selection is any of the selections available with the WHICH command. Non member-type selection categories covering aliases, catalog classifications and user-defined indexed attributes are accepted but are not meaningful in the context of a REFERRING-TO condition.

operator is optional. It can be one of the following:

```
EQUALS or EQ or =                 (equal to)
NE                                (not equal to)
GT or >                           (greater than)
GE                                (greater than or equal to)
LT or <                           (less than)
LE                                (less than or equal to)
```

The default operator is EQUALS.

n is in the range 0 to 2,147,483,647.

The keyword TIMES is optional and has no processing significance.

Selecting Index-Names Kept in a KEPT-DATA List

To find out which of a selection of index-names you have entered in an unnamed KEPT-DATA list, enter:

```
WHICH selection ARE KEPT ;
```

selection is one of the selections available with the WHICH command.

For example:

```
WHICH ITEMS ARE KEPT ;
```

selects ITEM members held in the unnamed KEPT-DATA list.

To find out which of a selection of index-names you have entered in a named KEPT-DATA list, enter:

```
WHICH selection ARE KEPT IN list-name ;
```

where *list-name* is the name of a KEPT-DATA list. Only one KEPT-DATA list name may be specified.

For example:

```
WHICH FILES ARE KEPT IN FLIST ;
```

selects FILE members held in the KEPT-DATA list named FLIST.

KEPT-DATA and KEPT-DATA IN list-name can be a selection as well as a condition.

For example:

```
WHICH KEPT-DATA HAVE ALIAS EQUALS HIST ;
```

selects members held in the unnamed KEPT-DATA list that have an alias named HIST.

Selecting Index-Names That Do or Do Not Meet a Combination of Conditions

To select index-names that meet all of a combination of conditions, enter:

```
WHICH selection condition AND condition ;
```

selection and condition can be any of the selections or conditions available with the WHICH command. Conditions are evaluated from left to right.

To select index-names that meet one or more of a combination of conditions, enter:

```
WHICH selection condition OR condition ;
```

A condition can be preceded by the keyword NOT, in which case index-names that do not meet that condition are selected. The keyword DOES (or DO) can be included in the command for readability, but has no effect.

For example:

```
WHICH MEMBERS DO NOT FORM MONTHLY ;
```

selects members that are not cataloged under the classification MONTHLY.

AND and OR conditions can be combined.

For example:

```
WHICH MEMBERS FORM MONTHLY AND USE ADMIN OR FORM WEEKLY AND USE PERSONNEL ;
```

selects out of the members that have both a classification of MONTHLY and use ADMIN, and out of the members with a classification of WEEKLY, only those that use PERSONNEL.

Parentheses can be used to establish an alternative sequence of evaluation. The number of open parentheses [(must equal the number of close parentheses)]. There is no restriction on the depth of nesting of parentheses.

For example:

```
WHICH MEMBERS FORM MONTHLY AND USE ADMIN OR (FORM WEEKLY AND USE
                                            PERSONNEL);
```

selects the members catalogued under the classification MONTHLY that use ADMIN and those with a classification of WEEKLY that use PERSONNEL.

Name, Status, Time and User-Related-Selections

Refer to "Applying Commands to a Selection Of Index-names Based on Name, Status, Time and User" on page 293 for details of the name, status, time and user-related selections.

WHICH Examples

To select MODULEs that directly or indirectly constitute MOD-PRINT-ERROR-RPT via any relationship type or relationship-type class, enter:

```
WHICH MODULES INDIRECTLY CONSTITUTE MOD-PRINT-ERROR-RPT ;
```

To select FILEs that directly constitute DMC28 via a relationship type or relationship class PRINTS, enter:

```
WHICH FILES DIRECTLY CONSTITUTE DMC28 VIA PRINTS ;
```

The above example assumes you have defined an appropriate relationship type or relationship-type class PRINTS. If PRINTS is a relationship type then it could be an ER relationship type or an EA relationship type [(that is, a renamed user-defined relationship (UDR)].

To select PROGRAMs and MODULEs that directly use

FILE-EMP-MASTER via an ER relationship type or relationship class INPUTS, enter:

```
WHICH PROGRAMS, MODULES DIRECTLY USE FILE-EMP-MASTER RELATED VIA INPUTS ;
```

To select members with the attribute RETENTION- PERIOD, enter:

```
WHICH MEMBERS HAVE RETENTION-PERIOD SPECIFIED ;
```

To select GROUPs in a KEPT-DATA list that are cataloged as EMPLOYEE, enter:

```
WHICH KEPT-DATA GROUPS HAVE CATALOG EQ EMPLOYEE ;
```

To select ITEMs in a KEPT-DATA list that have the word ALLOWANCES in the second line of their DESCRIPTION clause, enter:

```
WHICH KEPT-DATA HAVE DESCRIPTION FOR ITEMS LINE 2 INCLUDING 'ALLOWANCES' ;
```

To select FILEs added to the repository before 18th February 1990 owned by PERSONNEL, enter:

```
WHICH FILES IF INTRODUCED BEFORE '18 FEB 90' BELONG-TO PERSONNEL ;
```

To select members owned by PERSONNEL, enter:

```
WHICH MEMBERS BELONG-TO PERSONNEL ;
```

To select all members, except ITEMs and GROUPs, that are:

- Directly used by SYS-EMP-HIST via any relationship type
- Owned by PERSONNEL

enter:

```
WHICH EXCEPT ITEMS, GROUPS DIRECTLY CONSTITUTE SYS-EMP-HIST AND BELONG-TO
                                                                PERSONNEL ;
```

To select members that are catalogued under both the classifications BATCH and WEEKLY, enter:

```
WHICH MEMBERS HAVE CATALOG = BATCH, WEEKLY ;
```

(This is the same as the command WHAT FORMS BATCH, WEEKLY.)

WHICH Syntax

```
>>--WHICH --+-------------------------------+-- selection -->
            |-status-related-selection -|
            |-LOCKED -------------------|
            |-CHECK-OK -----------------|
            |-CHECK-NEEDED -------------|

>--+-------------------------+--+-----------------------+-->
   |-ALPHABETICALLY -|  |-name-related-selection -|

>--+-------------------------------------+-- clause -->
   |-time-and-user-related-selection -|

>--+-- ; --+--><
   |-- . --|
```

where *selection* is:

```
>--+-MEMBERS ------------------------------------------------+-->
   |-KEPT-DATA --+-------------------+--+-MEMBERS ----------+|
   |             |-IN list-name -|  |-INDEX-NAMES -------+|
   |             |                  |                    <<< , <<  |
   |             |                  |                    category  |
   |             |                  |-------------+-------+|
   |             |                  |             |-EXCEPT -|
   |                                <<< , <<
   |-------------+-------------+-category ----------------------+|
   |             |-EXCEPT -|
   |-INDEX-NAMES ------------------------------------------------+|
                 |-EXCEPT <<< category <<  -|
```

where:

list-name is the name of a KEPT-DATA list

category is:

```
>--+-ALIAS ---+-------------+-------------------------+-->
   |           |-alias-type -|                         |
   |           |-number -----|                         |
   |-alias-type ALIAS ---------------------------------|
   |-CATALOGS ----------------------------------------|
   |-CATALOGUES ---------------------------------------|
   |-ATTRIBUTES ---------------------------------------|
   |-------------+-------------+-member-type ----------|
                 |-GENERIC -|
```

where:

alias-type is one of the specific alias-type keywords available in your repository

number is an integer from 1 to the maximum number of general aliases allowed in your repository.

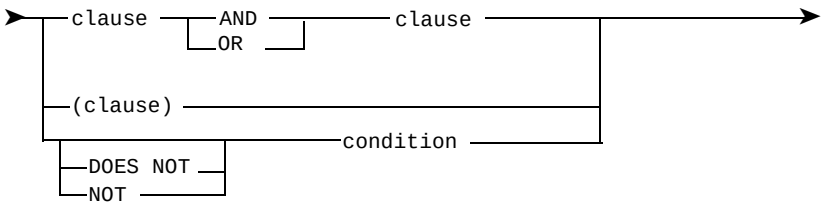
Use the SHOW ALIAS-TYPES command to find out the specific alias-types and the number of general aliases available in your repository.

member-type is the interrogate keyword for any member type/collective member type in your repository.

Use the SHOW UDS command to list the member-type interrogate keywords available in your repository.

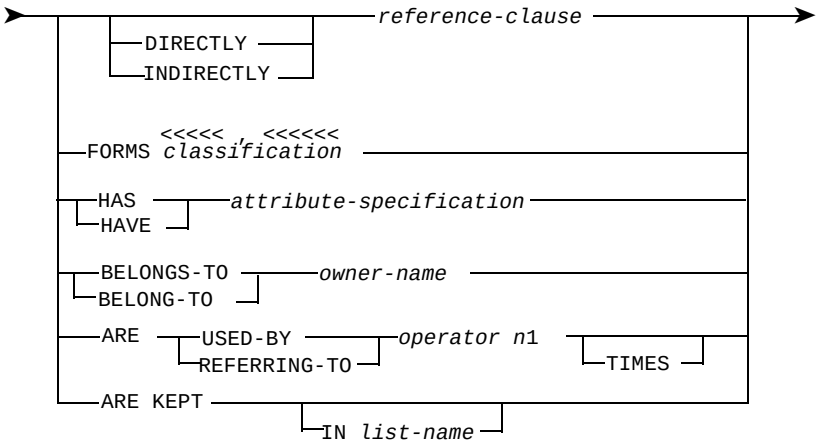
status, *name*, *time* and *user-related-selection* are defined in "Applying Commands to a Selection Of Index-names Based on Name, Status, Time and User" on page 293

clause is:



where:

condition is:



references-clause is:

```
>>-+-+-USES        -+-+-name-+-+-form-+-+-rel-type-+-+-+
```

classification is a catalog classification

attribute-specification is:

```
                    <<<
```

operator is:

```
➤──┬─EQUALS──────────────────────────────────────────────────────➤
   ├─=──────┤
   ├─EQ─────┤
   ├─NE─────┤
   ├─GT─────┤
   ├─>──────┤
   ├─GE─────┤
   ├─LT─────┤
   ├─<──────┤
   └─LE─────┘
```

string is as defined above.

value-attr is:

```
➤──┬─ALIAS──┬─────────────┬──┬──────────────────────────────────────➤
   │        ├─alias-type──┤  │
   │        └─number──────┘  │
   ├─alias-type ALIAS────────┤
   ├─CATALOGUE───────────────┤
   └─attribute───────────────┘
```

where:

alias-type is as defined above.

number is as defined above.

attribute is any attribute in a member definition that contains neither text nor a string (see output of the SHOW MEMBER-TYPE command).

v-relation is:

```
➤──operator <value, <──────────────────────────────────────────────➤
```

where:

operator is defined above.

value is any variable appropriate to the value-attribute.

owner-name is the name of an owner.

operator is as defined above.

n1 is an unsigned integer in the range 0 to 2,147,483,647.

list-name is as defined above.

Note:
Classification, *owner-name* and *string* may optionally be in quotes.

WHO (CMR-DD3)

The WHO command ascertains the ownership of specified members of the dictionary.

To find out if a member has an owner, enter:

```
WHO OWNS member-name-list ;
```

where *member-name-list* is a list of one or more member-names separated by commas. Because ownership information is obtained from the Index dataset, *member-name-list* may include both encoded and unencoded members.

For example, to find out if the member EMP-SAL has an owner, enter:

```
WHO OWNS EMP-SAL ;
```

If a member is owned, the following reply is given:

```
member-name IS OWNED BY owner-name
```

where *owner-name* is the name of the person or organization assigned ownership of the member via a PROTECT command.

If a member is not owned, the following reply is given:

```
NO SPECIFIC OWNERSHIP FOR member-name
```

WHO Syntax

```
                <<<< , <<<<
>>---WHO OWNX member-name ----------+---;---+---------------------><
                                    |   .   |
```

where *member-name* is the name of a dictionary member.

WHOSE (DMR-DD1)

The WHOSE command ascertains the encoded member(s) to which an alias relates.

To list encoded members that have a given alias, enter:

```
WHOSE ALIAS IS alias ;
```

All members which have alias in their encoded record are listed.

The command can be refined to list members that have a specific type of alias or a general alias that appears in a given sequence among the other general aliases in their encoded record.

To list the members that have an alias of a specific type, enter:

```
WHOSE alias-type ALIAS IS alias ;
```

where *alias-type* is one of the specific alias-type keywords available in your dictionary.

For example, to output the names of members that have DATE-UPD-DEPT as their COBOL alias, enter:

```
WHOSE COBOL ALIAS IS DATE-UPD-DEPT ;
```

To list members with an alias as their *nth* general alias, enter:

```
WHOSE ALIAS number IS alias ;
```

where *number* is an integer from one to the number of general aliases allowed in a member definition.

For example, to output the names of members that have TIME-UPD-DEPT as the fourth general alias in their encoded record, enter:

```
WHOSE ALIAS 4 IS TIME-UPD-DEPT ;
```

Use the SHOW ALIAS-TYPES command to output the specific alias types and the number of general aliases available in a member definition.

If the Audit and Security facility is installed (selectable unit CMR-DD3), only those members with an access security level equal to or lower than your (general or specific) security level are listed.

If a member has a higher access security level, one of these messages displays:

```
alias IS THE ALIAS number OF n INACCESSIBLE MEMBERS
```

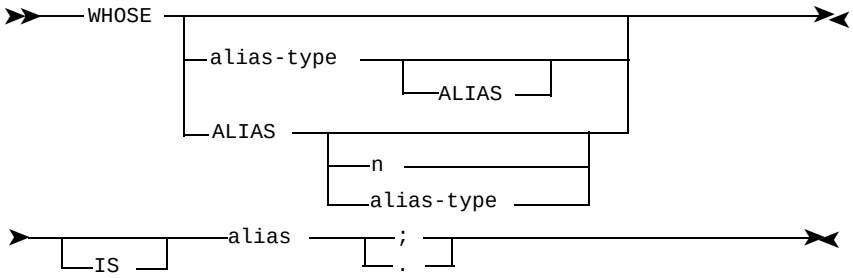
or

```
alias IS THE alias-type ALIAS OF n INACCESSIBLE MEMBERS
```

Refer to *ASG-Manager Products Status Concepts* for details of using the WHOSE command in a multi-status dictionary.

Refer to "Security Levels" on page 18 for details of security levels.

WHOSE Syntax



where:

alias-type is one of the specific alias-type keywords available in your dictionary

n can be any number from 1 to the maximum number of general aliases allowed in your dictionary

alias is a string of from 1 to 79 printable characters, being an alias recorded in the dictionary.

WRITE

Refer to "READ and WRITE (CMR-FE01, CMR-TP7, CMR-TP8)" on page 218 for details of WRITE.

Applying Commands to a Selection Of Index-names Based on Name, Status, Time and User

Name, status, time and user-related-selection provide keywords and clauses that are available for use in a number of different commands.

They give further selection capabilities, additional to the primary selection capabilities defined in the individual command syntax, that enable the user to confine the operation of the command to those index-names that satisfy stated conditions.

These selection capabilities fall into these groups:

- Selection based on name. These clauses restrict the operation of the command to those index-names that appear within a defined part of the index data set, or whose names contain a stated character string or are of a specified length
- Selection based on status. These keywords restrict the operation of the command to those index-names that satisfy a condition related to the status in which the user is working
- Selection based on time and user. These clauses restrict the operation of the command to those index-names that have been processed in particular ways within the selected time periods and/or by stated users and/or in any selected status.

Some categories of index-names are not relevant for all the commands in which these selection capabilities can be used; thus in this context:

- Catalog classifications and indexed attributes are relevant only for LIST, PERFORM and WHICH CONSTITUTE commands, and for Controller's analysis commands
- Aliases are relevant only for GLOSSARY, LIST, PERFORM and WHICH CONSTITUTE commands, and for Controller's analysis commands
- Member names are relevant for all commands in which these further selection capabilities can be specified.

Any of these categories of index-names can be excluded by the primary selection keywords or clauses present in the particular command.

Refer to "Definition Of Index-names" on page 316 for a description of index-names.

This section describes the secondary selections available with a number of different commands. The selections are documented in alphabetic order of selection name.

NAME-RELATED-SELECTION

NAME-RELATED-SELECTION applies a command to one or more index-names, selected by characters in their name.

Refer to "Name-Related Selection Syntax" on page 298 for the syntax of name-related-selection.

Selecting Index-Names Beginning with a Specific Character String

To select index-names whose names begin with a specified string, enter:

```
command ONLY string ;
```

where *command* can be one of the following:

BULK ENCODEBULK REPORTLISTWHICH
BULK PRINTGLOSSARYPERFORM

and certain Controller's commands.

For example:

```
LIST ONLY MOD ;
```

lists all members whose member name begins with MOD.

Selecting Index-Names Ending in a Specific Character String

To select index-names whose names end with a given character, enter:

```
command WHEN END EQ character ;
```

EQ stands for equals and can also be represented by =

To select index-names whose names end with a given string of characters, enter:

```
command WHEN END -n TO END EQ string ;
```

where:

n is a digit specifying character position in the index-name.

END is the last character position, END-1 is the second character from the end,

END-2 the third from the end, and so on.

string must not be longer than the specified number of characters from the end.

For example:

```
BULK PRINT ITEMS WHEN END -3 TO END EQ CODE ;
```

prints the ITEMs whose names end in CODE.

To select index-names with a string in a stated position from the end of the name, enter:

```
command WHEN END -n TO END -m EQ string ;
```

where *n* and *m* are digits specifying character positions in the index-name.

For example:

```
LIST WHEN END -5 TO END -3 EQ EMP ;
```

lists all the members with the string EMP beginning six characters from the end of the name.

If END -*n* TO END -*m* is specified, the string must not be longer than the difference between the two positions plus 1. If it is shorter, the condition is satisfied if the string appears anywhere in the stated range.

Whether a string or a character need to be delimited is determined by the character set from which they are formed. In this respect the string and character must comply with the naming rules governing dictionary member names.

To select index-names that do not meet a name condition, substitute NE (not equal) for EQ in the formats given above.

Selecting Index-Names Containing a Specific Character String

To select index-names whose names contain a specified string, enter:

command WHEN ANY EQ *string* ;

To select index-names whose names contain a string in stated positions, enter:

command WHEN *n* TO *m* EQ *string* ;

n and *m* are digits specifying character positions in the index-name.

For example:

```
LIST ALIAS WHEN 5 TO 6 EQ MR ;
```

selects all aliases whose names contain MR in the fifth and sixth character positions.

If you omit TO *m*, the string must be a single character.

To select index-names that do not meet a name condition, substitute NE (not equal) for EQ in the formats given above.

Selecting Index-Names of a Specific Length

To select index-names whose names are of a specific length, enter:

```
command WHEN LENGTH EQ n ;
```

where *n* is a digit specifying the number of characters the index-name contains.

For example:

```
LIST WHEN LENGTH EQ 5 ;
```

will list all member names of five characters.

In this case EQ, = or NE can alternatively be:

GT or > (greater than)
LT or < (less than)
GE (greater than or equal to)
LE (less than or equal to)

For example:

```
LIST INDEX-NAMES WHEN LENGTH LE 5 ;
```

lists index-names of one, two, three, four and five characters.

Selecting Index-Names in a Specified Range

To select a range of index-names that occur in alphanumeric order between two specified character strings, enter;

```
command FROM string-1 TO string-2 ;
```

where *string-1* and *string-2* are strings of up to 79 characters and string-2 has a greater alphanumeric value than string-1.

For example:

```
LIST INDEX-NAMES FROM DEPT TO EMP ;
```

lists all index-names that occur in alphanumeric order between DEPT and EMP, starting with the first name that begins with DEPT and ending with the last name that begins with EMP.

If the strings are themselves index-names, those index-names are also selected.

For example:

```
BULK ENCODE FROM PROG-RPT TO PROG-UPD-HIST ;
```

encodes PROG-RPT and all the member names that alphanumerically follow PROG-RPT, up to and including PROG-UPD-HIST;

If you omit FROM *string-1*, all index-names with names up to, and including, *string-2* in the collating sequence are selected.

If you omit TO *string-2*, all index-names with names from, and including, *string-1* in the collating sequence are selected.

Combining Several Name Selections

Two or more WHEN conditions can be connected by the AND or OR keyword. The conditions are evaluated from left to right.

If AND is specified, index-names must satisfy both the selection criteria.

For example:

```
BULK REPORT WHEN ANY EQ NO AND END -3 TO END EQ FILE ;
```

reports members whose name contains the string NO and ends in FILE.

If OR is specified, index-names can satisfy either of the selection criteria.

For example:

```
LIST CATALOG WHEN ANY = EMP OR END = T ;
```

lists catalogs whose classifications contain the string EMP or end with the character T.

Examples of Name-Related-Selections

To encode all members with a name ending in the character string CODE, enter:

```
BULK ENCODE WHEN END -3 TO END EQ CODE
```

To list those index-names beginning with EMP that were inserted in the dictionary by a user called PERSONNEL on 20 February 1990, enter:

```
LIST INDEX-NAMES ONLY EMP IF INTRODUCED BY PERSONNEL ON '20 FEB 1990' ;
```

Name-Related Selection Syntax

```
┌──────────────────────────────────────────────────────────────────────────────┐
│  FROM a ──────────────────────────────────────────────────────────────────   │
│            └─TO b ─┘                                                         │
│  TO b ────────────────────────────────────────────────────────────────────   │
│  ONLY a ──────────────────────────────────────────────────────────────────   │
└──────────────────────────────────────────────────────────────────────────────┘
┌──────────────────────────────────────────────────────────────────────────────┐
│  WHEN condition ──┬──────────────────────────────────────────────────────┐   │
│                   │  <<<<<<<<<< , <<<<<<<<<<                             │   │
│                   └─AND ──────┬─condition ─┘                              │   │
│                     ├─OR ─────┤                                           │   │
│                     └─OR WHEN ─┘                                          │   │
└──────────────────────────────────────────────────────────────────────────────┘
```

where:

a and *b* are character strings of not more than 79 characters, representing characters of entries in the Index data set.

If *a* and *b* are both present, the alphanumeric value of *b* must be equal to or greater than the alphanumeric value of *a*.

If *a* or *b* contain characters from the extended character set the string must be delimited.

condition is:

```
┌──────────────────────────────────────────────────────────────────────────────┐
│  ┌─ANY ──────────────────┬─┬─EQ ──┬──────────────────────────────────────┐   │
│  │                       │ │ ├─=  ┤                                      │   │
│  ├─P ──┬──────────────┬───┤ │ └─NE ┘                                      │   │
│  │      └─TO q ────┘    │   │   string                                    │   │
│  └─name-end ────────────┘   │                                             │   │
│                             │                                             │   │
│  └─LENGTH ──┬─EQ ──┬──────────────────────────────────────────────────────┘   │
│             ├─=  ──┤  integer                                                  │
│             ├─NE ──┤                                                           │
│             ├─GT ──┤                                                           │
│             ├─>  ──┤                                                           │
│             ├─GE ──┤                                                           │
│             ├─LT ──┤                                                           │
│             ├─<  ──┤                                                           │
│             └─LE ──┘                                                           │
└──────────────────────────────────────────────────────────────────────────────┘
```

where:

p and *q* are integers. The alphanumeric value of p must be less than the alphanumeric value of q.

string is a character string of between 1 and 32 printable characters

name-end is:

```
➤──── END ──┬──────────────────────────────────────────────┬──➤
            │   ┌──────────────────────────────────────────┐
            └─-q┬──────────────────────────────────────────┤
                │   ┌──────────────────────────────────────┐
                └───TO END ──┬──────────────────────────┬───┘
                             │   ┌──────────────────────┐
                             └───-p──────────────────────┘
```

where:

p and *q* are as defined above

EQ or= means equal to
NE means not equal to
GT or > means greater than
GE means greater than or equal to
LT or < means less than
LE means less than or equal to.

integer is an unsigned integer in the range of 1 to 79, specifying a number of characters.

Status-Related-Selection

Status-Related-Selection applies a command to one or more index-names, selected by comparison with other statuses.

The selection is described in the following numbered items:

Refer to "STATUS-RELATED-SELECTION Syntax" on page 308 for the syntax of status-related-selection.

Introduction to Status-Related-Selections

Status-related-selection selects index-names that satisfy conditions when they are compared in statuses other than the current status.

To use status-related-selection, enter:

command status-related-selection selection ;

where:

command is one of these commands:

BULK ENCODE	BULK REPORT	LIST	WHICH
BULK PRINT	GLOSSARY	PERFORM	

and certain Controller's commands.

`status-related-selection` is one or more of these keywords:

CURRENT
CHANGED
DIVERGING
REVERIFIED
NEW AMENDED
UNVERIFIED

Several keywords can be specified in one command if they are separated by commas.

`selection` is optional and may be any other selection criteria valid for use with the specified command.

Index-names that meet one or more of the specified status-related-selection criteria are selected, provided that they also meet any other selection criteria specified in the command.

Selecting Index-names that Exist in the Current Status

The CURRENT keyword selects index-names that have records which physically exist (as encoded records, source records, dummy records, or usage-tables) in the current status.

Index-names that are visible from the current status, but do not physically exist in it, are not selected.

For example:

```
LIST CURRENT ITEMS ;
```

lists all ITEM members with a record in the current status.

Three additional keywords are available which enable you to refine the selection made by CURRENT. These are DEFINITIONS, SOURCE-RECORDS, and USAGE-TABLES

CURRENT DEFINITIONS will only select members that have an encoded record in the current status.

CURRENT SOURCE-RECORDS will only select members that have a source record (encoded or unencoded) in the current status.

CURRENT USAGE-TABLES will only select index-names which exist in the current status as either:

- Usage-tables only
- Usage-tables and unencoded source records
- Dummy records only
- Dummy records and unencoded source records.

Selecting Index-Names that are New in the Current Status

The NEW keyword selects index-names that have records which physically exist (as encoded records, source records, dummy records or usage-tables) in the current status, but not in any of its direct and indirect base statuses.

For example:

```
LIST NEW ITEMS ;
```

lists all ITEM members that physically exist in the current status, but not in any of its base statuses.

Selecting Index-Names Changed in the Current Status

The CHANGED keyword enables you to select index-names whose condition in the current status has changed when compared with the nearest direct or indirect base status, in which it exists.

If you specify the additional keyword TYPES, then CHANGED will only select members whose member type has changed in the current status when compared with the nearest direct or indirect base status in which they exist. If the member does not exist in a direct or indirect base status it will not be selected.

For example:

```
LIST CHANGED TYPES ITEMS ;
```

selects ITEM member types in the current status that are a different member type in the nearest direct or indirect base status in which they exist.

These are examples of index-names selected by CHANGED:

- Members with a source record in both the current status and in a direct and/or indirect base of the current status
- Members which directly use another member whose member type has changed in the current status when compared with the nearest direct or indirect base status in which it exists
- Members directly used by another member whose member-type has changed in the current status when compared with the nearest direct or indirect base status in which it exists
- Members which have an unencoded source record in the current status and a dummy record in one of its direct and/or indirect base statuses
- Members directly used by a different set of members in the current status when compared with the nearest direct or indirect base status in which they exist
- Members directly used by the same set of members in the current status when compared with the nearest direct or indirect base status in which they exist, with the exception that one of those references is from a different clause in the referring member's data definition statement
- Any alias which is the alias of a different set of members in the current status when compared to the nearest direct or indirect base status in which it exists
- Any alias whose alias-type or number differs in the current status when compared with the nearest direct or indirect base status in which it exists
- Catalog classifications or attributes which relate to a different set of members in the current status when compared with the nearest direct or indirect base status in which they exist.

Selecting Members with Records in the Current and a Base Status

The AMENDED keyword enables you to select members that have a source record (encoded or unencoded) in the current status and in at least one of its direct and indirect base statuses.

The REVERIFIED keyword enables you to select members that have an encoded record in the current status and in at least one of its direct and indirect base statuses. in two visible non-base statuses.

Selecting Index-Names that Exist in Two Visible Non-Base Statuses

The DIVERGING keyword enables you to select index-names that satisfy all of the following conditions:

- They are visible from the current status
- They exist in at least two non-base statuses (one of which may be the current status).

This enables you to compare entity/data definitions across departments/projects.

Three additional keywords are available which enable you to refine the selection made by DIVERGING. These are TYPES, DEFINITIONS, and SOURCE-RECORDS

DIVERGING TYPES selects index-names that have diverging member types visible from the current status, where the member type is different in at least two statuses, which are not the direct or indirect base or dependent status of the other.

DIVERGING DEFINITIONS selects members with diverging encoded records.

DIVERGING SOURCE-RECORDS selects members with diverging source records.

Examples of Using the DIVERGING Keyword

For the following examples assume that:

- DEV1 is the current status
- Mem-1 is an ITEM member type in all statuses indicated and does not exist in any other status.

Example 1

The statuses in the dictionary are organized as represented by the following diagram:

```
D                  +===STATUS===+                    LEVEL 1
I                      'LIVE'                        root
C                       mem-1                        status
T                  +=====+======+
I
O       +                                 +
N +===STATUS===+   +===STATUS===+   +===STATUS===+   LEVEL 2
A     'TEST'           'DEV2'           'DEV1'       dependent
R                                        mem-1       statuses
Y +============+   +============+   +============+
                                      CURRENT STATUS
```

Command. LIST DIVERGING ITEMS ;

Response. Mem-1 is not selected. Although it is an ITEM member type and it exists in two statuses, it is not present in two non-base statuses.

Example 2

```
D                   +===STATUS===+                     LEVEL 1
I                       'LIVE'                         root
C                                                      status
T                   +=====+======+
I
O       +                                  +
N +===STATUS===+   +===STATUS===+   +===STATUS===+    LEVEL 2
A     'TEST'           'DEV2'           'DEV1'        dependent
R                       mem-1            mem-1         statuses
Y +============+   +============+   +============+
                                     CURRENT STATUS
```

Command. LIST DIVERGING ITEMS ;

Response. Mem-1 is selected. It is visible from the current status as an ITEM member type and it exists in at least two non-base statuses (one of which happens to be the current status) which are not the direct or indirect base or dependent status of the other.

The above example represents a typical situation in which two development teams record different definitions of the same member based upon its definition in the base status (which reflects its condition in the current system). Thus the dictionary contains diverging definitions of that member.

Example 3

```
D                   +===STATUS===+                     LEVEL 1
I                       'LIVE'                         root
C                        mem-1                         status
T                   +=====+======+
I
O       +                                  +
N +===STATUS===+   +===STATUS===+   +===STATUS===+    LEVEL 2
A     'TEST'           'DEV2'           'DEV1'        dependent
R     mem-1                              mem-1         statuses
Y +============+   +============+   +============+
                                     CURRENT STATUS
```

Command. LIST DIVERGING SOURCE-RECORDS ;

Response. Mem-1 is not selected. Although it is diverging, it does not have diverging source records. It only has source records in statuses DEV1 and LIVE which have a base/dependent relationship. It also exists in status TEST but only as a usage-table.

Selecting Unencoded Members Visible from the Current Status

The UNVERIFIED keyword selects members visible from the current status that have either:

- An unencoded source record
- A source record that differs from the encoded record, that is, the source record has been altered since the member was last successfully encoded.

Members which are dummy records only (that is, they have no source record) are not selected by UNVERIFIED.

Members whose condition is unverified are identified in output from a LIST command by an asterisk (*) and the entry 'SCE' in the CONDITION column.

When you PRINT or BULK-PRINT members of the dictionary, a message is displayed if they are unverified.

LIST command output indicates dummy members by an asterisk (*) and the entry 'DUM' in the CONDITION column.

Note:

The REINSTATE command enables you to remove unverified source records, thus reinstating a member in the condition in which it was last successfully encoded.

Tabulation: Selection made by Status-Related-Selection Keywords

```
MEMBERS' PRESENCE IN:
BASE            CURRENT       RELEVANT SELECTION KEYWORD
STATUSES         STATUS

Not Present        Dummy                                  C
                                                          U
Not Present        S + E        N                         R
                                E                         R
Not Present        S' + E       W                         E
                                                          N
Not Present        S                                      T

                                                          /
                                                          /
 S' + E      See                                          /
             through            U                         /
   S         See                N                         /
             through            V                         /
                                E
                                R
 Dummy       S                  I
                                F
                                I                 A
 S + E       S                  E                 M       C
                                D                 E       U
 S + E       S' + E                       R       N       R
                                          E       D       R
 S + E       S + E                        V       E       E
                                          E       D       N
 S + E       S' + E'                      R               T
                                          I               C
                                          F               H
                                          I               A
                                          E               N
                                          D               G
                                                          E
                                                          D
 S + E       See through with
             used-by change
 S + E       See through with
             change in
             member type of
             referred to member

 S + E       Removed by
             Controller.
             Dummy because of
             references to
             member in this
             status.

 S + E       See through
             No used-by or
             reference change
 S + E       Removed by
             Controller. No
             references to
             member in this
             status.
```

Note 1. The status-related-selection keyword DIVERGING does not appear in the above tabulation.

Note 2. CURRENT appears twice in this table. The characters used to indicate the continuation of the first occurrence to the second (///) serve no other purpose other than indicating that there is a continuation.

Examples of Status-Related-Selection

To encode all the members that exist in the current status but not in any base status, and were created by a user named PERSONNEL, enter:

```
BULK ENCODE NEW IF INTRODUCED BY PERSONNEL ;
```

To find out those members existing in the current status that directly use the member TAX-CODE, enter:

```
WHICH CURRENT DIRECTLY USE TAX-CODE ;
```

To report all members changed in comparison with the nearest base status, enter:

```
BULK REPORT CHANGED ;
```

To list all the members whose names begin with the character string EMP and have a source record in the current status and a base status, enter:

```
LIST AMENDED ONLY EMP ;
```

To produce a glossary of members with an out of step or unencoded source record, and find out what directly uses them, enter:

```
GLOSSARY UNVERIFIED GIVING DIRECT USAGES ;
```

To print all members that are visible from the current status and exist in two non-base statuses, enter:

```
BULK PRINT DIVERGING ;
```

STATUS-RELATED-SELECTION Syntax

Secondary Selection Clauses: `status-related-selection`

```
     <<<<<<<<<<<<<<<<<<< , <<<<<<<<<<<<<<<<<<
 ►──┬─NEW──────────────────────────────────────────────────►
    ├─CHANGED──┬──────────────────────────┬──────────────────
    │          └──TYPES──┘                │
    │                                     │
    ├─AMENDED─────────────────────────────┤
    ├─REVERIFIED──────────────────────────┤
    ├─DIVERGING──┬──────────────────────┬─┤
    │            ├──DEFINITIONS─────────┤ │
    │            ├──SOURCE-RECORDS──────┤ │
    │            └──TYPES───────────────┘ │
    │                                     │
    ├─UNVERIFIED──────────────────────────┤
    └─CURRENT──┬──────────────────────┬──┘
               ├──DEFINITIONS─────────┤
               ├──SOURCE-RECORDS──────┤
               └──TYPES───────────────┘
```

TIME and USER-RELATED-SELECTION (CMR-CM01)

TIME and USER-RELATED-SELECTION applies a command to one or more index-names, updated at a specific time or by a specific user.

Refer to "TIME and USER-RELATED-SELECTION Syntax" on page 314 for the syntax of time and user-related-selection.

Introduction to Time and User-Related-Selections

Time and user-related-selection allows you to select members INTRODUCED, AMENDED or VERIFIED in the dictionary within a specified period of time, or by a specific user.

You can specify time and user-related-selection in these commands:

BULK ENCODE	BULK REPORT	LIST	WHICH
BULK PRINT	GLOSSARY	PERFORM	

and certain Controller's commands.

The keyword INTRODUCED selects:

- Members whose source was introduced within a period of time or by a specific user
- Aliases, catalog classifications or indexed attributes whose index records were created within a given period of time or by a specific user
- Dummies with no source records created within a given period of time or by a specific user.

The keyword AMENDED selects:

- The most recent update to a member's source or encoded record, for example by an ALTER, SFILE or FILE command. AMENDED does not select members encoded by the ENCODE command.

The keyword VERIFIED selects:

- The most recent encoding of a member, for example by a MODIFY, FILE or ENCODE command.

Making a Selection Based on Date

To select index-names updated on a given date, enter:

```
command IF updated ON date ;
```

where:

updated is one of the keywords INTRODUCED, AMENDED or VERIFIED.

date must be in the format which applies at your installation, for example: '12 DEC 1990' or 12/12/90. If the separator between the elements of the date is a space, period, semi-colon, comma or right bracket, date must be delimited.

For example:

```
LIST ALIASES IF INTRODUCED ON '12 DEC 1990' ;
```

lists the aliases introduced on 12 December 1990.

Instead of ON date you can specify:

- BEFORE date
- AFTER date
- BETWEEN date AND date.

If BEFORE or AFTER is specified, members INTRODUCED, AMENDED or VERIFIED on the given date are not included in the selection.

For example:

```
LIST IF VERIFIED AFTER '1 FEB 1990' ;
```

lists all the members that were encoded from 2 February up to the current date.

If BETWEEN date AND date is specified, members INTRODUCED, AMENDED or VERIFIED on the given dates are included in the selection.

For example:

```
BULK ENCODE IF AMENDED BETWEEN '12 DEC 1989' AND '21 JAN 1990' ;
```

encodes members if they were last amended between 12 December 1989 and 21 January 1990 inclusive.

To select all index-names that were not updated on a given date, enter:

```
command IF NOT updated ON date ;
```

For example:

```
GLOSSARY IF NOT VERIFIED ON '20 APR 1990' GIVING ALIAS ;
```

lists the names and aliases of any members that were not encoded on 20 April 1990.

Aliases, catalog classifications and indexed attributes cannot be amended or verified and are not selected by IF AMENDED or IF VERIFIED but they are selected by IF NOT AMENDED or IF NOT VERIFIED.

Making a Selection Based on Date and Time

To select index-names updated on a given date at a specified time, enter:

command IF updated ON date AT *time* ;

where:

updated is one of the keywords INTRODUCED, AMENDED or VERIFIED.

time must be in the format which applies in your installation, for example '12.15.54'.

If the separator between the elements of the time is a space, period, semi-colon, comma or right bracket, time must be delimited.

For example;

```
WHICH ITEMS IF INTRODUCED ON '19 MAY 1990' AT 15.34.10' DIRECTLY
                                                CONSTITUTE ORDER ;
```

lists the ITEMs directly used by the member ORDER if they were inserted in the dictionary at 15.34.10 on 19 May 1990.

Instead of AT time you can specify:

- BEFORE time
- AFTER time
- BETWEEN time AND time.

If BEFORE or AFTER is specified, members INTRODUCED, AMENDED or VERIFIED at the given time are not included in the selection.

If BETWEEN time AND time is specified, members INTRODUCED, AMENDED or VERIFIED at the given time are included in the selection.

Making a Selection Based On User

To select index-names updated by a named user, enter:

command `IF updated BY` *user-name* `;`

where:

updated is one of the keywords INTRODUCED, AMENDED or VERIFIED.

user-name is the name of a user registered in the dictionary by the dictionary Controller. The keyword MASTER can replace user-name in which case

index-names affected by operations carried out by the dictionary Controller are selected.

For example:

```
BULK REPORT IF VERIFIED BY ACCOUNTS ;
```

reports all members encoded by a user named ACCOUNTS.

You can select all index-names that have not been updated by a named user by entering:

```
command IF NOT updated BY user-name ;
```

For example:

```
LIST IF NOT AMENDED BY SALES ;
```

lists all the members not amended by the user SALES.

Aliases, catalog classifications and indexed attributes cannot be amended or encoded and therefore are not selected by IF AMENDED or IF VERIFIED.

They are selected by IF NOT AMENDED or IF NOT VERIFIED.

Making a Secondary Selection from a Specific Status

To apply time and user related selections to the index-names in a specific status, enter:

```
command IF updated IN STATUS status-name ;
```

where:

updated is one of the keywords INTRODUCED, AMENDED or VERIFIED.

status-name is the name of a specific status. It must be visible from the current status.

The keyword STATUS is optional.

For example:

```
BULK ENCODE IF AMENDED IN STATUS PROD ;
```

and

```
BULK ENCODE IF AMENDED IN PROD ;
```

encodes both amended members in the status PROD.

Combining Time and User-Related-Selections

When AND is specified, index-names must satisfy both the conditions stated.

When OR is specified, index-names can satisfy any one of the conditions stated.

Any number of date, time, user or status conditions can be linked by the keywords AND and OR.

For example:

```
LIST INDEX-NAMES IF INTRODUCED ON '9 JUL 1990' OR ON '18 JUL 1990' OR BY
                                                                PERSONNEL ;
```

lists index-names introduced on 9 July 1990, or on 18 July 1990 or introduced by a user named PERSONNEL.

One of the keywords, IF AMENDED, IF VERIFIED or IF INTRODUCED, must be stated for the first condition of a set, but can be omitted from subsequent conditions. If you do not state a keyword immediately before a condition, then the last keyword stated applies to that condition.

IF AMENDED, VERIFIED or INTRODUCED can be combined using the keywords AND and OR.

For example:

```
BULK ENCODE IF INTRODUCED BY ACCOUNTS AND AMENDED ON ’5 AUG 1990’;
```

encodes all members introduced by a user named ACCOUNTS and subsequently amended on 5 August 1990.

Conditions are evaluated from left to right.

Parentheses can be used to control the order in which conditions are processed. The number of open parentheses [(must equal the number of close parentheses)] and there is no restriction on the depth to which they can be nested.

For example:

```
LIST IF INTRODUCED BY PERSONNEL AND AMENDED ON ’3 AUG 90’ OR VERIFIED ON
                                                                ’10 AUG 90’;
```

lists members inserted in the dictionary by PERSONNEL and most recently amended on the 3 August 1990, and lists members encoded on 10 August 1990.

However:

```
LIST IF INTRODUCED BY PERSONNEL AND (A  MENDED ON ’3 AUG 90’ OR VERIFIED
                                                          ON ’10 AUG 90’ ) ;
```

lists members introduced by PERSONNEL that were amended on 3 August 1990 and members introduced by PERSONNEL that were encoded on 10 August 1990.

Examples of Time and User-Related-Selections

To print all members amended on 10 July or on 10 August 1989, enter:

```
BULK PRINT IF AMENDED ON '10 JUL 1989' OR '10 AUG 1989' ;
```

To encode all the members introduced by a user named ACCOUNTS and all the members amended by the dictionary Controller, enter:

```
BULK ENCODE IF INTRODUCED BY ACCOUNTS AND AMENDED BY MASTER ;
```

To encode all the members introduced by a user called ACCOUNTS on 17 March and amended by the dictionary Controller on 20 March 1990, enter:

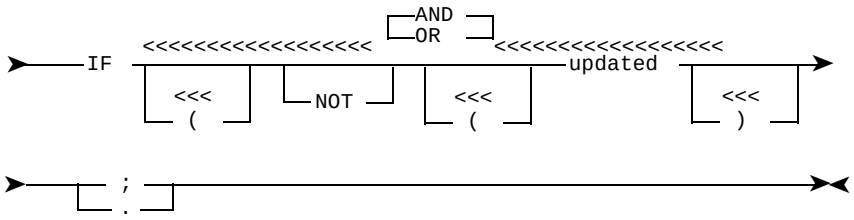
```
BULK ENCODE IF INTRODUCED BY ACCOUNTS ON ’17 MAR 1990’ AND AMENDED BY
                                              MASTER ON  ’20 MAR 1990’ ;
```

To list all members introduced by a user named PERSONNEL in status PROD or all the members encoded on 13 April 1990 in status DEV, enter:

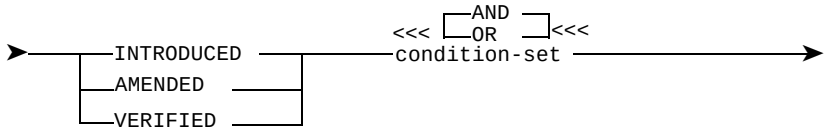
```
LIST IF INTRODUCED BY PERSONNEL IN STATUS PROD OR VERIFIED ON '13 APR
                                                        1990' IN STATUS DEV ;
```

TIME and USER-RELATED-SELECTION Syntax

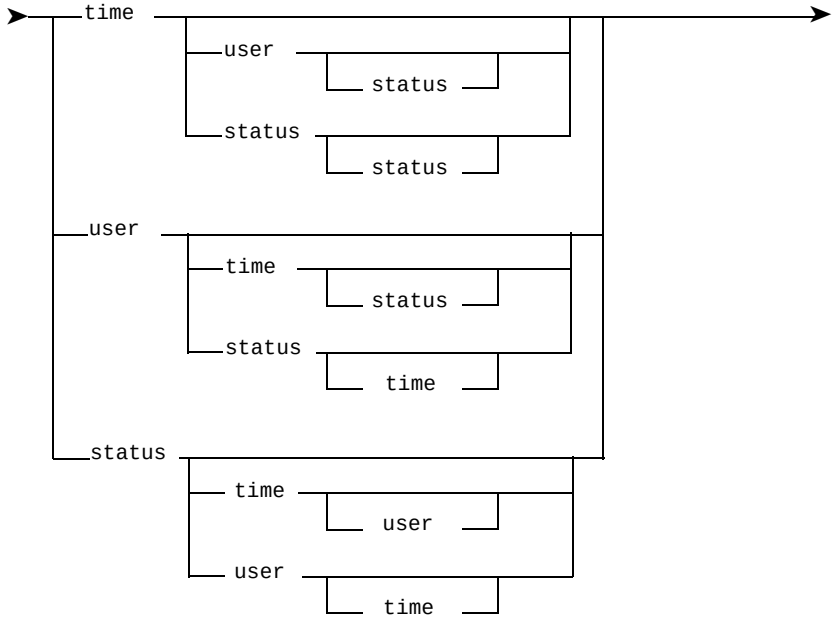
Secondary Selection Clauses: time and user-related-selection



where *updated* is:



where *condition-set* is:



where *time* is:

```
➤──┬─ON d ──┬──┬─┬─AT ──────┬─ t ─┬──┬──────────────────────────────────────────────➤
   │        │  │ │ BEFORE ──┤     │  │
   │        │  │ └─AFTER ───┘     │  │
   │        │  └─BETWEEN t AND t ─┘  │
   │        │                        │
   ├─BEFORE ┬─┬─d ──┬──┬─AT t ──────┬─┤
   │ AFTER ─┘ │     │  └─BETWEEN t and t ─┘
   │          │     │
   │          │     │
   ├─BETWEEN d ──┬──┬─┬─AT t ─┬─AND d ──┬──┬─┬─AT t ─┬─┬──┬──┬──
```

Collective Member Types

A collective member type is an interrogate keyword used within a command to select all members whose member type is one of a pre-defined group of member types.

Use the SHOW UDS command to find out which member types are available in your dictionary.

If you include the collective member type keyword FILES in your interrogation command, details of all members with the member type of conventional files and database files are output.

For example, if your dictionary contains the following member types: FILE, FILE ADABAS and FILE MARK-IV, to output details of all members that have one of these member types, enter:

```
command FILES ;
```

If you want to output details of only those members with a member type of FILE, enter:

```
command CONVENTIONAL-FILES ;
```

where *command* is any command in which this selection is available.

The collective member type FILES includes both conventional files and any member types defining database files that are available in your dictionary.

The member type CONVENTIONAL-FILES includes only those members whose member type is FILE, that is, members that are not defined as being a database file.

If you include the collective member type keyword DATABASES in your interrogation command, details of all members defining a database system are output, such as DB2-DATABASE, TOTAL DATABASE or IMS-DATABASE HSAM SIMPLE.

Definition Of Index-names

Index-names are all valid member names and those aliases, catalog classifications and (if the User Defined Syntax facility, selectable unit CMR-UD1, is installed) user-defined indexed attributes specified in a member's encoded record.

5 Dictionary Contents - Member Definition

This chapter includes these sections:

Manager Products Member Types 317

Member Types Provided By Manager Products 318
Member Types which Document Data and the Processes that Act Upon it . . . 320

Member Definition Statements 322
ITEM Member Definition Statement 322
GROUP Member Type 351
File Member Type. 357
FILE Member Definition 368
SYSTEM, PROGRAM and MODULE Member Definition 380
COMMAND-STREAM Member Definition 387
OBSOLETE-DEFINITION Member Type. 389

Common Clauses. 395
Common Clause Definitions. 395

Manager Products Member Types

Information is held on the Corporate Dictionary/Repository in members of particular member types. Manager Products provide various predefined member types for various purposes. The types of relationships that may be established between members of different member types are predefined to suit those purposes.

For example the ITEM member type is provided for the documentation of the lowest levels of data used in an organization: the smallest named unit into which data is divided. The GROUP member type is provided to document combinations of data elements. A relationship may be established in which a GROUP contains several ITEMs. It is not possible for an ITEM to contain a GROUP or GROUPs as that would be inconsistent with the purpose for which each member type is provided.

You can also define your own member types with their own names and types of relationships to one another. For example you could define a new member type called RECORD modeled upon the ITEM member type. In order to do so you must have the User Defined Syntax facility installed.

You can use all of the predefined Manager Products member types except for those provided by DataManager to document Database Management Systems and the MARK IV File Management System. For these you must have the relevant selectable units installed.

Although most member types can be entered in your dictionary many of them are only relevant to particular purposes provided by specific Manager Products. This enables users who are intending to purchase additional Manager Products to populate their dictionary in advance of acquiring the additional selectable units.

Refer to the *ASG-Manager Products User Defined Syntax* manual for details of how to define your own member types.

Member Types Provided By Manager Products

Manager Products provide predefined member types for various purposes.

To document data and the processes that act upon it use these member types:

ITEM	SYSTEM
GROUP	The Database Management
FILE	System Interfaces and MARKIV
MODULE	Interface member types
PROGRAM	

Refer to "Member Types which Document Data and the Processes that Act Upon it" on page 320 and "Member Definition Statements" on page 322 for details of the above member types. An overview of the Database Management System Interfaces and MARKIV Interface member types is given in sections "Member Types which Document Data and the Processes that Act Upon it" on page 320, "Interfaces to Database Management Systems" on page 404 and "Source Language Generation" on page 406; for full details refer to the relevant DataManager Interface facility manual.

To define any information that is obsolete use the following member type:

OBSOLETE-DEFINITION

Refer to "Member Definition Statements" on page 322 for details of this member type.

To store streams of Manager Product commands use the following member type:

COMMAND-STREAM

Refer to "Member Definition Statements" on page 322 for details of this member type.

To define your own member types and attributes use the following member types:

HIERARCHY	ATTRIBUTE-TYPE
MEMBER-TYPE	ATTRIBUTE-GROUP
MEMBER-TYPE-GROUP	

Refer to *ASG-Manager Products User Defined Syntax* for details of the above member types.

To define a sequence of Manager Product instructions and commands use this member type:

EXECUTIVE-ROUTINE

Refer to *ASG-Manager Products User Defined Syntax* for details of this member type.

To establish the credentials of users logging on to Manager Products use these member types:

GLOBAL-PROFILE
LOGON-PROFILE

Refer to the *ASG-Manager Products Systems Administrator's Guide* for details of the member types.

To define enterprise and application models for subsequent logical design, use these member types:

ITEMVIEWSET
GROUPENTITY
USERVIEWFORMAT

Refer to *ASG-DesignManager User's Guide*, *ASG-DesignManager Enterprise Modeling*, or *ASG-DesignManager User Formatted Output* for details of these member types.

To define the rules by which Manager Products member types are translated into the format of equivalent member types in other Manager Products dictionary/repositories or in other vendors dictionary, repository or directory systems, use this member type:

TRANSLATION-RULE

Refer to *ASG-DictionaryManager User's Guide* for details of the above member type.

To generate COBOL programmes use this member type:

REUSABLE-CODE

To adapt to the supplied online documentation or create your own use this member type:

INFOBANK-PANEL

Refer to *ASG-Manager Products User Defined InfoSystem* for details of the above member type.

To tailor the layout of output from the REPORT and BULK REPORT commands except when those commands are applied to the Workbench Design Area, use this member type:

FORMAT

Refer to *ASG-DataManager User Defined Output* for details of this member type.

Refer to the Terminal User's Quick Reference for the syntax of these member types. For the syntax of REUSABLE-CODE and the Database Management System Interfaces and MARKIV Interface member types refer to the relevant manual specified above.

Member Types which Document Data and the Processes that Act Upon it

A hierarchy of member types is provided with which all elements and assemblages of data and the processes that act on the data can be described.

The number of member types in this basic hierarchy has been kept as small as possible while meeting these requirements. These are the basic member types:

ITEM	MODULE
GROUP	PROGRAM
FILE	SYSTEM
Database	

ITEM. A fundamental element of data, the smallest named unit into which data is divided in the particular user organization. An ITEM is at the lowest level of the member type hierarchy.

GROUP. A combination of items and/or other groups. Groups can be nested (that is, contained in other groups) to any depth. Thus combinations of data items that in other terminologies may be called, for example, group items, subrecords or records, are all called groups.

FILE. A file or dataset, and can be a file of the traditional type or the equivalent association of data in a database. The physical and organizational characteristics of the file can be defined, as well as the records that constitute it. Its records are defined (as part of the FILE definition) as a combination of groups and/or items. A file can also be defined as containing other files: thus, sub-file structures can be defined.

Database. Included in the member type hierarchy only if a database interface facility is included in the particular installation of DataManager. Database interfaces are optional additional facilities; one or more than one such facility can optionally be included in any installation.

The member type is a Database Management System specific database definition; for example, TOTAL-DATABASE or IMS-DATABASE or ADABAS-DATABASE. The database definition includes, among the declarations of other characteristics required to define the particular type of database, a declaration of the data dictionary members that directly constitute the database. The members concerned may be, in Manager Products terminology, files (for instance, to declare TOTAL datasets) or groups and items (for instance, to declare the components of a SYSTEMS 2000 database), depending on the type of database being defined; but some Database Management Systems require further DataManager member types to define the constituents of the databases (for instance, a SEGMENT member type for IMS).

Where the database definition declares files, the definition of each such file includes a declaration to the effect that it is a database file of a particular type, for example, an ADABAS file or a TOTAL MASTER file or a TOTAL VARIABLE file. Thus, if two or more database interfaces are included in an installation of DataManager, databases of different types cannot in their definitions refer to the same file; though they could indirectly (via their respective constituent files) refer to the same group and item definitions.

According to the type of database, database members come between MODULE and FILE members or between MODULE and GROUP members in the hierarchical structure. Some types of database members can cross-refer to one another (within the same database type).

MODULE. A piece of program that directly or indirectly (that is, via other modules that it calls) acts on data. Its definition includes a declaration of the databases, files, groups and/or items used or created by the module, and/or a declaration of other modules called or contained by it. Chains of calls from module to module can extend to any depth.

PROGRAM. Defined in terms of the modules and/or other programs it contains or calls, and/or in terms of the files, groups and/or items that it inputs, outputs or updates, or the database that it accesses. Programs can be nested (that is, can be sub-programs to other programs) to any depth.

SYSTEM. A combination of related programs and/or modules and/or other systems. Its definition includes a declaration of the databases, files, groups and/or items processed in the system, and/or a declaration of the programs, modules and/or other systems that it comprises. Systems can be nested (that is, can be sub-systems to other systems) to any depth. A SYSTEM is at the highest level of the member type hierarchy.

Refer to the relevant DataManager Interface Facility manual for further details about databases.

Member Definition Statements

ITEM Member Definition Statement

The ITEM member definition statement defines a data element.

Refer to "ITEM Syntax" on page 346 of the ITEM member type.

An ITEM member basically consists of:

```
>>──ITEM ──┬──────┬──────────────────────────────────────><
           └<<<<  ┘
           └form  ┘
```

where form is:

```
>>──form-keyword form-description ──┬──────────────────────────────────>
                                    └──USAGE ──┘

>>──┬──────────┬──┬──────────┬──┬──────────────┬──────────────────────>
    └──CONTENTS┘  └──HEADING ┘  └──USER-EXIT ┘
```

An ITEM member definition must commence with the member identifier keyword ITEM. The clauses and keywords forming the body of the member definition statement are all optional. They can be specified in any order except for form-keyword which if used, can only be preceded by common clauses and must be immediately followed by an associated form-description. Each form-keyword may optionally have a USAGE, CONTENTS, HEADINGS and USER-EXIT clause. Up to 15 versions of the form-keyword can be defined in an ITEM member.

Any common clauses must be declared either before the form-keyword or after that form keyword's associated form-description and clauses.

The Form-keyword

Form-keyword is one of the following:

ENTERED-AS	the form in which the item is input
HELD-AS	the form in which the item is held and processed within the computer, and/or held in its secondary storage
REPORTED-AS	the form in which the item is output
DEFAULTED-AS	when the item may relate to either input internal processing or output

The form-keyword may be followed by a version number and must be followed by a form-description.

An ITEM member definition can contain a maximum of 46 form-keywords. Within this total up to 15 versions of each of the keywords ENTERED-AS, HELD-AS and REPORTED-AS can be specified in any combination. Only one DEFAULTED-AS keyword is allowed.

Where two or more versions of a form is specified then each must be identified with a version number which need not be consecutive but must be in the range of 1 to 15. If a version number is not specified then a default value of 1 is assumed.

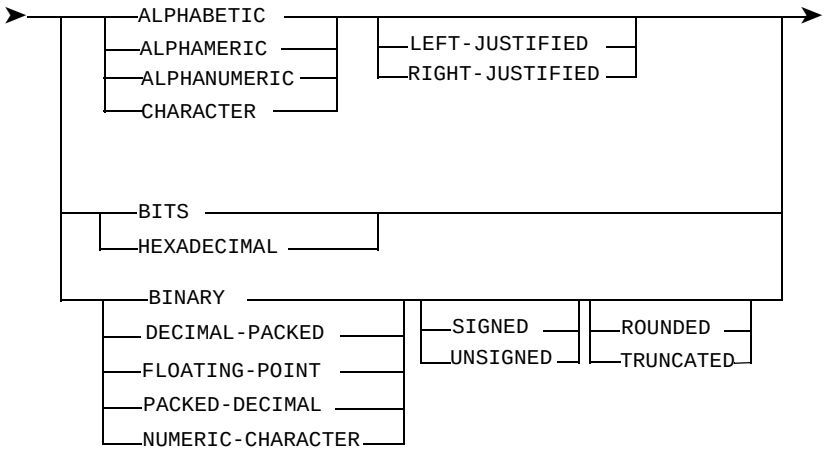
For example:

```
ITEM
DEFAULTED-AS form-description
ENTERED-AS        form-description
HELD-AS 1         form-description
HELD-AS 2         form-description
REPORTED-AS       form-description
```

Here there are two versions of the HELD-AS (internal processing) form. Since the version numbers for ENTERED-AS and REPORTED-AS are omitted, both are defaulted to one.

The Form-Description

A form-description directly follows a form-keyword, and defines the nature and form of the data described by the ITEM. The first part of form-description can be written as follows:



The keywords SIGNED and UNSIGNED may alternatively precede their associated form-description keywords except where that keyword is FLOATING-POINT.

This must be followed by a length as follows:

```
──┬─VARIABLE──────────┬──┬─COMPRESSED─┬──┬─p TO─┬──q──┬─K─┬──►
  ├─NULL-SUPPRESSED───┤  │            │  │      │     ├─M─┤
  └─FIXED─────────────┘  │            │  │      │     └─G─┘
                         │            │  │      │
──┬─WITH──────────────────────────────────────────────SIGN─┬──►
  │          └─SEPARATE──┬─LEADING──┬──┘
  │                      └─TRAILING─┘
```

p and *q* are unsigned decimal numbers specifying respectively the minimum and maximum lengths of the item. The elements *p* TO are present only in the definition of a variable length item. The element *q* may be zero. If it is zero, Manager Products will assume a value of one. The maximum value that can be declared for the length *q* depends on the associated form-description keyword.

The length units K, M, and G represent kilobytes, megabytes, and gigabytes respectively. They can only be specified when the form-description is CHARACTER and blob data is defined in the USAGE attribute.

Certain form-description keywords are relevant to Source Language Generation, though they can be used in other contexts.

The ROUNDED and TRUNCATED keywords are relevant to MARK IV Source Language Generation. The VARIABLE, NULL-SUPPRESSED and FIXED keywords are relevant to ADABAS Source Language Generation. The COMPRESSED keyword is relevant to SYSTEM 2000/80 Source Language Generation.

The WITH clause may be used only when both NUMERIC-CHARACTER is specified and UNSIGNED is not specified.

If the keyword SEPARATE is included in the WITH clause, then when Source Language Generation is performed from the member, an extra byte is generated to contain the sign. If SEPARATE is omitted from the WITH clause, the clause specifies which end of the field defined by the member contains the sign overpunch.

One way of describing the item is explained in this panel. Alternatively, you can describe the format of a data item by means of a PICTURE clause, or you can describe the item as having the same format as another item by using a NAME clause.

Form-Description Keywords

If the ALPHABETIC form-description keyword is specified:

- The item can contain letters of the alphabet and/or spaces
- The maximum value that can be declared for the length is 32767
- The units to which the length relates are characters.

If the ALPHAMERIC or ALPHANUMERIC form-description keyword is specified:

- The item can contain letters of the alphabet and/or numeric characters (hexadecimal F0 to F9) and/or spaces (hexadecimal 40)
- The maximum value that can be declared for the length is 32767
- The units to which the length relates are characters

If the BINARY form-description keyword is specified:

- The item can contain a binary number
- The maximum value that can be declared for the length is 31 or 31.0
- The units to which the length relates are decimal digits in the decimal equivalent of the maximum number the item can hold

If the BITS form-description keyword is specified:

- The item can contain a bit string.
- The maximum value that can be declared for the length is 4096
- The units to which the length relates are bits

If the CHARACTER form-description keyword is specified:

- The item can contain any printable and/or non-printable characters
- The maximum value that can be declared for the length is 32767
- The units to which the length relates are characters unless the length unit K, M, or G is specified

If the DECIMAL-PACKED or PACKED-DECIMAL form-description keyword is specified:

- The item can contain a packed decimal number (relevant to COBOL COMPUTATIONAL-3 or PL/I DECIMAL FIXED)
- The maximum value that can be declared for the length is 31 or 31.0
- The units to which the length relates are decimal digits

If the FLOATING-POINT form-description keyword is specified:

- The item can contain exponent and mantissa
- The maximum value that can be declared for the length is 16
- The units to which the length relates are decimal digits in the mantissa

If the HEXADECIMAL form-description keyword is specified:

- The item can contain a hexadecimal string
- The maximum value that can be declared for the length is 512
- The units to which the length relates are hexadecimal characters

If the NUMERIC-CHARACTER form-description keyword is specified:

- The item can contain numeric characters (hexadecimal F0 to F9)
- The maximum value that can be declared for the length is 31 or 31.0
- The units to which the length relates are decimal digits

The maximum and minimum length of an item with a BINARY, DECIMAL-PACKED, NUMERIC-CHARACTER or PACKED-DECIMAL form-description is expressed in the form:

n.m

where:

n is the number of digits before the decimal point

m is the number of digits after the decimal point. n plus m must not exceed 31.

Any non-significant zeros either before or after the decimal point are omitted. Therefore, if *n* is zero *n* can be omitted and if *m* is zero *m* can be omitted. For example, when *m* is zero, a value of 31.0 is expressed as 31. The decimal point itself is omitted if the figure is an integer.

Specifying the Form-Description Pictorially

To represent the form-description of an ITEM pictorially with symbols enter:

```
PICTURE 'picture'
```

Picture. A string of one to thirty characters. The character string must contain only picture symbols and repetition factors. It must contain at least:

- One of the symbols A X Y Z 9 *, or
- A floating S £ + or - (that is, a leading string of two or more consecutive occurrences of one of these symbols).

Repetition Factor

A repetition factor is an integer in parentheses which specifies the number of consecutive occurrences of the symbol it immediately follows. For example:

A(5)

is equivalent to AAAAA.

The value of the repetition factor must be in the range 1 to 32767.

Calculating the Length of an Item

The length of an item in reports is the sum of the repetition factors in the picture, subject to the following conditions:

- If the repetition factor of a symbol is not stated, then it is assumed to have a value of one
- The implied repetition factor of any K P or V symbol is excluded from the count
- In a numeric floating point picture, if the sign symbol of the mantissa and/or the exponent is omitted, then the defaulted sign symbol is included in the count.

Calculating the Number of Digit Positions

In numeric pictures the number of digit positions is the sum of the repetition factors of the following symbol:

```
9
```

In numeric edited pictures the number of digit positions is the sum of the repetition factors of the following symbols:

```
I R T Y Z 0 9 *
```

plus for each of the following symbols, the total of its repetition factors minus one:

```
S £ + -
```

In numeric floating point pictures the number of digit positions is the sum of the repetition factors of the following symbols:

```
I R T 9
```

PICTURE Clause Symbols

Picture Symbol	Description
A	Any letter or a space.
B	A space is inserted unless the position is required for a floating symbol.
CR	The characters CR are inserted if the item is negative. Otherwise, spaces are inserted.
DB	The characters DB are inserted if the item is negative. Otherwise, spaces are inserted.
E	Inserted in floating-point pictures on output to indicate that the exponent follows.
I	A decimal digit which is overpunched in the 12 punching position, signifying + , if the item is positive or zero and is output to a card punch.
K	The position of an implied exponent symbol in a floating-point PICTURE clause.
P	Implied decimal scaling position, if the location of the decimal point falls outside the item.
R	A decimal digit which is overpunched in the 11 punching position, signifying - , if the item is negative and is output to a card punch.
S	A + symbol is inserted if the item is positive or zero, a - symbol if the item is negative; or a floating sign position.
R	A decimal digit which, if the item is output to a card punch, is overpunched in the 12 punching position, signifying + , if the item is positive or zero, or in the 1 1 punching position, signifying - , if the item is negative.
V	The position of an implied decimal point.
X	Any character.
Y	A zero-suppressed decimal digit (replaced by a space if zero).
Z	A leading zero-suppressed decimal digit replaced by a space if zero and not preceded by significant digits.
/	A position into which the character / is inserted.
+	In numeric edited pictures: • a + symbol is inserted if the item is positive or zero • a space is inserted if the item is negative • a floating + position.

Picture Symbol	Description
	In numeric floating point pictures: • a + symbol is inserted if the following value (mantissa or exponent) is positive or zero • a space is inserted if the following value (mantissa or exponent) is negative. In numeric edited pictures: • a - symbol is inserted if the item is negative • a space is inserted if the item is positive or zero • a floating - position. In numeric floating point pictures: • a - symbol is inserted if the following value (mantissa or exponent) is negative • a space is inserted if the following value (mantissa or exponent) is positive.
,	A position into which a comma is inserted providing there is a non-suppressed preceding digit.
*	A decimal digit to be replaced by the symbol * if it is zero and not preceded by significant digits.
£	A symbol is inserted, or a floating position.
.	A decimal point is inserted on output.
0	A position into which a zero is inserted on output unless the position is needed for a floating symbol or rendered to a space by a floating symbol.
9	A decimal digit.

Note:

The £ can be replaced by any local currency character whose internal code is hexadecimal 5B.

The valid picture symbols in each picture type are summarized here:

PICTURE TYPE	FORMAT SYMBOL:																							
	A	B	CR	DB	E	I	K	P	R	S	T	V	X	Y	Z	0	9	£	+	-	*	,	.	/
ALPHABETIC	*																							
ALPHANUMERIC	*												*				*							
ALPHANUMERIC EDITED	*	*											*			*	*							*
NUMERIC								*				*					*							
NUMERIC EDITED		*	*	*		*		*	*	*	*	*		*	*	*	*	*	*	*	*	*	*	*
NUMERIC FLOATING POINT					*	*	*		*		*	*					*		*	*			*	

PICTURE Clause Floating Symbols

The following symbols may be floating symbols:

S + - E

These symbols become floating symbols if they occur in a picture as two or more consecutive leading characters, or as three or more leading characters interspersed, but not before the second character position, with any combination of the following:

- One full stop or one V symbol
- Any number of B symbols
- Any number of 0 symbols
- Any number of comma symbols

The string of characters from the first to the last floating symbol, together with any of the insertion characters B 0 or comma immediately following the last floating symbol, is called a floating string.

If the floating string does not contain a full stop, the second and subsequent occurrences of the floating symbol represent numeric digit positions which, together with any preceding interspersed symbols, are not preceded by a significant digit. The floating symbol is inserted into the character position immediately preceding the first digit position occupied by a significant digit.

For example, if a picture is declared as £££££ and the item contains zero, the item is output as all spaces, while if a picture is declared as ££££9 and the item contains zero, the item is output as £0 (because the symbol 9 implies that the digit is to be regarded as significant regardless of its value). If the item contains one, and the picture is £££££ or ££££9, it is output as £l.

If the floating string contains a full stop the same rules apply, except that if the item is non-zero, the floating symbol can float only as far as the character position immediately preceding the full stop (decimal point).

Alphabetic PICTURE Clause

The permitted picture symbol is:

A

The maximum number of character positions is 32767.

The following are valid alphabetic PICTURE clauses:

- ’A’
- ’A(1)’
- ’A(32767)’
- ’AA(32765)A’.

Note:

The second example is valid, although the repetition factor of one is superfluous.

These are invalid alphabetic PICTURE clauses:

’A(32768)’
’AA(32767)A’
’AAA(23)AA(100)AAAAAAAAAAAAAAAAA’.

The reasons for invalidity are respectively:

- The repetition factor is greater than 32767
- The length of the item would be greater than 32767, that is, 1 (A) + 32766 (32766 times A) + 1(A) = 32768
- The character string has more than 30 characters

Alphanumeric PICTURE Clause

These are the permitted picture symbols:

A X 9

The maximum number of character positions is 32767.

These are valid alphanumeric PICTURE clauses:

'X'
'X9'
'AX'
'9XA'
'A9'
'A(32765)X9'

This is an invalid alphanumeric PICTURE clause:

'A(32766)X9'.

The example is invalid because the length of the item would be greater than 32767, that is, 32766 (32766 times A)+ 1 (X) + 1 (9) = 32768.

Alphanumeric Edited PICTURE Clause

These are the permitted picture symbols:

A B X 0 9

The maximum number of character positions is 32767.

An alphanumeric edited PICTURE clause must contain:

- At least one B and one X, or
- At least one 0 and one X, or
- At least one 0 and one A.

These are valid alphanumeric edited PICTURE clauses:

'XXXBXXX'
,XX00XXBAA'
'A0'.

These are invalid alphanumeric edited PICTURE clauses:

'AB'
'B0'.

Numeric PICTURE Clause

These are the permitted picture symbols:

P V 9

The maximum number of character positions is 31 for digits and 1 for sign.

The rules governing numeric PICTURE clauses are as follows:

- The symbol P may only occur as a string of characters to the left or right of a 9 symbol
- If there are P symbols, then V may only occur:
  - To the left of a left-hand P string, or
  - To the right of a right-hand P string.
- If there are no P symbols then a V symbol can occur anywhere.

The following are valid numeric PICTURE clauses:

’999’
’99PPP’
’999V99’
’99PPV’
’PP999’
’9(8)V9(10)’
’VPP99’.

The following are invalid numeric PICTURE clauses:

’PP999PP’
’PP99V’
’V99PP’
’9(22)V9(11)’.

The reasons for invalidity are respectively:

- The P symbols occur on both sides
- The V symbol is misplaced
- The V symbol is misplaced
- The length of the item would be greater than 32, that is, 22 (22 times 9) + 1 (V) + 11 (11 times 9) = 34.

Numeric Edited PICTURE Clause

The permitted picture symbols are:

```
B CR DB I P R ST V Y Z 0 9 £ * / + - comma and full stop
```

The maximum number of character positions is 127, of which the maximum number of digit positions is 31.

The rules governing numeric edited PICTURE clauses are as follows:

- The following symbols must not occur more than once: CR DB I R T V and full stop
- The symbols in the following groups are mutually exclusive:
  - P and full stop
  - + - S CR DB T I and R
  - Z * and floating + - S.
- The symbols S + and - can occur only as the first or last character of the picture, or in a floating string. It must not be both the first and the last character unless it is in a floating string extending over the whole picture
- The only characters that can precede a symbol are a single S + - or characters of a floating string
- The last symbol must not be / or full stop or comma
- An * or Z symbol must not occur to the right of the symbols 9 T I R, nor to the right of a full stop or V symbol if a 9 symbol is to the right of the * or Z symbol
- If there are two or more P symbols, they must be consecutive and either:
  - To the left of the picture, optionally preceded by a single S + - or symbol (or by a V symbol which though permitted is redundant),

  or

  - To the right of the picture, optionally followed by a single S + - CR or DB symbol (or a V symbol which though permitted is redundant)
- CR or DB can occur only as the last symbol.

The following are valid numeric edited PICTURE clauses:

’.’
-’PPP99+’
-’++V+’
-’999PPV’
-’---B--,
-’ZZ99’
-’,,9.99’
-’R9(5)V9’.

The following are invalid numeric edited PICTURE clauses:

-’++VV+’
-’CR999’
-’+ +SSS’
-’99/99/99/’
-’VSP99’
-’999ZZ’
-’999’
-’9PP99’.

Numeric Floating Point PICTURE Clause

The permitted picture symbols are:

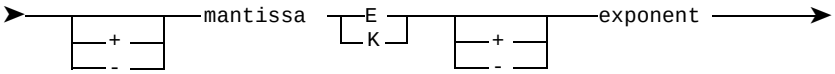
```
E I K R T V 9 + - and full stop
```

The maximum number of character positions is:

- mantissa 17 (16 for digits and 1 for sign)
- exponent 3 (2 for digits and 1 for sign)
- inserted E 1.

The rules governing numeric floating point PICTURE clauses are:

- The range is 5.4 x 10 to exponent -79 to 0.72 x 10 to exponent + 76
- The picture is expressed as:



Mantissa can contain:

- From one to sixteen 9 symbols, or
- From one to fifteen 9 symbols and one I R or T symbol, or
- One I R or T symbol, and optionally
- A full stop and/or a V symbol.

The symbols can be in any order. If an I R or T is included, there must be no preceding + or - symbol.

Exponent can contain:

- One or two 9 symbols or
- A 9 symbol with one I R or T symbol or
- One I R or T symbol

If an I R or T symbol is included, there must be no preceding + or - symbol.

- If you omit the sign symbol from before the mantissa or the exponent, and no I R or T symbol occurs in either of them, a - symbol is defaulted. You will see a warning message and processing will continue.

These are valid numeric floating point clauses:

'99V9E-99'
'99.9E-9'
'+ V9(16)E+9'
'9(7)K9'
'-.9K+99'.

These are invalid numeric floating point clauses:

'+9(29)E+99'
'9.9K999'
'E+99'
'9.9E7'
'9.99V9E99'
'E-99'

Specifying That Two Items Have the Same Form-Description

To show that two items in a dictionary have the same form-description, enter:

```
➤──────NAME item-name──┬──────────────────────────────────➤
                       └──version──┘
```

where:

item-name is the name of a referenced ITEM which has the same form-description as the defined ITEM

version identifies a particular version of the referenced ITEM. If you do not specify a number, version 1 is assumed.

Use of the NAME form-description has three effects:

- The defined ITEM will have the same form-description as the referenced ITEM
- If the referenced ITEM has a USAGE clause and the defined ITEM has not, then this USAGE clause will also apply to the defined ITEM. However, if both the referenced ITEM and the defined ITEM have USAGE clauses, then the USAGE clause for the defined ITEM will override that of the referenced ITEM. The same is true for the CONTENTS, HEADINGS, and USER-EXIT clauses.
- Any change in the form-description of the referenced ITEM will automatically change that of the defined ITEM. The same will apply to any shared USAGE, CONTENTS, HEADINGS, and USER-EXIT clauses.

The form-description of an ITEM must be compatible with its CONTENTS clause. This applies to CONTENTS clauses shared with a referenced ITEM as well as those associated solely with the ITEM being defined.

Usage Definition in ITEM Member Definition

The optional usage clauses are mainly used for Source Language Generation, but may also be used for documentation purposes.

They have the form:

```
➤──┬──────────────────────────────────────┬──┬──────────────────┬──➤
   └──USAGE──┬──DATE──────────────────────┬┘  └──DISTINCT-TYPE──┘
             ├──GRAPHIC────────────────────┤
             ├──INDEX──────────────────────┤
             ├──MONEY──────────────────────┤
             ├──POINTER────────────────────┤
             ├──TIME───────────────────────┤
             ├──TIMESTAMP──────────────────┤
             ├──LOB────────────────────────┤
             ├──CLOB───────────────────────┤
             ├──BLOB───────────────────────┤
             ├──DBCLOB─────────────────────┤
             └──ROWID──────────────────────┘
```

The DATE and MONEY keywords are particularly relevant to System 2000/80 Source Language Generation.

The POINTER keyword is particularly relevant to the generation of PL/I data description statements; an item with this usage can be generated as a PL/I pointer variable.

The INDEX keyword requires the item to have a minimum length p of four.

The LOB, CLOB, BLOB, and DBCLOB keywords document data representing a large binary object; this attribute can be used to document and generate DB2 data definitions.

The DISTINCT TYPE and ROWID keywords are for DB2 purposes.

These clauses can also be used in other context.

Specifying Conditions That Define the Form and Version

This provides a range of stated conditions for defining the form and version of the item more closely. The CONTENTS clause has the form:

```
>------CONTENTS contents------------------------------------------>

>-----+-------------------+-+-------------------------------------+-->
      |                   | |  <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<  |
      |--IF clause--------| |--ELSE content-----------------------|
                                                  |--IF clause--|
```

where:

content is:

```
>-----+-ALPHABETIC-------------+-+-------------------------------->
      |-ALPHAMERIC-------------| |--FORMAT 'format'--|
      |-ALPHANUMERIC-----------|
      |-CHARACTER--------------|
      |-BITS-------------------|
      |-BINARY-----------------|
      |-HEXADECIMAL------------|
      |-NUMERIC-CHARACTER------|
      |-FLOATING-POINT---------|

>-----+-------------------------------------------------------------->
      |        <<<<<<< , <<<<<<<<<<<<<<<<<<<
```

format is a symbol or a string of not more than 30 characters.

literal may be:

- A delimited character string, or
- A decimal number of no more than 31 digits, or
- A floating point number.

If the form-description of an ITEM is a Numeric Edited PICTURE clause then the literal must be enclosed in quotes.

The following keywords are also valid for use with a literal:

```
HIGH-VALUES, LOW-VALUES, SPACES, QUOTES, ZEROES, ZEROS, ALL.
```

When ALL is specified, a string *'str'* may be used in addition to the above keywords to qualify ALL.

'str' is identical to *literal* except when content is:

```
DECIMAL-PACKED, PACKED-DECIMAL, FLOATING-POINT or NUMERIC-CHARACTER.
```

In these cases, *'str'* is a single numeric character.

Additionally, when content is DECIMAL-PACKED, PACKED-DECIMAL, FLOATING-POINT or NUMERIC-CHARACTER, the keywords HIGH-VALUES, LOW-VALUES, SPACES and QUOTES are not valid.

condition-name is a name of not more than 32 characters which must conform to the rules for member names.

The CONDITION-NAME clause is used for the Source Language Generation of COBOL level 88 data description statements.

Any *content* can be specified as conditional, that is, applying only if a stated condition or combination of conditions is satisfied. For a content to be conditional, it must be immediately followed by an IF clause. Up to 16 conditional terms may be specified within an IF clause. All contents definitions can have an alternative contents which may also be conditional. Alternative content declarations must be separated by the keyword ELSE. The first conditional term of an alternative content declaration is added to the total number of conditional terms specified in the preceding IF clause.

An item’s form-description and CONTENTS clause must be compatible. This applies to content-description keywords, FORMAT clause symbols and the literals of both the RANGE and IF clauses.

Examples:

```
ITEM
HELD-AS PACKED-DECIMAL 1
CONTENTS IS 0 ELSE RANGE 3 TO 8
;

ITEM
ENTERED-AS ALPHANUMERIC 30
CONTENTS NUMERIC IF RECORD-TYPE EQ 20
OR RECORD-TYPE EQ 99
ELSE ALPHABETIC IF RECORD-TYPE GT 0
 AND RECORD-TYPE LT 20
ELSE ALPHANUMERIC IF RECORD-TYPE > 20
   AND RECORD-TYPE < 99
;
```

ITEM FORMAT Symbols

The following table lists the symbols used in the FORMAT sub-clause of the CONTENTS clause with a brief explanation of their meaning:

Symbol	Meaning
A	Any letter or a space.
N	A numeric digit.
O	A first or last numeric digit which is overpunched if the item is negative.
P	A first or last numeric digit which is overpunched if the item is positive or Zero.
S	A space.
X	Any character.
.	A decimal point or a full stop in a date.
/	Forward slash character for use only in a date.
D	Date format; position occupied by the day.
M	Date format; position occupied by the month.
Y	Date format; position occupied by the year.
+	A plus sign if the item is positive, or zero and a minus sign if the item is negative. Must be either the first or last character in a FORMAT which otherwise comprises only N symbols.
-	A space if the item is positive, or zero and a minus sign if the item is negative. Must be either the first or last character in a FORMAT which otherwise comprises only N symbols.

FORM-DESCRIPTION	FORMAT SYMBOL:										
	A	N	O	P	S	X	+	-	.	/	DMY
ALPHABETIC	*				*						
ALPHAMERIC	*	*	*	*	*		*	*	*		*
ALPHANUMERIC	*	*	*	*	*		*	*	*		*
BINARY							*	*			
BITS											
CHARACTER	*	*			*	*			*	*	*
DECIMAL-PACKED		*					*	*	*		*
FLOATING-POINT		*					*	*	*		
HEXADECIMAL											
NUMERIC-CHARACTER		*	*	*			*	*	*		*
PICTURE	*	*	*	*	*	*	*	*	*	*	*
* indicates permissible combinations of symbols and keywords.											

Examples:

```
ITEM
REPORTED-AS PICTURE '99X99X99'
CONTENTS FORMAT 'DD/MM/YY'
;

ITEM
ENTERED-AS ALPHANUMERIC 7
CONTENTS FORMAT 'NNNNNNX'
;
```

Compatible Form-description and Contents-description Keywords

To display the full list of compatible form-description and content-description keywords, page through the following screens by pressing ENTER.

Form-description	Contents-description	
ALPHABETIC	ALPHABETIC CONDITION-NAME FORMAT	IS RANGE TO
ALPHAMERIC	ALPHABETIC ALPHAMERIC ALPHANUMERIC CONDITION-NAME FORMAT	IS NUMERIC-CHARACTER RANGE TO
BINARY	BINARY CONDITION-NAME FORMAT	IS RANGE TO
BITS	BITS CONDITION-NAME IS	RANGE TO
CHARACTER	ALPHABETIC ALPHAMERIC ALPHANUMERIC CHARACTER CONDITION-NAME	FORMAT IS NUMERIC-CHARACTER RANGE TO
DECIMAL-PACKED PACKED-DECIMAL	CONDITION-NAME DECIMAL-PACKED FORMAT IS	PACKED-DECIMAL RANGE TO
FLOATING-POINT	CONDITION-NAME FLOATING-POINT FORMAT	IS RANGE TO
HEXADECIMAL	CONDITION-NAME HEXADECIMAL IS	NUMERIC-CHARACTER RANGE TO
NUMERIC-CHARACTER	CONDITION-NAME FORMAT IS	NUMERIC-CHARACTER RANGE TO
NAME PICTURE	ALPHABETIC ALPHAMERIC ALPHANUMERIC CONDITION-NAME CHARACTER FORMAT	FLOATING-POINT IS NUMERIC-CHARACTER RANGE TO

The IF Keyword

The IF clause compares the contents of one item with the contents of another item or a sub-literal. Up to sixteen conditional terms may be specified within each IF clause. If there are two or more conditional terms in an IF clause they must be separated by an AND or OR keyword. Any number of IF clauses may appear in an individual member definition. The IF clause has the form:

```
                          ┌─ AND─┐
        <<<<<<<<<<<<<<<<  ─ OR ─┘<<<<<<<<<<<<<<<<<<<<<<<<
➤──IF item ──────────┬─EQ──┬──┬──item ──────────┬──────────────────➤
                     └─V─┘  │  │                 └─V─┘
                            ├─= ─┤  └──sub-literal──┘
                            ├─NE─┤
                            ├─GT─┤
                            ├─> ─┤
                            ├─GE─┤
                            ├─LT─┤
                            ├─< ─┤
                            └─LE─┘
```

where:

item is the name of an item

v is an unsigned integer in the range 1 to 15, specifying the version of the associated item. It has a default value of 1.

sub-literal can be one of these:

- A delimited character string of not more than 256 characters,
- An undelimited signed or unsigned decimal number of not more than 31 digits, optionally with a decimal point,
- An undelimited signed or unsigned floating point number.

If you specify a sub-literal, it must be compatible with the item's form-description and the contents-description of its CONTENTS clause.

The operators in IF clauses have these meanings:

- EQ or = means equal to
- NE means not equal to
- GT or > means greater than
- GE means greater than or equal to
- LT or < means less than
- LE means less than or equal to.

Specifying a Line of Headings

This optional clause defines a line (or lines) of headings for an item.

For example, in MARK IV Source Language Generation, a PRODUCE command generates file definition forms; the column headings for these forms can be defined in a HEADINGS clause.

The clause has the form:

where:

```
➤──HEADINGS '<<string<<'──────────────────────────────────────➤
```

`'string'` is a delimited character string; each generates a line of heading.

When processed, the strings are concatenated to form one string. No spaces are inserted in this process, so if any are needed they must be present in the HEADINGS clause.

If a string is preceded by a comma, then it is not concatenated but output to a separate line.

For example:

```
HEADINGS 'DATE OF','ORDER'
```

would produce:

```
DATE OF
ORDER
```

Specifying the Module in Which the Form and Version is Always Processed

The USER-EXIT clause declares the name of a module in which the form and version of the item is always processed.

The clause has the form:

```
➤──┬──────────────────────┬──────────────────────────────────➤
   └──USER-EXIT module-name──┘
```

where:

module-name is the name of a module.

Example:

```
ITEM
ENTERED-AS ALPHANUMERIC 7
CONTENTS FORMAT ’NNNNNNX’
USER-EXIT CHKDGT11
```

This shows that the ENTERED-AS form of the item is always submitted to a check-digit validation module, CHKDGTH, before further processing.

Examples

```
ITEM
HELD-AS  PACKED 2.2
CONTENTS RANGE 0 TO 99.99
;

ITEM
REPORTED-ASPICTURE '+9,999'
CONTENTS RANGE '-50' TO '+4,999'
;

ITEM
ENTERED-AS   NUMERIC 6
CONTENTS     FORMAT 'DDMMYY'
REPORTED-AS  PICTURE '99X99X99'
CONTENTS     FORMAT 'DD/MM/YY'
;

ITEM
ENTERED-AS      ALPHANUMERIC 30
CONTENTS        NUMERIC IF RECORD-TYPE EQ 20
OR RECORD-TYPE EQ 99
ELSE ALPHABETIC IF RECORD-TYPE GT 0
  AND RECORD-TYPE LT 20
ELSE ALPHANUMERIC IF RECORD-TYPE > 20
     AND RECORD-TYPE < 99
;
```

ITEM Syntax

```
ITEM ┬─<<<<─┬─┬─common-clauses─┬
     │ form │ │                │
     ┴      ┴ ┴                ┴
;
.
```

where *form* is:

```
DEFAULTED-AS
ENTERED-AS ─┬─version─┬
HELD-AS    ─┤         │
REPORTED-AS ┴         ┴
form-description
USAGE ─┬─DATE        ─┬─┬─DISTINCT-TYPE─┬
       ├─GRAPHIC      ─┤ │               │
       ├─INDEX        ─┤ │               │
       ├─MONEY        ─┤ │               │
       ├─POINTER      ─┤ │               │
       ├─TIME         ─┤ │               │
       ├─TIMESTAMP    ─┤ │               │
       ├─LOB          ─┤ │               │
       ├─CLOB         ─┤ │               │
       ├─BLOB         ─┤ │               │
       ├─DBCLOB       ─┤ │               │
       └─ROWID        ─┴ ┴               ┴
CONTENTS content ─┬─<<<<<<<<< , <<<<<<<<<─┬─┬─HEADINGS 'string'─┬─┬─<< , <<─┬
                  │                       │ │                   │ │          │
                  └─if-clause─────────────┴ ┴                   ┴ ┴          ┴
USER-EXIT module-name
```

where:

version is an unsigned integer in the range of 1 to 15.

form-description is:

```
PICTURE 'picture'
NAME item-name-r ─┬─version-r─┬
                  │           │
                  ┴           ┴
ALPHABETIC ─┬─┬─LEFT-JUSTIFIED─┬─┬─var-a─┬
            │ │                │ │       │
            │ │                │ │       │
            │ │                │ │       │
```

where:

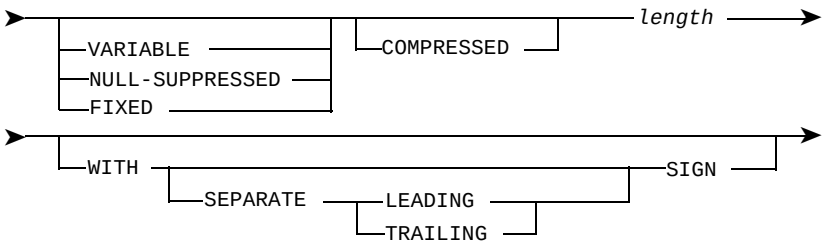
picture is a symbol or a string of not more than 30 characters. This table shows picture symbols:

PICTURE TYPE	FORMAT SYMBOL:																							
	A	B	CR	DB	E	I	K	P	R	S	T	V	X	Y	Z	0	9	£	+	-	*	,	.	/
ALPHABETIC	*																							
ALPHANUMERIC	*												*				*							
ALPHANUMERIC EDITED	*	*											*			*	*							*
NUMERIC								*				*					*							
NUMERIC EDITED		*	*	*		*		*	*	*	*	*		*	*	*	*	*	*	*	*	*	*	*
NUMERIC FLOATING POINT					*	*	*		*		*	*					*		*	*			*	
* indicates permissible combinations of symbols within picture types.																								

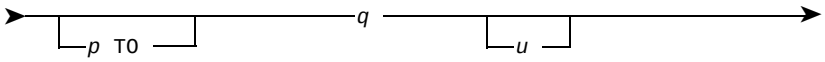
item-name-r is the name of an item.

version-r is an unsigned integer in the range 1 to 15.

var-a is:



length is:



p and *q* are unassigned decimal numbers specifying respectively the minimum and maximum lengths of the item.

u is a length unit of K, M, or G which can be used when defining blob data.

var-b is:

```
➤──┬──┬─BINARY──────────────────┬──┬─SIGNED───┬──┬──┬─ROUNDED───┬──┬──➤
   │  ├─DECIMAL-PACKED──────────┤  └─UNSIGNED─┘  │  └─TRUNCATED─┘  │
   │  ├─FLOATING-POINT──────────┤                │                │
   │  ├─PACKED-DECIMAL──────────┤                │                │
   │  └─NUMERIC-CHARACTER───────┘                │                │
   │                                             │                │
   │  ┌─SIGNED───┐  ┌─BINARY──────────────┐      │                │
   │  └─UNSIGNED─┘  ├─DECIMAL-PACKED──────┤      │                │
   │                ├─FLOATING-POINT──────┤      │                │
   │                ├─PACKED-DECIMAL──────┤      │                │
   │                └─NUMERIC-CHARACTER───┘      │                │
   └──────────────────────────────────────────────┘                │
```

content is:

```
➤──┬─ALPHABETIC──────────┬──┬─FORMAT 'format'─┬──➤
   ├─ALPHAMERIC──────────┤  └─────────────────┘
   ├─ALPHANUMERIC────────┤
   ├─CHARACTER───────────┤
   ├─BITS────────────────┤
   ├─BINARY──────────────┤
   ├─DECIMAL-PACKED──────┤
   ├─PACKED-DECIMAL──────┤
   ├─HEXADECIMAL─────────┤
   ├─NUMERIC-CHARACTER───┤
   └─FLOATING-POINT──────┘

➤──┬──────────────────────────────────────┬──➤
   │      <<<<<<< , <<<<<<<<              │
   ├─RANGE literal TO literal─────────────┤
   │                                      │
   │    <<<<<<< , <<<<<<<<                │
   └─IS literal TO literal────────────────┘

➤──┬─CONDITION-NAME condition-name─┬──➤
   └───────────────────────────────┘
```

where:

format is a symbol or a string of not more than 30 characters. This table shows format symbols:

FORM-DESCRIPTION	FORMAT SYMBOL:										
	A	N	O	P	S	X	+	-	.	/	DMY
ALPHABETIC	*				*						
ALPHAMERIC	*	*	*	*	*		*	*	*		*
ALPHANUMERIC	*	*	*	*	*		*	*	*		*
BINARY							*	*			
BITS											
CHARACTER	*	*			*	*			*	*	*
DECIMAL-PACKED		*					*	*	*		*

FLOATING-POINT		*					*	*	*		
HEXADECIMAL											
NUMERIC-CHARACTER		*	*	*			*	*	*		*
PICTURE	*	*	*	*	*	*	*	*	*	*	*

* indicates permissible combinations of symbols and keywords.

literal is:

```
►──┬──────────────────────────────────────────────────────────────────────►
   │                    sub-literal                                       │
   ├──HIGH-VALUES ──┤                                                     │
   ├──LOW-VALUES ───┤                                                     │
   ├──SPACES ───────┤                                                     │
   ├──QUOTES ───────┤                                                     │
   ├──ZEROES ───────┤                                                     │
   ├──ZEROS ────────┤                                                     │
   │                                                                     │
   └──ALL ──┬──'str' ──────────────────────────────────────────────────┘
            ├──HIGH-VALUES ──┤
            ├──LOW-VALUES ───┤
            ├──SPACES ───────┤
            ├──QUOTES ───────┤
            ├──ZEROES ───────┤
            └──ZEROS ────────┘
```

where:

sub-literal is:

- A delimited character string of not more than 256 characters or a numeral in the range 1 to 9
- An undelimited signed or unsigned decimal number of not more than 31 digits, optionally with a decimal point
- An undelimited signed or unsigned floating point number

str is identical to literal excepts when content is DECIMAL-PACKED, PACKED-DECIMAL, FLOATING-POINT, NUMERIC-CHARACTER, or BINARY. In these cases, *str* is a single numeric character.

Additionally, when *content* is DECIMAL-PACKED, PACKED-DECIMAL, FLOATING-POINT or NUMERIC-CHARACTER, the keywords HIGH-VALUES, SPACES and QUOTES are not valid.

condition-name is a name of not more than 32 characters, conforming to the rules for member names.

if-clause is:

```
➤──┬─IF clause─┬──┬──────────────────────────────────────────────┬──➤
   │           │  │  <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<  │
   │           │  └─ELSE content──────────────────────────────────┤
   │           │                                  ┌─IF clause─┐   │
   └───────────┘                                  └───────────┘   └
```

clause is:

```
                                  ┌─ AND ─┐
          <<<<<<<<<<<<<<<<<<<<<<<<<┤ OR  ├<<<<<<<<<<<<<<<<<<
```

GROUP Member Type

A GROUP member has the form:

```
GROUP [form-keyword] [alignment-keyword] [CONTAINS] [KEYS]
                                [USER EXIT] [common clauses]
;
```

A GROUP member definition must commence with the member identifier keyword GROUP. The clauses and keywords forming the body of the member definition statement are all optional. They can be specified in any order except for the CONTAINS clause which must not precede the form-keyword, or be followed by an alignment keyword.

Form-keyword

Form-keyword specifies the form required for all items contained within the GROUP. It is one of these:

ENTERED-AS	The form in which the item is input
HELD-AS	The form in which the item is held and processed within the computer, and/ or held in its secondary storage
REPORTED-AS	The form in which the item is output
DEFAULTED-AS	When the item may relate to either input, internal processing or output.

If no form-keyword is specified, it defaults to DEFAULTED-AS. Only one form-keyword may be used in each member definition.

If the same combination of items constitutes a group for internal processing (the HELD-AS form) and for output (the REPORTED-AS form), and if the definitions of these two forms differ, then the combination must be defined as two separate GROUP members, with different names.

If the member is to be used for Source Language Generation, all groups and items within it must have the same form-keyword as this group. However, the consistency of form-keywords in groups and subordinate groups is not checked during encoding.

Alignment-keyword in GROUP Member Definition

Alignment-keyword is one of the following:

```
ALIGNED
UNALIGNED
NOT-ALIGNED
```

If no alignment-keyword is specified, the default is to UNALIGNED. UNALIGNED and NOT-ALIGNED are synonymous.

UNALIGNED or NOT-ALIGNED means that binary items and floating-point items declared (as individual items or as elements of an array) as being contained in the group are not necessarily aligned to word or half-word boundaries and that bit string items are not necessarily aligned to byte boundaries.

ALIGNED is the equivalent of COBOL SYNCHRONIZED or PL/I ALIGNED. ALIGNED means that all binary items and all floating point items declared (as individual items or as elements of an array) as being contained in the group are aligned to half word, full word or double-word boundaries. as follows:

- Binary items having a length of four decimal digits or less occupy a complete half word
- Binary items having a length of from five to nine decimal digits occupy a full word
- Binary items having a length of from ten to eighteen decimal digits occupy two full words, but are not necessarily aligned to a double word boundary
- Floating-point items having six digits or less in the mantissa occupy a full word
- Floating-point items having from seven to sixteen digits in the mantissa occupy a double word.

ALIGNED also causes any bit string items to be aligned to byte boundaries.

Where an alignment is being specified for a group and all its components then the alignment-keyword must be declared prior to the CONTAINS clause in the GROUP member definition statement. For example:

```
GROUP
HELD-AS ALIGNED USER-EXIT MOD4071
;
```

A group's alignment may be overridden for the individual components of the group by specifying an alternative alignment-keyword within the GROUP member's CONTAINS clause, after the selected component's name and preceding any associated ELSE and IF clauses. Alignment-keywords may also be specified in the same manner for the components of a group for which no alignment-keyword has been declared. For example:

```
GROUP
HELD-AS ALIGNED
CONTAINS
;
ITEM1
,ITEM-2
,ITEM3 UNALIGNED
;
```

in this example the GROUP and therefore ITEM1 and ITEM2 are aligned, while ITEM3 is not.

ALIGNED or UNALIGNED keyword does not apply to items contained within subordinate groups declared (as individual groups or as elements of an array) as being contained in the group here defined. The data definitions of the subordinate groups determine the alignment or non-alignment of such indirectly-referenced items. It is therefore not meaningful to include alignment keywords in a content declaration that declares a subordinate group or an array of subordinate groups.

CONTAINS Clause in GROUP Member Definition

The CONTAINS clause is used to declare the contents of the group under a range of different stated conditions. The contents may be an item, a subordinate group or an array.

Up to 16 conditional terms may be specified using IF clauses. Any definition may have alternative contents which may also be conditional.

Alternative content declarations must be separated by the keyword ELSE. The first conditional term of an alternative content declaration is added to the total number of conditional terms specified in the preceding IF clause.

The CONTAINS clause has the form:

```
CONTAINS content [IF clause] [ELSE content [IF clause]]...
       [,content [IF clause] [ELSE content [IF clause]]...]...
```

where *content* is:

```
{ |group-name          | [alignment-keyword]                  }
{ |item-name [version] |            [KNOWN-AS local-name]    }
{                                                             }
{ (|integer                 |)|group-name          |         }
{  |item-name-1 [version]   | |item-name [version]|         }
{                 [alignment-keyword][KNOWN-AS local-name]    }
{                                   [INDEXED-BY index-name]   }
```

group-name is the name of a subordinate group contained within the group here defined.

item-name is the name of an item.

version is an unsigned integer in the range 1 to 15, being a number specifying which version of the relevant item is relevant to this group. The version is within the form (ENTERED-AS, HELD-AS, REPORTED-AS or DEFAULTED-AS) specified for this group. If version is omitted, a default value of 1 is assumed.

alignment-keyword is UNALIGNED, NOT-ALIGNED or ALIGNED and determines whether or not binary items and floating point items declared as being contained in the group are aligned to half word, full word or double word boundaries. The keyword must precede any IF or ELSE clauses.

integer is an unsigned integer of from 1 to 18, being the number of times item-name or group-name occurs in the array.

item-name-1 is the name of an item. This form of array declaration declares that when the group here defined is processed by an application program or module, the number of times item-name or group-name occurs in the array is contained in the item-name-l.

local-name is an alternative name for any item or subordinate group within the group, by which it is to be known when record layouts or source language descriptions are generated from this data definition by the DataManager Source Language Generation facility (selectable unit DMR-SL1).

index-name is the name used as the index name when COBOL data descriptions are generated by Source Language Generation.

The *local-name* and *index-name* are not separately recorded in the data dictionary and so cannot be interrogated and can be the same as another name, an alias or a catalog classification. Local-name is the name by which the contained member is known only within the group defined by this data definition. Both local-names and index-names must conform to the rules for member names.

Refer to "ITEM Member Definition Statement" on page 322 for details of the IF clause and of version numbers.

Refer to the *ASG-ControlManager User's Guide* for details of the rules governing member names.

KEYS Clause in GROUP Member Definition

The KEYS clause allows the key fields of a group to be declared for documentation purposes. The clause has the form:

```
KEYS {item-name  } [{UNIQUE    }] [{ASCENDING }]
     {group-name } [{DUPLICATED}]  {DESCENDING}

     [{item-name  } [{UNIQUE    }] [{ASCENDING }]]...
      {group-name } [{DUPLICATED}]  {DESCENDING}
```

item-name is the name of an ITEM.

group-name is the name of a subordinate GROUP.

UNIQUE means a key value has only one occurrence.

DUPLICATED means a key value may have many occurrences.

ASCENDING and *DESCENDING* indicate whether the file is ordered in ascending or descending order of the key.

USER-EXIT Clause in GROUP Member Definition

The USER-EXIT clause declares the name of a module in which the group is always processed. The clause has the form:

```
USER-EXIT module-name
```

module-name is the name of a module.

Example

```
GROUP ENTERED-AS CONTAINS
   DD,
   MM,
   YY
   USER-EXIT DATEVAL
;
```

In this example, DATEVAL is the name of a module which checks for valid day, month and year whenever the above group is input.

Examples of GROUP Members

Example 1

```
GROUP HELD-AS CONTAINS
   MONTH-NO,
   (12) MONTH-TOTALS-LAST-YEAR,
   (MONTH-NO) MONTH-TOTALS-THIS-YEAR,
;
```

This example illustrates a simple GROUP data definition statement. The group comprises an item and two item arrays. The two ways in which an array can be declared are illustrated.

Example 2

```
GROUP CONTAINS
      ARTICLE-NO,
  CUSTM-ORDER-NO IF REC-TYPE <20
  ELSE SUPPL-ORDER-NO IF REC-TYPE GE 20 AND REC-TYPE < 40
  ELSE PRODN-ORDER-NO IF REC-TYPE GE 40 AND REC-TYPE < 60
;
```

In this example the group has two parts, the second of which has alternative contents. The example illustrates the use of IF and ELSE clauses. As the statement contains no form keyword, a default form of DEFAULTED-AS is implied.

GROUP Syntax

```
GROUP   [ ENTERED-AS      ] [ ALIGNED       ]
        [ HELD-AS         ]   UNALIGNED     ]
        [ REPORTED-AS     ]   NOT-ALIGNED   ]
        [ DEFAULTED-AS    ]

    [CONTAINS content [IF clause] [ELSE content [IF clause] ]...

         [, content [IF clause] [ELSE content [IF clause]]...]...]
    [KEYS  [ item-name  ] [ UNIQUE     ] [ ASCENDING  ]
           [ group-name ] [ DUPLICATED ] [ DESCENDING ]

           [, [ item-name  ] [ UNIQUE     ] [ ASCENDING  ]]...]
              [ group-name ] [ DUPLICATED ] [ DESCENDING ]

    [USER-EXIT module-name]

    [common-clauses]

  [ ; ]
  [ . ]
```

where:

content is:

```
{ [ item-name [version] ]   [ ALIGNED      ]   [USER-EXIT module-name] }
{ [ group-name          ]   [ UNALIGNED    ]                           }
{                           [ NOT-ALIGNED  ]                           }
{ ( [ integer                ] ) [ item-name [version] ] [ ALIGNED      ] }
{   [ item-name-a [version]  ]   [ group-name          ] [ UNALIGNED    ] }
{                                                        [ NOT-ALIGNED  ] }

    [KNOWN-AS local-name]   [INDEXED-BY index-name]...
```

clause is:

```
b [ EQ ]  [ c       ] [ AND ]  b [ EQ ]  [ c       ]...
  [ =  ]  [ literal ] [ OR  ]    [ =  ]  [ literal ]
  [ NE ]                         [ NE ]
  [ GT ]                         [ GT ]
  [ >  ]                         [ >  ]
  [ GE ]                         [ GE ]
  [ LT ]                         [ LT ]
  [ <  ]                         [ <  ]
  [ LE ]                         [ LE ]
```

b is item-name-b [version-b]

c is item-name-c [version-c]

common-clauses are defined in "Common Clauses" on page 395

File Member Type

A FILE member has the form:

```
FILE
[clauses and keywords] [common-clauses]
```

A FILE member definition must commence with the member identifier keyword FILE. The clauses and keywords forming the body of the member definition statement are all optional. They may be specified in any order except the alignment keyword which, if used, must precede the CONTAINS clause.

Form-keyword in FILE Member Definition

To specify the form of all sub-files, groups and items contained within the file, enter one of the following form-keywords:

```
ENTERED-AS
HELD-AS
REPORTED-AS
DEFAULTED-AS
```

This means that the keywords and clauses which follow the form-keyword specify the file as it is entered, held, reported or defaulted.

If the Source Language Generation facility is to be used to generate source language statements from the file, all items within the file should be of the same form. Therefore any groups contained within the file must (actually or by default) have the same form-keyword in their data definition as has this file.

If no form-keyword is specified, it defaults to DEFAULTED-AS. Only one form-keyword may be used in a FILE member definition.

Alignment-keyword in FILE Definition

Alignment-keyword is one of the following:

```
UNALIGNED
NOT-ALIGNED
ALIGNED
```

If no alignment-keyword is specified, the default is to UNALIGNED. UNALIGNED and NOT-ALIGNED are synonymous.

UNALIGNED means that binary items and floating-point items declared (as individual items or as elements of an array) as being contained in the file are not necessarily aligned to word or half word boundaries and that bit string items are not necessarily aligned to byte boundaries.

ALIGNED is the equivalent of COBOL SYNCHRONIZED or PL/I ALIGNED. It means that all binary items and all floating point items declared (as individual items or as elements of an array) as being contained in the file are aligned to half word, full word or double word boundaries, thus:

- Binary items having a length of four decimal digits or less occupy a complete half word,
- Binary items having a length of from five to nine decimal digits occupy a full word,
- Binary items having a length of from ten to eighteen decimal digits occupy two full words, but are not necessarily aligned to a double word boundary,
- Floating-point items having six digits or less in the mantissa occupy a full word,
- Floating-point items having from seven to sixteen digits in the mantissa occupy a double word.

ALIGNED also causes any bit string items to be aligned to byte boundaries.

Where an alignment is being specified for a file and all its components then the alignment-keyword must be declared prior to the CONTAINS clause in the FILE member definition statement. For example:

```
FILE
ENTERED-AS PARTITIONED ALIGNED VOLUME 'VOL041'
;
```

This file alignment may be overridden for the individual components of the file by specifying an alternative alignment-keyword within the FILE member's CONTAINS clause, after the selected component's name and preceding any associated ELSE and IF clauses. Alignment-keywords may also be specified in the same manner for the components of a file for which no alignment-keyword has been declared.

For example:

```
FILE
ENTERED-AS PARTITIONED ALIGNED VOLUME 'V0L041'
CONTAINS
  ITEM1
 ,ITEM2
 ,ITEM3 UNALIGNED
;
```

In the example the FILE and therefore ITEM1 and ITEM2 are aligned, while ITEM3 is not.

The ALIGNED or UNALIGNED keyword does not apply to items contained within groups declared (as individual groups or as elements of an array) as being contained in the file. The data definitions of the groups determine the alignment or non-alignment of such indirectly-referenced items. Similarly, the ALIGNED or UNALIGNED keyword does not apply to items or groups contained within subordinate files declared as being contained in the file. It is therefore not meaningful to include alignment-keywords in a content-declaration that declares a file, a group or an array of groups.

CONTAINS Clause in FILE Definition

The CONTAINS clause is used to declare the contents of the file under a range of different stated conditions. The contents may be an item, a group, a subordinate file or an array.

Up to 16 conditional terms may be specified using IF clauses. Any definition may have an alternative contents which may also be conditional.

Alternative content declarations must be separated by the keyword ELSE. The first conditional term of an alternative content declaration is added to the total number of conditional terms specified in the preceding IF clause.

The CONTAINS clause has the form:

```
CONTAINS content [IF clause] [ELSE content [IF clause]]...
       [,content [IF clause] [ELSE content [IF clause]...]...
```

where content is:

```
{ files-name                                                  }
{ group-name                                                  }
{ item-name [version]                                         }
{ (|integer                  | ) |group-name         |       }
{  |item-name-1 [version]    |   |item-name[version]|       }
{              [alignment-keyword]      [KNOWN-AS            }
```

file-name is the name of a file.

group-name is the name of a group.

item-name is the name of an item.

version is an unsigned integer in the range 1 to 15, being a number specifying which version of the relevant item is relevant to this file. The version is associated with the form (ENTERED-AS, HELD-AS, REPORTED-AS or DEFAULTED-AS) specified for this file. If version is omitted, a default value of 1 is assumed.

integer is an unsigned integer of from 1 to 18, being the number of times *item-name* or *group-name* occurs in the array.

item-name-1 is the name of an item. This form of array declaration declares that when the file here defined is processed by an application program or module, the number of times *item-name* or *group-name* occurs in the array is contained in the item *item-name-1*.

alignment-keyword is UNALIGNED, NOT-ALIGNED or ALIGNED and determines whether or not binary items and floating point items declared as being contained in the file are aligned to half word, full word or double word boundaries. The keyword must precede any IF or ELSE clauses.

local-name is an alternative name for any item, group or subordinate file within the file by which it is to be known when record layouts or source language descriptions are generated from this data definition by the DataManager Source Language Generation facility (selectable unit DMR-SL1). The local-name is not separately recorded in the data dictionary and so cannot be interrogated and can be the same as another name, an alias or a catalog classification. *Local-name* is the name by which the contained member is known only within the file defined by this definition. *Local-name* must conform to the naming rules for member-names.

Refer to "ITEM Member Definition Statement" on page 322 for details of the IF clause and of version numbers.

Refer to *ASG-ControlManager User's Guide* for details of the rules governing member names.

Examples of the Uses of the CONTAINS Clause

You can use the CONTAINS clause in any of these ways:

- To state the name of a GROUP member defining the one type of record that the file contains. For example:

  ```
  FILE CONTAINS
  RECORD-A
  ```

- If the file contains records of more than one type, to list the names of the GROUP members which define the different record types. For example:

  ```
  FILE CONTAINS
  RECORD-A ELSE RECORD-B
  ELSE RECORD-C
  ```

- If a file contains records of more than one type, and also contains an indicator to signify which type of record it contains, to state the name of the member (ITEM or GROUP) which defines the indicator, and to list the names of the GROUP members which define the different record types, together with the indicator to which they are related. It is also possible to give the names of other items or groups that are common to all the record types in the file. For example:

  ```
  FILE CONTAINS
  INDICATOR,
  KEY,
  RECORD-A IF INDICATOR = 1
  ELSE RECORD-B IF INDICATOR = 2
  ELSE RECORD-C
  ```

- To give the names of other files that constitute sub-files of the file being defined.

Organization-keyword(s) in FILE Member Definition

You can use one or more of these keywords to describe the physical organization of the file:

- SEQUENTIAL is used when the position of the records on the file is determined by the order in which they are written to it.
- INDEXED is used when the position of each record in the file is located from an index or indexes created with the file and maintained by the system. Access can be either sequential or random.
- DIRECT is used when records are addressed by their position, either relative to the physical storage medium or relative to the start of the file.
- KEYED is used when records are located approximately by their position, either relative to the physical storage medium or relative to the start of the file, and are finally located by a key held within the record.
- VSAM is used when a file is accessed by IBM's Virtual Sequential Access Method, VSAM. VSAM KEYED is used to define a key-sequenced data set (KSDS). VSAM SEQUENTIAL is used to define an entry-sequenced data set (ESDS). VSAM DIRECT is used to define a relative-record data set (RRDS).
- PARTITIONED is used when a file is accessed by IBM's Basic Partitioned Access Method (BPAM).

For example:

```
FILE
HELD-AS SEQUENTIAL KEYED VOLUME 'VOL041'
```

Labels Clause in FILE Member Definition

The labeling of a file may optionally be declared in the following ways:

```
NO-LABELS
STANDARD-LABELS
USER-LABELS module-name
```

NO-LABELS is used if there are no labels, or you do not want the labels to be processed by standard routines.

STANDARD-LABELS is used if the file is labelled in the standard way.

USER-LABELS is used when the file has non-standard or user labels.

module-name is the name of the module in which the user labels are defined or processed.

These declarations do not apply to header or trailer labels.

Data Blocking Clause in FILE Member Definition

You can declare the way in which data is blocked as one of the following:

```
FIXED [block-size [record-size]]
VARIABLE [maximum-block-size [maximum-record-size]]
UNDEFINED [maximum-block-size]
SPANNED [maximum-block-size [maximum-record-size]]
```

FIXED is used when all records are of the same length and each record is contained within one block.

VARIABLE is used when records are fixed or variable in length, the record length being recorded within each record. Each record is contained within one block, which may contain more than one record.

UNDEFINED is used when there may be only one record of fixed or variable length within each block.

SPANNED is used when records are fixed or variable in length, the record length being recorded within each record. Records may span more than one block, and blocks may contain more than one record.

block-size is an unsigned integer not greater than 32767, being the number of characters per block.

record-size is an unsigned integer not greater than 32767, being the number of characters per record.

maximum-block-size is an unsigned integer not greater than 32767, being the maximum number of characters per block.

maximum-record-size is an unsigned integer not greater than 32767, being the maximum number of characters per record.

SORT-KEY Clause in FILE Member Definition

The SORT-KEY clause declares the names of not more than fifteen items or groups on which the file is ordered, in descending order of significance.

The clause has the form:

```
item-name-a [version]  [ {ASCENDING  } ]
                         {DESCENDING } }
                       [, member-name   [ {ASCENDING  } ] ]]..
                                          {DESCENDING } }
```

member-name is the name of an ITEM or GROUP.

ASCENDING and DESCENDING may be used to indicate whether the file is in ascending or descending order of the key. Both ASCENDING and DESCENDING keywords can appear in the same SORT-KEY clause.

GENERATION-CYCLE Clause in FILE Member Definition

The GENERATION-CYCLE clause states:

- How many generations of the file are retained if a new physical file is generated by each updating run; or
- The number of backup copies retained (plus the current version) if the file is updated in place.

The clause has the form:

```
GENERATION-CYCLE integer
```

where *integer* is an unsigned integer not greater than 32,767.

RETENTION-PERIOD Clause in FILE Member Definition

The RETENTION-PERIOD clause states:

- How long a generation of the file is kept before purging, if a new physical file is generated by each updating run, or
- How long backup copies of the file are kept, if the file is updated in place.

The clause has the form:

```
RETENTION-PERIOD integer   [ ('string') ]
                             { DAYS    }
                             { MONTHS  }
                             { YEARS   }
```

where:

integer is an unsigned integer not greater than 32,767.

string is a character string of not more than 256 printable characters. Spaces (hexadecimal 40) are regarded as printable characters.

GROWTH-RATE Clause in FILE Member Definition

You can record in a GROWTH-RATE clause either the file's anticipated or its historical rate of growth. This may be in terms of a percentage increment or an increase in the number of records or other chosen parameter over a stated time period. The form of the clause is:

```
RETENTION-PERIOD integer [PERCENT] 'string'
                                   { MONTHLY }
                                   { YEARLY  }
```

for example:

```
GROWTH-RATE 1500 YEARLY
```

or:

```
GROWTH-RATE 3 PERCENT 'QUARTERLY'
```

where:

integer is an unsigned integer not greater than 32,767.

string is a character string of not more than 256 printable characters.

DEVICE Clause in FILE Member Definition

The DEVICE clause records the type of peripheral device on which the file is normally processed. The clause has the form:

```
DEVICE  {device  [{number }]}
         {number  {'model'}}
         {'model'           }
```

device is one of these:

AUDIO	DISC	OCR	PRINTER
CARDS	DISK	PAPERTAPE	REMOTE
CONSOLE	DRUM	PAPER-TAPE	TAPE
DASD	MICR	PLOTTER	TERMINAL

number is a manufacturer's type number or model number, with a maximum of 8 digits. No validation of specific numbers is made.

model is a delimited string of 8 or less characters, being a manufacturer's model identifier.

DENSITY Clause in FILE Member Definition

The DENSITY clause states the packing density of the file, in bits per inch or in such other items as may be defined by your controller. The clause has the form:

```
DENSITY integer
```

where *integer* is an integer of not more than 32767.

VOLUME Clause in FILE Member Definition

The VOLUME clause states the name or serial number of the disk or tape volume or other storage medium containing the file. The clause has the form:

```
VOLUME 'name'
```

where *name* is a string of up to 256 characters.

SIZE Clause in FILE Member Definition

You can declare the size of the file with a SIZE clause as follows:

```
SIZE size  [ {KBYTES    } ] [,size [ {KBYTES    } ]]
             {TRACKS    }            {TRACKS    }
             {CYLINDERS }            {CYLINDERS }
             {'string'  }            {'string'  }
```

where:

size is an integer with a maximum value of 2,l47,483,647.

'string' is a delimited string of not more than 256 characters which permits other units of measurement.

If two sizes are declared, they refer to the primary and secondary FILE areas respectively. KBYTES means 1024 bytes.

Example of FILE Member Definition

```
FILE HELD-AS ALIGNED CONTAINS
   RECORD-TYPE, CUSTOMER-NUMBER,
   RETRIEVALS UNALIGNED
   ELSE DEBlT-ENTRIES IF RECORD-TYPE GE 2
   AND RECORD-TYPE LT 6
   ELSE CREDIT-ENTRIES IF RECORD-TYPE GE 6
   VARIABLE 2000 600
   SEQUENTIAL STANDARD
   SORT-KEY CUSTOMER-NUMBER, RECORD-TYPE
   NOTE "FILE CONTAINS VALIDATED TRANSACTIONS"
   "AND RETRIEVED INFORMATION FROM"
   "CUSTOMER FILE AND ARTICLE FILE"
;
```

Group Member Definition Syntax

```
FILE   [ ENTERED-AS    ]  [ ALGNED        ]
       [ HELD-AS       ]  [ UNALIGNED     ]
       [ REPORTED-AS   ]  [ NOT-ALIGNED   ]
       [ DEFAULTED-AS  ]  [               ]

    [CONTAINS content [IF clause] [ELSE content [IF clause]]...
        [, content [IF clause] [ELSE content [IF

    [ SEQUENTIAL    ...]  [ NO-LABELS                ]
    [ INDEXED       ]     [ STANDARD-LABELS          ]
    [ DIRECT        ]     [ USER-LABELS module-name  ]
    [ KEYED         ]
    [ VSAM          ]
    [ PARTITIONED   ]

    [ FIXED [blocked-size [record-size]]                    ]
    [ VARIABLE [maximum-block-size [maximum-record-size]]   ]
    [ UNDEFINED [maximum-block-size]                        ]
    [ SPANNED [maximum-block-size [maximum-record-size]]    ]

    [SORT-KEY  [ item-name  ]   [ ASCENDING  ]
               [ group-name ]   [ DESCENDING ]

        [,  [ item-name  ]  [ ASCENDING  ]...]
            [ group-name ]  [ DESCENDING ]

    [GENERATION-CYCLE integer]

    [RETENTION-PERIOD integer   [ 'string' ]]
                                [ DAYS     ]
                                [ MONTHS   ]
                                [ YEARS    ]

    [GROWTH-RATE integer [PERCENT] [ 'string' ]
                                   [ MONTHLY  ]
                                   [ YEARLY   ]

    [DEVICE [device    [ number ] ]  ]  [DENSITY integer
            [          [ 'model' ]  ]
            [number                 ]
            ['model'                ]

    [VOLUME 'name'

    SIZE size  [ KBYTES    ]  [,size [ KBYTES    ]]]
               [ TRACKS    ]         [ TRACKS    ]
               [ CYLINDERS ]         [ CYLINDERS ]
               [ 'string'  ]         [ 'string'  ]

    [common-clauses]

    [ ; ]
    [ . ]
```

where:

content is:

```
{file-name
 group-name
 item-name [version]
 (integer            ) {item-name [version]}
  {item-name-a [version]}{group-name        }

[KNOWN-AS local-name] [{ALIGNED    }]
                       {UNALIGNED  }
                       {NOT-ALIGNED}
```

clause is:

```
b {EQ}  {c       }[{AND}  b {EQ}   {c       }]...
  {=  }  {literal }{{OR }      {=  }  {literal }
  {NE }                        {NE }
  {GT }                        {GT }
  {>  }                        {>  }
  {GE }                        {GE }
  {LT }                        {LT }
  {<  }                        {<  }
  {LE }                        {LE }
```

b is item-name-b [version-b]

c is item-name-c [version-c]

common-clauses are defined in section 5.2

FILE Member Definition

A FILE member has the form:

```
FILE
[clauses and keywords] [common-clauses]
;
```

A FILE member definition must commence with the member identifier keyword FILE. The clauses and keywords forming the body of the member definition statement are all optional. They may be specified in any order except the alignment keyword which, if used, must precede the CONTAINS clause.

Form-keyword in FILE Member Definition

To specify the form of all sub-files, groups and items contained within the file, enter one of the following form-keywords:

```
ENTERED-AS
HELD-AS
REPORTED-AS
DEFAULTED-AS
```

This means that the keywords and clauses which follow the form-keyword specify the file as it is entered, held, reported or defaulted.

If the Source Language Generation facility is to be used to generate source language statements from the file, all items within the file should be of the same form. Therefore any groups contained within the file must (actually or by default) have the same form-keyword in their data definition as has this file.

If no form-keyword is specified, it defaults to DEFAULTED-AS. Only one form-keyword may be used in a FILE member definition.

Alignment-keyword in FILE Definition

Alignment-keyword is one of the following:

```
UNALIGNED
NOT-ALIGNED
ALIGNED
```

If no alignment-keyword is specified, the default is to UNALIGNED. UNALIGNED and NOT-ALIGNED are synonymous.

UNALIGNED. Binary items and floating-point items declared (as individual items or as elements of an array) as being contained in the file are not necessarily aligned to word or half word boundaries and that bit string items are not necessarily aligned to byte boundaries.

ALIGNED. The equivalent of COBOL SYNCHRONIZED or PL/I ALIGNED. It means that all binary items and all floating point items declared (as individual items or as elements of an array) as being contained in the file are aligned to half word, full word or double word boundaries, thus:

- Binary items having a length of four decimal digits or less occupy a complete half word
- Binary items having a length of from five to nine decimal digits occupy a full word
- Binary items having a length of from ten to eighteen decimal digits occupy two full words, but are not necessarily aligned to a double word boundary
- Floating-point items having six digits or less in the mantissa occupy a full word
- Floating-point items having from seven to sixteen digits in the mantissa occupy a double word

ALIGNED also causes any bit string items to be aligned to byte boundaries.

Where an alignment is being specified for a file and all its components then the alignment-keyword must be declared prior to the CONTAINS clause in the FILE member definition statement. For example:

```
FILE
ENTERED-AS PARTITIONED ALIGNED VOLUME 'VOL041'
```

This file alignment may be overridden for the individual components of the file by specifying an alternative alignment-keyword within the FILE member's CONTAINS clause, after the selected component's name and preceding any associated ELSE and IF clauses. Alignment-keywords may also be specified in the same manner for the components of a file for which no alignment-keyword has been declared.

For example:

```
FILE
ENTERED-AS PARTITIONED ALIGNED VOLUME ’V0L041’
CONTAINS
  ITEM1
 ,ITEM2
 ,ITEM3 UNALIGNED
;
```

In the example the FILE and therefore ITEM1 and ITEM2 are aligned, while ITEM3 is not.

The ALIGNED or UNALIGNED keyword does not apply to items contained within groups declared (as individual groups or as elements of an array) as being contained in the file. The data definitions of the groups determine the alignment or non-alignment of such indirectly-referenced items. Similarly, the ALIGNED or UNALIGNED keyword does not apply to items or groups contained within subordinate files declared as being contained in the file. It is therefore not meaningful to include alignment-keywords in a content-declaration that declares a file, a group or an array of groups.

CONTAINS Clause in FILE Definition

The CONTAINS clause is used to declare the contents of the file under a range of different stated conditions. The contents may be an item, a group, a subordinate file or an array.

Up to 16 conditional terms may be specified using IF clauses. Any definition may have an alternative contents which may also be conditional.

Alternative content declarations must be separated by the keyword ELSE. The first conditional term of an alternative content declaration is added to the total number of conditional terms specified in the preceding IF clause.

The CONTAINS clause has the form:

```
CONTAINS content [IF clause] [ELSE content [IF clause]]...
       [,content [IF clause] [ELSE content [IF clause]...]...
```

where `content` is:

```
{ file-name                                                  }
{ group-name                                                 }
{ item-name [version]                                        }
{  (  {integer               }  )  {group-name          }   }
{     {item-name-1 [version] }     {item-name[version] }   }
                   [alignment-keyword] [KNOWN-AS local-name]
```

`file-name` is the name of a file.

`group-name` is the name of a group.

`item-name` is the name of an item.

`version` is an unsigned integer in the range 1 to 15, being a number specifying which version of the relevant item is relevant to this file. The version is associated with the form (ENTERED-AS, HELD-AS, REPORTED-AS or DEFAULTED-AS) specified for this file. If version is omitted, a default value of 1 is assumed.

integer is an unsigned integer of from 1 to 18, being the number of times *item-name* or group-name occurs in the array.

item-name-1 is the name of an item. This form of array declaration declares that when the file here defined is processed by an application program or module, the number of times item-name or *group-name* occurs in the array is contained in the item *item-name-1*.

alignment-keyword is UNALIGNED, NOT-ALIGNED or ALIGNED and determines whether or not binary items and floating point items declared as being contained in the file are aligned to half word, full word or double word boundaries. The keyword must precede any IF or ELSE clauses.

local-name is an alternative name for any item, group or subordinate file within the file by which it is to be known when record layouts or source language descriptions are generated from this data definition by the DataManager Source Language Generation facility (selectable unit DMR-SL1). The local-name is not separately recorded in the data dictionary and so cannot be interrogated and can be the same as another name, an alias or a catalog classification. *Local-name* is the name by which the contained member is known only within the file defined by this definition. *Local-name* must conform to the naming rules for member-names.

Refer to "Alignment-keyword in FILE Definition" on page 369 for details of alignment keywords.

Refer to "The IF Keyword" on page 343 for details of the IF clause and of version numbers.

Refer to the *ASG-ControlManager User's Guide* for details of the rules governing member names.

Examples of the Uses of the CONTAINS Clause

You can use the CONTAINS clause in any of these ways:

- To state the name of a GROUP member defining the one type of record that the file contains. For example:

```
FILE CONTAINS
   RECORD-A
;
```

- If the file contains records of more than one type, to list the names of the GROUP members which define the different record types. For example:

```
FILE CONTAINS
   RECORD-A ELSE RECORD-B
   ELSE RECORD-C
;
```

- If a file contains records of more than one type, and also contains an indicator to signify which type of record it contains, to state the name of the member (ITEM or GROUP) which defines the indicator, and to list the names of the GROUP members which define the different record types, together with the indicator to which they are related. It is also possible to give the names of other items or groups that are common to all the record types in the file. For example:

```
FILE CONTAINS
   INDICATOR,
   KEY,
   RECORD-A IF INDICATOR = 1
   ELSE RECORD-B IF INDICATOR = 2
   ELSE RECORD-C
;
```

- To give the names of other files that constitute sub-files of the file being defined.

Organization-keyword(s) in FILE Member Definition

You can use one or more of these keywords to describe the physical organization of the file:

SEQUENTIAL. Is used when the position of the records on the file is determined by the order in which they are written to it.

INDEXED. Used when the position of each record in the file is located from an index or indexes created with the file and maintained by the system. Access can be either sequential or random.

DIRECT. Used when records are addressed by their position, either relative to the physical storage medium or relative to the start of the file.

KEYED. Used when records are located approximately by their position, either relative to the physical storage medium or relative to the start of the file, and are finally located by a key held within the record.

VSAM. Used when a file is accessed by IBM's Virtual Sequential Access Method, VSAM. VSAM KEYED is used to define a key-sequenced data set (KSDS). VSAM SEQUENTIAL is used to define an entry-sequenced data set (ESDS). VSAM DIRECT is used to define a relative-record data set (RRDS).

PARTITIONED. Used when a file is accessed by IBM's Basic Partitioned Access Method (BPAM).

For example:

```
FILE
HELD-AS SEQUENTIAL KEYED VOLUME 'VOL041'
;
```

Labels Clause in FILE Member Definition

The labeling of a file may optionally be declared in the following ways:

```
NO-LABELS
STANDARD-LABELS
USER-LABELS module-name
```

where:

NO-LABELS is used if there are no labels, or you do not want the labels to be processed by standard routines.

STANDARD-LABELS is used if the file is labelled in the standard way.

USER-LABELS is used when the file has non-standard or user labels.

module-name is the name of the module in which the user labels are defined or processed.

These declarations do not apply to header or trailer labels.

Data Blocking Clause in FILE Member Definition

You can declare the way in which data is blocked as one of the following:

```
FIXED [block-size [record-size]]
VARIABLE [maximum-block-size [maximum-record-size]]
UNDEFINED [maximum-block-size]
SPANNED [maximum-block-size [maximum-record-size]]
```

FIXED is used when all records are of the same length and each record is contained within one block.

VARIABLE is used when records are fixed or variable in length, the record length being recorded within each record. Each record is contained within one block, which may contain more than one record.

UNDEFINED is used when there may be only one record of fixed or variable length within each block.

SPANNED is used when records are fixed or variable in length, the record length being recorded within each record. Records may span more than one block, and blocks may contain more than one record.

block-size is an unsigned integer not greater than 32767, being the number of characters per block.

record-size is an unsigned integer not greater than 32767, being the number of characters per record.

maximum-block-size is an unsigned integer not greater than 32767, being the maximum number of characters per block.

maximum-record-size is an unsigned integer not greater than 32767, being the maximum number of characters per record.

SORT-KEY Clause in FILE Member Definition

The SORT-KEY clause declares the names of not more than fifteen items or groups on which the file is ordered, in descending order of significance.

The clause has the form:

```
SORT-KEY member-name [ {ASCENDING } ]
                       {DESCENDING}
                     [, member-name   [ {ASCENDING } ]]...
                                        {DESCENDING}
```

member-name is the name of an ITEM or GROUP.

ASCENDING and DESCENDING may be used to indicate whether the file is in ascending or descending order of the key. Both ASCENDING and DESCENDING keywords can appear in the same SORT-KEY clause.

GENERATION-CYCLE Clause in FILE Member Definition

The GENERATION-CYCLE clause states:

- How many generations of the file are retained if a new physical file is generated by each updating run; or
- The number of backup copies retained (plus the current version) if the file is updated in place.

The clause has the form:

```
GENERATION-CYCLE integer
```

integer is an unsigned integer not greater than 32,767.

RETENTION-PERIOD Clause in FILE Member Definition

The RETENTION-PERIOD clause states:

- How long a generation of the file is kept before purging, if a new physical file is generated by each updating run, or
- How long backup copies of the file are kept, if the file is updated in place.

The clause has the form:

```
RETENTION-PERIOD integer [ { 'string'  } ]
                           { DAYS     }
                           { MONTHS   }
                           { YEARS    }
```

integer is an unsigned integer not greater than 32,767.

string is a character string of not more than 256 printable characters. Spaces (hexadecimal 40) are regarded as printable characters.

GROWTH-RATE Clause in FILE Member Definition

You can record in a GROWTH-RATE clause either the file's anticipated or its historical rate of growth. This may be in terms of a percentage increment or an increase in the number of records or other chosen parameter over a stated time period.

The form of the clause is:

```
GROWTH-RATE integer [PERCENT]    { 'string' }
                                 { DAYS     }
                                 { MONTHS   }
                                 { YEARS    }
```

for example:

```
GROWTH-RATE 1500 YEARLY
```

or:

```
GROWTH-RATE 3 PERCENT 'QUARTERLY'
```

integer is an unsigned integer not greater than 32,767.

string is a character string of not more than 256 printable characters.

DEVICE Clause in FILE Member Definition

The DEVICE clause records the type of peripheral device on which the file is normally processed.

The clause has the form:

```
DEVICE { device  [ { number  } ] }
       { number    { 'model' }   }
       { 'model'                 }
```

where:

device is one of these:

```
AUDIO    DISC     OCR          PRINTER
CARDS    DISK     PAPERTAPE    REMOTE
CONSOLE  DRUM     PAPER-TAPE   TAPE
DASD     MICR     PLOTTER      TERMINAL
```

number is a manufacturer's type number or model number, with a maximum of 8 digits. No validation of specific numbers is made.

model is a delimited string of 8 or less characters, being a manufacturer's model identifier.

DENSITY Clause in FILE Member Definition

The DENSITY clause states the packing density of the file, in bits per inch or in such other items as may be defined by your controller. The clause has the form:

```
DENSITY integer
```

where *integer* is an integer of not more than 32767.

VOLUME Clause in FILE Member Definition

The VOLUME clause states the name or serial number of the disk or tape volume or other storage medium containing the file. The clause has the form:

```
VOLUME 'name'
```

where *name* is a string of up to 256 characters.

SIZE Clause in FILE Member Definition

You can declare the size of the file with a SIZE clause as follows:

```
SIZE size [ KBYTES     ]  [,size [ KBYTES     ]]
          { TRACKS     }          { TRACKS     }
          { CYLINDERS  }          { CYLINDERS  }
          { 'string'   }          { 'string'   }
```

where:

size is an integer with a maximum value of 2,l47,483,647.

'string' is a delimited string of not more than 256 characters which permits other units of measurement.

If two sizes are declared, they refer to the primary and secondary FILE areas respectively. KBYTES means 1024 bytes.

Example of FILE Member Definition

```
FILE HELD-AS ALIGNED CONTAINS
   RECORD-TYPE, CUSTOMER-NUMBER,
   RETRIEVALS UNALIGNED
   ELSE DEBlT-ENTRIES IF RECORD-TYPE GE 2
   AND RECORD-TYPE LT 6
   ELSE CREDIT-ENTRIES IF RECORD-TYPE GE 6
   VARIABLE 2000 600
   SEQUENTIAL STANDARD
   SORT-KEY CUSTOMER-NUMBER, RECORD-TYPE
   NOTE "FILE CONTAINS VALIDATED TRANSACTIONS"
   "AND RETRIEVED INFORMATION FROM"
   "CUSTOMER FILE AND ARTICLE FILE"
;
```

FILE Member Definition Syntax

```
FILE   [ ENTERED-AS    ]  [ ALGNED        ]
       { HELD-AS       }  { UNALIGNED     }
       { REPORTED-AS   }  { NOT-ALIGNED   }
       { DEFAULTED-AS  }  {               }

    [CONTAINS content [IF clause] [ELSE content [IF clause]]...
        [, content [IF clause] [ELSE content [IF

    [ SEQUENTIAL    ...]  [ NO-LABELS                ]
    { INDEXED       }     { STANDARD-LABELS          }
    { DIRECT        }     { USER-LABELS module-name  }
    { KEYED         }
    { VSAM          }
    { PARTITIONED   }

    [ FIXED [blocked-size [record-size]]                    ]
    { VARIABLE [maximum-block-size [maximum-record-size]]   }
    { UNDEFINED [maximum-block-size]                        }
    { SPANNED [maximum-block-size [maximum-record-size]]    }

    [SORT-KEY { item-name  }   [ ASCENDING  ]
              { group-name }   { DESCENDING }

        [,  { item-name  }  [ ASCENDING  ]...]
            { group-name }  { DESCENDING }

    [GENERATION-CYCLE integer]

    [RETENTION-PERIOD integer   [ 'string' ]]
                                { DAYS     }
                                { MONTHS   }
                                { YEARS    }

    [GROWTH-RATE integer [PERCENT]  { 'string' }]
                                    { MONTHLY  }
                                    { YEARLY   }

    [DEVICE { device    [ number ] ]  [DENSITY integer
            {           [ 'model ] }
            { number                }
            { 'model'               }

    [VOLUME 'name'

    SIZE size  [ KBYTES    ]  [,size [ KBYTES    ]]]
               { TRACKS    }         { TRACKS    }
               { CYLINDERS }         { CYLINDERS }
               { 'string'  }         { 'string'  }

   [common-clauses]

   { ; }
   { . }
```

where:

content is:

```
{file-name
 group-name
 item-name [version]
 (integer            ) {item-name [version]}
  {item-name-a [version]} {group-name        }

[KNOWN-AS local-name] [{ALIGNED    }]
                       {UNALIGNED  }
                       {NOT-ALIGNED}
```

clause is:

```
b {EQ}  {c       }[{AND}  b {EQ}   {c       }]...
  {=  }  {literal }{OR }    {=  }  {literal }
  {NE }                     {NE }
  {GT }                     {GT }
  {>  }                     {>  }
  {GE }                     {GE }
  {LT }                     {LT }
  {<  }                     {<  }
  {LE }                     {LE }
```

b is item-name-b [version-b]

c is item-name-c [version-c]

common-clauses are defined in "Common Clauses" on page 395

SYSTEM, PROGRAM and MODULE Member Definition

To define a SYSTEM, PROGRAM or MODULE, enter:

```
{SYSTEM  } process-clauses] [common-clauses]
{PROGRAM }
{MODULE  }
{;       }
```

A process member definition must commence with the relevant member identifier keyword, either SYSTEM, PROGRAM or MODULE. The clauses and keywords forming the body of the member definition statement are all optional and with the exception of the subclauses of the CALLS and ENTRY-POINT clauses may be specified in any order.

CONTAINS, INPUTS, OUTPUTS and UPDATES Clauses

The CONTAINS, INPUTS, OUTPUTS and UPDATES clauses, which are all optional, have the same form:

```
CONTAINS member-name [,member-name]...
INPUTS member-name [,member-name]...
OUTPUTS member-name [,member-name]...
UPDATES member-name [,member-name]...
```

The CONTAINS clause declares member-names at the same or lower level in the member type hierarchy which form part of the system, program or module being defined.

The INPUTS, OUTPUTS, and UPDATES clauses are used to declare the database, FILE, GROUP or ITEM member-names which are respectively input to, output from, or updated in the original by the process being defined. When not updating the original, the member-name involved should be declared in both an INPUTS and an OUTPUTS clause.

Example:

```
PROGRAM CONTAINS CON11A2, CON11A22
INPUTS      CONINP
UPDATES BRGFIL
OUTPUTS CONLIST, BRGLIST
```

Refer to "Member Types which Document Data and the Processes that Act Upon it" on page 320 for details of the member type hierarchy.

CALLS Clause

The CALLS clause declares the names of members called by a SYSTEM, PROGRAM or MODULE.

The clause has the form:

```
CALLS process-name [,process-name]...
```

or:

```
CALLS clause [CALLS clause]...
```

where:

clause is:

```
process-name [AT 'label']
                    [PASSING data-name [,data-name]...]
```

process-name is the name of a process member that:

- In a SYSTEM data definition, can be a MODULE, a PROGRAM or a SYSTEM
- In a PROGRAM data definition, can be a MODULE or a PROGRAM
- In a MODULE data definition, must be a MODULE.

label declares the label of the entry-point of the system, program or module being defined.

data-name is the name of a FILE, GROUP or ITEM member.

The PASSING sub-clause declares the names of members defining parameters that are passed to the process member being defined.

The CALLS keyword must not be separated from a subordinate AT or PASSING clause by any other clause (such as a common clause) allowed in the process member definition.

Example:

```
PROGRAM
CONTAINS P1U02
CALLS P1101 PASSING PM1, PM2, PM3, PM4
ALIAS         'UPDATE-EMPLOYEE-MASTER-FILE'
;
```

PARAMETERS and ENTRY-POINT Clauses

The PARAMETERS clause declares the names of members defining parameters that are passed to the SYSTEM, PROGRAM or MODULE being defined. It has the form:

```
PARAMETERS data-name [, data-name]...
```

The ENTRY-POINT clause declares the labels of an entry point of the SYSTEM, PROGRAM or MODULE being defined, and may contain PARAMETERS sub-clauses. It has the form:

```
ENTRY-POINT clause  [, ENTRY-POINT clause]...
```

No member may contain both PARAMETERS and ENTRY-POINT clauses.

where:

clause is:

```
'label' [PARAMETERS data-name [,data-name]...]
```

data-name is the name of a FILE, GROUP or ITEM member.

label declares the label of the entry-point of the process.

The ENTRY-POINT keyword must not be separated from its subordinate clause by any other clause allowed in the SYSTEM (for example, common clauses).

Example:

```
MODULE
PARAMETERS UNIT, QUANTITY
UPDATES FLAG-2
LANGUAGE 'COBOL'
;
```

LANGUAGE, AUTHOR, INSTALLATION, DATE-WRITTEN, SOURCE-COMPUTER, OBJECT -COMPUTER, SPECIAL-NAMES, I-O-CONTROL, ASSIGNMENT, EDIT-INPUT, EDIT-OUTPUT and EDIT-UPDATE Clauses

These clauses are all optional. Except for the LANGUAGE clause, they are used by SOURCEMANAGER to generate COBOL source programs. The EDIT-INPUT, EDIT-OUTPUT, and EDIT-UPDATE clauses are intended to contain PRODUCE commands for Source Language Generation. The other clauses (in the absence of SOURCEMANAGER) are useful for documentation.

The clauses have these forms:

```
LANGUAGE ’String’SPECIAL-NAMES ’text’
AUTHOR ’text’I-O-CONTROL ’text’
INSTALLATION’text’ASSIGNMENT ’text’
DATE-HRITTEN ’date’EDIT-INPUT’text’
SOURCE-COMPUTER ’text’EDIT-OUTPUT ’text’
OBJECT-COMPUTER ’text’EDIT-UPDATE ’text’
```

where:

string is a character string, *text* is a sequence of delimited character strings.

date must be in the format which applies in your installation, for example, ’12 DEC 1986’ or 12/12/86. If the separator between the elements of the date is a space, period, semi-colon, comma or right bracket, date must be delimited.

The LANGUAGE clause declares the name of the computer language in which the member is programmed. No validation is performed by DataManager on the string declared in the LANGUAGE clause.

The clauses:

- EDIT-INPUT
- EDIT-OUTPUT
- EDIT-UPDATE

relate respectively to the INPUTS, OUTPUTS and UPDATES clauses. They are intended to contain Source Language Generation PRODUCE command generation-control-options and/or name-editing-options. SourceManager generates internal PRODUCE commands embodying these options in respect of the data-names specified in the relevant INPUTS, OUTPUTS or UPDATES clause. The PRODUCE commands are syntax checked by DataManager when activated; the text strings of the EDIT-INPUT, EDIT-OUTPUT and EDIT-UPDATE clauses are not validated as PRODUCE command elements when the member is encoded.

When the member is processed by a REPORT or GLOSSARY command, the text strings of the process clauses detailed in this panel (excluding the LANGUAGE and DATE-WRITTEN clauses) are not concatenated, but, within each reported or glossaried clause, are output aligned one beneath the other on their opening delimiters (irrespective of the number of strings on an input line).

Example

```
PROGRAM
DESCRIPTION 'VALIDATES EMPLOYEE CODE NUMBERS'
INPUTS P1U02MEMP, FORM09
OUTPUTS P1V01WAL
LANGUAGE 'SOURCEMANAGER COBOL'
CATALOG 'ON-LINE'
'VALIDATION'
AUTHOR 'T WATSON'
INSTALLATION 'MSP - LONDON'
DATE-WRITTEN '1SEP81'
SECURITY-CLASSIFICATION 'AUTHORIZATION NEEDED'
SOURCE-COMPUTER 'IBM-370'
OBJECT-COMPUTER 'IBM-370'
SPECIAL-NAMES
'DECIMAL POINT IS COMMA'
I-O-CONTROL
'APPLY WRITE-ONLY ON P1V01WAL'
ASSIGNMENT
'SELECT P1U02MEMP'
'ASSIGN TO SYS001-DA-2314-D-PMEMP'
EDIT-INPUT
"DROP ALL WHEN ANY EQ 'DUM'"
```

Refer to "CONTAINS, INPUTS, OUTPUTS and UPDATES Clauses" on page 381 for details of the INPUT, OUTPUT and UPDATE clauses.

Examples of SYSTEM, PROGRAM and MODULE Member Definition

Example 1

```
SYSTEM
INPUTS EMPLOYEE-TRANSACTIONS-FORM
OUTPUTS UPDATE-CONTROL-REPORT, EMPLOYEE-REPORT
CONTAINS P1V01,P1S01,P1U01,P1P01
ALIAS       'EMPLOYEE-RECORDS-SYSTEM'
CATALOGUE 'PERSONNEL'
DESCRIPTION
'THIS SYSTEM MAINTAINS PRODUCTION DIVISION'
'EMPLOYEE RECORDS'
NOTE ' THIS SYSTEM IS MADE UP OF TWO JOBS'
     ' JOB 1 IS RUN IN ON-LINE MODE DAILY'
     ' JOB 2 IS RUN IN BATCH MODE AT THE END OF A WEEK'
```

Example 2

```
PROGRAM CONTAINS READMOD, VETMOD, PROC1, PROC2, WRITEMOD
.
```

Example 3

```
SYSTEM CONTAINS REQVET, REQACC,
REQCLC, MATMIS, ORDRET, REQRPT
.
```

Example 4

```
SYSTEM NOTE "MONTHLY PAYROLL"
CONTAINS PAYCALCM
CATALOGUE 'PERSONNEL'
;
```

Example 5

```
MODULE INPUTS INP1, INP2, ARTFILE
CALLS KEYVET, LOGSCR
UPDATES RECCNT
OUTPUTS KEY, DATA, ARTDESC
.
```

SYSTEM, PROGRAM, and MODULE Member Definition Syntax

```
{SYSTEM  }
{PROGRAM }
{MODULE  }
   [CONTAINS member-name [, member-name]...]
   [INPUTS data-name [, data-name]...]
   [OUTPUTS data-name [, data-name]...]
   [UPDATES data-name [, data-name]...]
   [{CALLS process-name [, process-name]... }]
    {calls-clause-2...                       }
   [{PARAMETERS data-name [, data-name]...      }]
    {entry-point-clause [entry-point-clause]...}
   [LANGUAGE 'string'] [AUTHOR 'text'] [INSTALLATION 'text']
   [DATA-WRITTEN date] [SOURCE-COMPUTER 'text'
   [OBJECT-COMPUTER 'text']
   [SPECIAL-NAMES 'text']
   [I-O-CONTROL 'text'] [ASSIGNMENT 'text']
   [EDIT-INPUT 'text'] [EDIT-OUTPUT 'text'] [EDIT-UPDAE 'text']
   [common-clauses]

;
.
```

where:

member-name identifies a member at the same level or a lower level in the member type hierarchy.

process-name identifies a process member at the same level or a lower level in the member type hierarchy.

data-name identifies a data member.

calls-clause-2 is:

```
CALLS process-name [AT  'string']
    [PASSING data-name [, data-name]...]
```

entry-point-clause is:

```
ENTRY-POINT  'string' [PARAMETERS data-name [, data-name]...]
```

common-clauses are defined in "Common Clauses" on page 395.

COMMAND-STREAM

COMMAND-STREAM Member Definition

The purpose of COMMAND-STREAM members is to store a sequence of commands which may subsequently be run using the PERFORM command.

A COMMAND-STREAM member has the form:

```
COMMAND-STREAM
[common clauses]
CONTENTS
input-line ...
```

common clauses must not precede the COMMAND-STREAM member type identifier keyword or appear within the CONTENTS clause.

input-line is any Manager Products command.

Commands must include terminators. Where the input-line would contain a terminator in the first character position of the line then an escape-character must be entered in its place and the terminator moved to the second character position. Escape-characters are not required in fully interactive environments. The escape-character has a default setting of %.

An asterisk * or other perform-character may appear in place of a member name, alias, catalog classification or user defined indexed attribute. In circumstances where the index-name would have required delimiters the perform-character representing it must also be delimited.

Refer to "PERFORM (DMR-DD1)" on page 195 for details of the PERFORM command.

Refer to "Common Clauses" on page 395 for details of common clauses.

Refer to *ASG-ControlManager User's Guide* for details of the escape-character and of the perform-character.

COMMAND-STREAM Examples

Example 1

```
COMMAND-STREAM
DESCRIPTION 'TO CHANGE CATALOG CLASSIFICATION ON'
            'ACCEPTANCE OF A DEFINITION AFTER TESTING'
CONTENTS
ALTER *;
LOCATE 'CATALOG'
LOCATE NEXT 'TRIAL'
CHANGE LINE 'TRIAL' TO 'ACCEPTED'
```

This example illustrates the coding of an amendment terminator within a COMMAND-STREAM member. If a member definition statement terminator were included in the stored input-lines, it would be coded in the same way. This example assumes an installation standard whereby the CATALOG[UE] keyword is on the line before that containing the classifications.

Example 2

```
COMMAND-STREAM
ADMINISTRATIVE-DATA
   'TO BE RUN AT 09.00 ON FIRST WORKING DAY OF EACH MONTH'
CONTENTS
KEEP WHAT FORMS 'PROJECTX';
SWITCH ON NAME-CONCATENATION
SELECT NAME, AL1AS 1, DESCRIPTION LINE 5
SEPARATOR - ;
SWITCH OFF MESSAGES LEVEL W;
SKIP;
TEXT "MONTHLY REPORT ON PROJECT X, DATE AS ABOVE";
LIST KEPT-DATA;
BULK REPORT KEPT-DATA;
SWITCH OFF NAME-CONCATENATION;
SWITCH ON MESSAGES;
```

If the above member was entered into a data dictionary by the command:

```
ADD X-REP;
```

it could be activated at any time by the command:

```
PERFORM COMMAND-STREAM X-REP;
```

Example 3

```
COMMAND-STREAM
NOTE 'PRODUCE RECORD-LAYOUTS COMMAND TO BE USED FOR'
     'ANY MEMBERS CATALOGUED UNDER "SYS3"'
CATALOGUE 'COBOL', 'SYS3'
CONTENTS
PRODUCE RECORD-LAYOUTS FOR COBOL FROM *
WITH-ALIAS COBOL GIVING 2 NOTES;
;
```

If the above member was entered into a data dictionary by the command:

```
ADD PRODREC3;
```

it could be activated at any time by the commands:

```
KEEP WHAT FORMS SYS3;
DROP LIST COMMAND-STREAMS;
PERFORM COMMAND-STREAM PRODREC3 KEPT-DATA;
```

Those members that are catalogued under SYS3 could include a SEE PRODREC3 FOR PRODUCE RECORD-LAYOUTS common clause in order to point the user towards this member when generating record layouts.

COMMAND-STREAM Syntax

```
COMMAND-STREAM

    [common-clause]

    CONTENTS

    input-line...

{;}
{.}
```

where:

input-line is any valid input line (except a PERFORM command, unless the ControlManager User Defined Commands facility is installed). The input line may contain an asterisk (or other character determined by a SET PERFORM-CHARACTER command) where a member-name, alias, catalog, classification, or (if the User Defined Syntax facility is installed) a user-defined indexed attribute would normally occur. If the input line would normally be a terminator in the first character position, that terminator must be placed in the second character position, with % (or other character determined by a SET ESCAPE-CHARACTER command) in the first character position.

common-clauses are as defined in "Common Clauses" on page 395

OBSOLETE-DEFINITION Member Type

The OBSOLETE-DEFINITION member type enables you to make members that have become obsolete invisible from the current status and its dependent statuses. This is done simply by changing the member's member type to OBSOLETE-DEFINITION.

This will be useful in situations where the member cannot be deleted because of its presence in other statuses or where a member is currently obsolete but may be required for future use/reference.

OBSOLETE-DEFINITION members are ignored by all interrogation commands unless you deliberately include the keyword OBSOLETE-DEFINITIONs or you use the INDEX-NAMES or the MEMBERS member-name-list selections.

You can convert a member into an OBSOLETE-DEFINITION by placing the identifying keyword OBSOLETE-DEFINITION, and optionally ARCHIVE, in front of its existing definition.

A member's original definition may, optionally, be retained intact within the ARCHIVE clause. ARCHIVE is a free form text attribute. Anything contained within it is regarded by ControlManager as documentation only. Consequently, the original definition may be retained with its references to other members but those references are not active.

If ARCHIVE is not used, then the original definition must be deleted in order that the member may be successfully encoded as an OBSOLETE-DEFINITION.

A member must be truly obsolete before you can successfully encode it as an OBSOLETE-DEFINITION. That is, it must not be referred to by any other members. If it is, then those references must be deleted or further consideration given to whether the member is in fact obsolete. However, the member to be made obsolete may be referred to by other OBSOLETE-DEFINITIONs via a SEE clause.

References from the member to be made obsolete to other members may be dealt with in these ways:

- Retained via a SEE clause in the OBSOLETE-DEFINITION statement
- Deleted
- Retained as documentation only within the ARCHIVE clause

The SEE clause is the only clause by which an OBSOLETE-DEFINITION may refer to other dictionary members but other member types cannot refer to OBSOLETE-DEFINITIONs via a SEE or any other type of clause.

Note:

The member will be invisible from the current status even if it also exists in the direct or indirect base of the current status. The OBSOLETE-DEFINITION in the current status overlays any definitions in the direct or indirect base statuses.

Creating an OBSOLETE-DEFINITION

To change a member to an OBSOLETE-DEFINITION you must update the member's definition as:

- Preface the existing definition with the keyword OBSOLETE-DEFINITION
- Optionally include any common clauses required
- Optionally include the ARCHIVE keyword immediately before the original definition
- Optionally delete the original definition

and then encode the member. An OBSOLETE-DEFINITION statement has this format:

```
OBSOLETE-DEFINITION
[common-clauses]
[ARCHIVE obsolete-definition]
{;}
{.}
```

obsolete-definition is the original definition of the obsolete member, stored as free form text within the ARCHIVE clause.

When the member is encoded, its syntax is checked until the keyword ARCHIVE is encountered. Once the ARCHIVE keyword has been read, ControlManager looks only for a terminator in the first character position of a line, signifying the end of the member. (Only one terminator is required. If the original definition is retained within an ARCHIVE clause then its terminator will serve as the terminator for the OBSOLETE-DEFINITION statement.)

Note:

Common clauses may appear both before and within the ARCHIVE clause. However, in the latter case they are not available for processing. For example, if you create a KEPT-DATA list containing OBSOLETE-DEFINITION members and then enter:

```
GLOSSARY KEPT GIVING NOTES;
```

the output will include any NOTES clauses in those OBSOLETE-DEFINITIONs, except for those within an ARCHIVE clause.

To bring an OBSOLETE-DEFINITION back into normal use, delete:

- The OBSOLETE-DEFINITION keyword
- Any common clauses added when you created the OBSOLETE-DEFINITION member
- The ARCHIVE keyword

and encode (FILE/ENCODE/MODIFY) the amended definition.

An OBSOLETE-DEFINITION may not be successfully encoded if it is referred to by other members other than OBSOLETE-DEFINITIONs. OBSOLETE-DEFINITIONs may refer to any member via the SEE clause.

Processing OBSOLETE-DEFINITIONs

OBSOLETE-DEFINITION members are not processed by any of these commands:

WHAT and WHICH
LIST, GLOSSARY, BULK PRINT and BULK REPORT
KEEP, ALSO KEEP and DROP
PERFORM

unless they include either:

- index-name-type selection OBSOLETE-DEFINITIONS
- INDEX-NAMES selection keyword
- MEMBERS member-name-list selection,

in which case OBSOLETE-DEFINITION members will be selected if they meet other criteria specified in the command.

For example:

```
KEEP LIST IN K1 ONLY A;
```

would not include any OBSOLETE-DEFINITION members in the resulting KEPT-DATA list, but

```
KEEP LIST IN K1 INDEX-NAMES ONLY A;
```

would include any OBSOLETE-DEFINITIONs which have an index-name beginning with 'A'.

Note:

Once an OBSOLETE-DEFINITION member is included in a KEPT-DATA list it is processed as a member of that list regardless of its member type. That is, commands which process the KEPT-DATA list need not specify either INDEX-NAMES, or OBSOLETE-DEFINITIONS, in order that OBSOLETE-DEFINITIONs included in the list be processed.

Examples - OBSOLETE-DEFINITION Statements

Example 1

This example shows a data definition which has become obsolete and has been retained in the dictionary in the ARCHIVE clause of an OBSOLETE-DEFINITION member:

```
OBSOLETE-DEFINITION
OBSOLETE-DATE 1/OCT/1987
NOTE 'THIS MODULE HAS REPLACED BY MODULE GMC1'
SEE GMC1
ARCHIVE
NODULE
PARAMETER TRANSACTION-RECORD
OUTPUTS EMPLOYEE-LIST
ALIAS 'PRINT-EMPLOYEE-LIST'
CATALOG 'COBOL', 'MONTHLY', 'SPECIAL'
;
```

Example 2

This example shows an OBSOLETE-DEFINITION member which does not include the member's former data definition, but which contains the common clause NOTE indicating where that data definition can be found in another status.

```
OBSOLETE-DEFINITION
NOTE 'THE FORMER DATA-DEFINITION OF THIS MEMBER'
     "CAN BE FOUND IN BASE STATUS 'PROD'"
```

Visibility of OBSOLETE-DEFINITIONs

Refer to Figure 1 on page 394 for an illustration of the visibility of OBSOLETE-DEFINITIONs.

Normally, a user working in DEV2 would see through to the definition of mem-1 in DEV1. However, because it is an OBSOLETE-DEFINITION in DEV1, it is invisible to users in both DEV1 and DEV1. And, its physical presence in DEV1 means that the definition in LIVE is not visible either.

So, unless either:

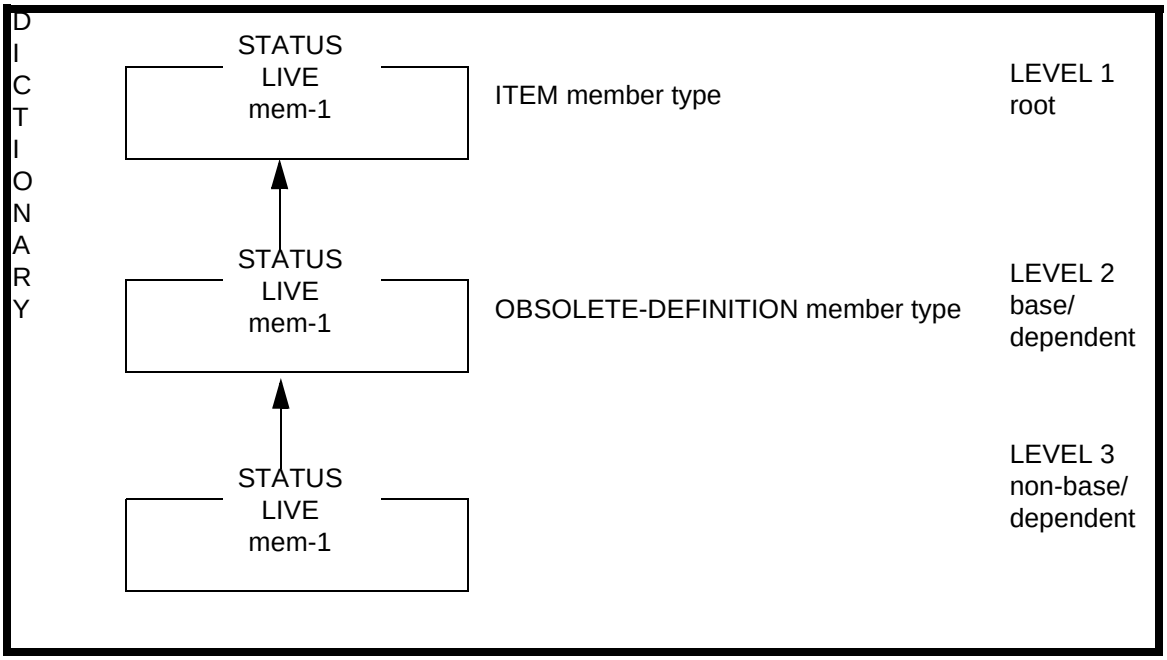
- *`index-name-type`* selection OBSOLETE-DEFINITIONs
- INDEX-NAMES selection keyword

or

- MEMBERS *`member-name-list`* selection

is specified in a command, mem-1 is only visible to users in LIVE.

Figure 1 • An illustration of the visibility of OBSOLETE-DEFINITIONs



OBSOLETE-DEFINITION Member Type Syntax

```
OBSOLETE-DEFINITION
[common clauses]

[ARCHIVE obsolete-definitional
 ;
 ;
```

Common Clauses

Common clauses can be used in members of any member type. They are all optional. Only one of each type of clause may be included in a member definition. They may appear in any order.

Common Clause Definitions

ACCESS-AUTHORITY

The ACCESS-AUTHORITY clause documents who has access to the data described by the member and the nature of that access.

The syntax of the ACCESS-AUTHORITY clause is as follows:

```
ACCESS-AUTHORITY 'name' < access > < , 'name' < access> >...
```

name is the name of a person, department or other entity, not exceeding 32 characters in length and conforming to the rules governing user names and owner names.

access may be READ-ONLY, UPDATE, or SECURITY-CONTROL. It applies to the immediately preceding *name*.

For example:

```
ACCESS-AUTHDRlTY 'ACCOUNTS' READ-ONLY, 'PERSONNEL' UPDATE,
'PERSONNEL-MANAGER' SECURITY-CONTROL
```

UPDATE authority is assumed to include READ-ONLY authority, and SECURITY-CONTROL authority is assumed to include UPDATE authority.

The Controller can set a default access in case it is not specified.

Refer to *ASG-ControlManager User's Guide* for details governing the naming of user and owner names.

ADMINISTRATIVE-DATA, COMMENT, COPYRIGHT, DESCRIPTION, NOTE AND QUERY

There are six common clauses that enable you to include text in member definition statements. The clauses have the same syntax, but are used to store different kinds of information.

The format of these common clauses is as follows:

```
ADMINISTRATIVE-DATA 'text'
COMMENT 'text'
COPYRIGHT 'text'
DESCRIPTION 'text'
NOTE 'text'
QUERY 'text'
```

where *text* is up to 32767 delimited character strings, each string having a maximum of 246 characters, including the delimiters.

For example, as in the following ADMINISTRATIVE-DATA clause:

```
ADMINISTRATIVE-DATA 'Specified by: A.Wake     '
                    'Coded by:        I.Hewitt'
                    'Test data:       D.Saunders'
```

When a member is processed by a REPORT or GLOSSARY command, strings in these text common clauses are not concatenated, but are aligned one beneath the other on the opening delimiters.

This enables you to include, for example, tables or lists in member definition statements.

ALIAS

The ALIAS clause records alternative names for a member. For example, the same member might be referred to differently by various departments of an organization or a member might be known by different names in different programming languages. A maximum of 16 aliases is permitted.

The syntax of the ALIAS clause is as follows:

```
ALIAS < alias-type > 'alias' < , < alias-type > 'alias' >...
```

where:

alias-type may be COBOL, ASSEMBLER, PL/I, TOTAL, IMS, ADABAS, SQL or a keyword determined by the macro DALIAS and specific to your installation. A maximum of 16 alias-type keywords are allowed. If no alias-type is specified for an alias, the alias is a general alias; otherwise it is a specific alias.

alias is the alternative name for the member and must be not more than 79 printable characters.

For example:

```
ALIAS 'ARTICLE NUMBER', 'PART NUMBER',
      ASSEMBLER 'ARTNUM'
```

An alias may be any string of characters and can be the same as the name of a member or the alias of another member. The only relationship recorded for an alias is to the member in whose ALIAS clause the alias appears.

When an alias is encountered for the first time on encoding any member, a data entries record is created for that alias. If the alias is subsequently encountered on encoding another member, the alias record is updated in respect of that member; a second data entries record is not created for the alias.

Refer to the installation manual relevant to your environment for details of the available alias-type keywords.

Refer to *ASG-ControlManager User's Guide* for details of the rules governing the naming of aliases.

CATALOGUE

The CATALOGUE clause classifies dictionary members into your own categories. This classification is independent of any other stated attributes of the member. It may be used, for example, to label all the data and programs used by a particular department in an organization.

The syntax of the CATALOGUE clause is as follows:

```
CATALOGUE 'classification' [, 'classification']...
```

where `classification` is a string of not more than 79 characters.

For example:

```
CATALOGUE 'PAYROLL', 'COSTING', 'MACHINESHOP',
                                        'R & D. DEPT.'
```

When a classification is encountered for the first time on encoding any member, a data entries record is created for that classification. If the classification is subsequently encountered on encoding another member, the classification is updated in respect of that member; a second data entries record is not created for the classification.

Refer to *ASG-ControlManager User's Guide* for details of the rules governing the naming of catalogue classifications.

EFFECTIVE-DATE and OBSOLETE-DATE

The EFFECTIVE-DATE clause records the date on which a member definition came into effect or is to come into effect. The OBSOLETE-DATE clause records the date on which a member definition became obsolete or is to become obsolete.

This is the syntax of the EFFECTIVE-DATE and OBSOLETE-DATE clauses:

```
EFFECTIVE-DATE date
OBSOLETE-DATE date
```

where:

date must be in the format which applies in your installation. If the separator between the elements of the date is a space, period, semi-colon, comma or right bracket, date must be delimited.

For example:

```
EFFECTIVE-DATE '30 JUN 84'
OBSOLETE-DATE 21/7/86
```

FREQUENCY

The FREQUENCY clause documents the frequency with which the data or process described by the member is accessed, run, updated or backed up.

This is the syntax of the FREQUENCY clause:

```
FREQUENCY frequency
```

or:

```
FREQUENCY {ACCESS    frequency}
          {RUN       frequency}
          {UPDATE    frequency}
          {BACK-UP   frequency}
```

frequency is:

```
{MONTHLY}   [ {n        } ]
{WEEKLY }     {'string'}
{DAILY  }
{HOURLY }
{number }
{'sting'}
```

where *n* is an integer and string is a string of not more than 256 characters.

For example:

```
FREQUENCY 'LAST FRIDAY OF EACH MONTH'
FREQUENCY RUN WEEKLY ACCESS 250
```

A FREQUENCY clause can contain:

- One frequency specification or
- Any number of ACCESS subordinate clauses

  and/or RUN subordinate clauses

  and/or UPDATE subordinate clauses

  and/or BACK-UP subordinate clauses,

in any order.

SECURITY-CLASSIFICATION

The SECURITY-CLASSIFICATION clause documents the security classification of the data or process to which the member relates. It provides a method of recording the security history of your data.

This is the syntax of the SECURITY-CLASSIFICATION clause:

```
SECURITY-CLASSIFICATION 'string' [FROM date]
                          [, 'string' [FROM date]]...
```

where:

string is a string of not more than 256 characters which defines the security classification.

date is the date from which the security classification is or was applicable, and must be in the format which applies in your installation, for example '12 DEC 1983' or 12/12/83. If the separator between the elements of the date is a space, period, semi-colon, comma or right bracket, date must be delimited.

For example:

```
SECURITY-CLASSIFICATION 'RESTRICTED' FROM 830811
                        'CONFIDENTIAL' FROM 840201
SECURITY-CLASSIFICATION 'SECRET'
```

The FROM clause specifies the date on which the security classification is (or was) first effective. A later date specified in a FROM clause associated with another 'string' may specify the date on which the security classification stated in the earlier-dated 'string' became (or is to become) superseded, or may state the effective date of a parallel classification, or other details. This depends on the security system defined for the real data in your particular installation. The SECURITY-CLASSIFICATION clause must be interpreted in relation to that system.

SEE

The SEE clause documents cross references to other members.

The syntax of the SEE clause is as follows:

```
SEE member-name [FOR 'string']
                   [, member-name [FOR 'string']]...
```

member-name is the name of a member, and *string* is a string of not more than 256 characters. If *member-name* is not already encoded, it is created as a dummy member of the same member type as the member containing the SEE clause.

For example:

```
SEE GENCOB3 FOR 'PRODUCE COBOL COMMAND',
GENRECIN3 FOR 'PRODUCE RECORD LAYOUTS OF INPUT FORM',
GENRECOU3 FOR 'PRODUCE RECORD LAYOUTS OF OUTPUT REPORT'
```

The clause may be used to direct a user to a member which contains relevant information.

Refer to *ASG-ControlManager User's Guide* for details of the rules governing the naming of dictionary members.

Examples of Common Clauses

```
ADMINISTRATIVE-DATA
'BATCHED INPUT FORMS WITH CONTROL TOTALS TO BE SENT'
'FROM PROGRESS DEPT TO DATA CONTROL BY 4 P.M. DAILY'
COMMENT 'THIS IS THE THIRD VERSION OF THIS DEFINITION.'
'IT TAKES INTO ACCOUNT ALL COMMENTS RECEIVED'
'UP TO 30 JUNE 1981.'

DESCRIPTION 'THIS ITEM IS THE TOTAL EX WORKS VALUE'
'OF THE ORDER IN LOCAL CURRENCY'

NOTE 'INPUT DATA FOR "SAMIS"'
'SUITE OF PROGRAMS'

QUERY 'REFER ANY QUESTIONS TO T.BAKER, PRODUCT'
'DEVELOPMENT'

ACCESS-AUTH0RITY 'DEPARTMENTAL MANAGERS ONLY'
,'PERSONNEL MANAGER' SECURITY-CONTROL
,'A.SMITH OR J.BROWN' UPDATE

CATALOG 'STAFF NUMBER', 'STAFF NAME', 'ADDRESS'
EFFECTIVE-DATE 870611
OBSOLETE-DATE '8.8.87'
ALIAS CO 'SUPPLIER-NO', AS 'SUPNO'
FREQUENCY ACCESS '15 PER HOUR' UPDATE DAILY 10
SECURITY-CLASSIFICATION 'COMPANY SECRET', 'ENCRYPTED'
SEE PERSONNEL-RECORDS FOR 'ADMINISTRATIVE-DATA'
```

6 Selectable Units for Use in Dictionary Management

Additional capabilities are provided by selectable units. These can be incorporated into a Manager Products Nucleus to provide just those capabilities that your installation needs.

An overview of these DataManager selectable units is given in this chapter:

Code	Facility Name	Other Facilities Required
DMR-DD1	DataManager Nucleus	CMR-CM01, DYR-DY01
DMR-DD4	TOTAL Definition	DMR-DD1
DMR-DD5	IMS (DL/I) Definition	DMR-DD1
DMR-DD6	ADABAS Definition	DMR-DD1
DMR-DD8	IDMS Definition	DMR-DD1
DMR-DD9	System 2000/80 Definition	DMR-DD1
DMR-DD10	Siemens Universal Database System Definition	DMR-DD1
DMR-DD11	SESAM Definition	DMR-DD1
DMR-DD12P	DB2 Definition Prototype	DMR-DD1, DYR-TE00, DYR-TE12P, DYR-UD15, CMR-UD1, CMR-UI1 and DMR-SL1 or DMR-SL2 or DMR-SL3
DMR-DD32P	SQL/DS Definition Prototype	DMR-DD1, DYR-TE00, DYR-TE32P, DYR-UD15, CMR-UD1, CMR-UI1 and DMR-SL1 or DMR-SL2 or DMR-SL3
DMR-MP01	Manager Products Integrator	DMR-DD1 and DSR-MP03 or SMR-MP04
DMR-SL1	Source Language Generation (COBOL)	DMR-DD1
DMR-SL2	Source Language Generation (PL/I)	DMR-DD1

Code	Facility Name	Other Facilities Required
DMR-SL3	Source Language Generation (Assembler)	DMR-DD1
DMR-SL4	TOTAL Generation	DMR-DD4, DMR-DD1
DMR-SL5	IMS (DL/I) Generation	DMR-DD5, DMR-DD1
DMR-SL6	ADABAS Generation	DMR-DD6, DMR-DD1
DMR-SL7	Mark IV Generation	DMR-DD1
DMR-SL9	System 2000/80 Generation	DMR-DD1, DMR-DD9
DMR-SL10	Siemens Universal Database System Generation	DMR-DD1, DMR-DD10
DMR-SL11	SESAM Generation	DMR-DD1, DMR-DD11

An overview of the following ControlManager selectable units which are of use in dictionary management is also given in this chapter:

Code	Facility Name	Other Facilities Required
CMR-AD2l	Advanced Status	CMR-CM01, CMR-DD2
CMR-DD2	Basic Status	CMR-CM01
CMR-DD3	Audit and Security	CMR-CM01
CMR-UD1	User Defined Syntax	CMR-CM01

Interfaces to Database Management Systems

DataManager provides Interfaces to the major relational, network and hierarchical Database Management Systems (DBMSs). With the appropriate facility installed you can:

- Define your database in the dictionary and process the definitions using any of the dictionary management commands provided with Manager Products. This necessitates, specifically for each database management system for which an interface facility is provided:
  - One or more additional member types to define the database and its constituents
  - Extensions to some of the basic member types, so that the constituent parts of the database can be fully defined
  - Additional keywords for use in certain dictionary management commands

- Generate from the dictionary source data descriptions in the DBMS's data description language, either for a complete database or for parts thereof, for input either to the DBMS's dictionary/directory or direct to database utilities.
- Generate from the dictionary source data descriptions in the major programming languages (COBOL, PL/I or Assembler) for input into the required source library. The ability to generate File Definition forms for input to the MARK IV File Management System is also provided with the Mark IV Interface.

In order to document a database in your Manager Products dictionary you must have the appropriate DataManager database selectable unit installed. The selectable units available are as follows (the title of each selectable unit implies the DBMS it supports):

DMR-DD4 TOTAL Definition
DMR-DD5 IMS (DL/1) Definition
DMR-DD6 ADABAS Definition
DMR-DD8 IDMS Definition
DMR-DD9 System 2000/80 Definition
DMR-DD10 Siemens Universal Database System Definition
DMR-DD11 Siemens SESAM Definition
DMR-DD12P DB2 Definition Prototype
DMR-DD32p SQL/DS Definition prototype

In order to generate source data descriptions for an IDMS/R, DB2 or SQL/DS environment you must have the appropriate DictionaryManager selectable unit installed. The selectable units required are as follows:

- DYR-TE08 Corporate Dictionary/Repository Definition Export for IDD
- DYR-TE12P Corporate Dictionary/Repository Definition Export for DB2 Prototype
- DYR-TE32P Corporate Dictionary/Repository Definition Export for SQL/DS Prototype

In order to generate source data descriptions for the other Interfaces you must have the appropriate DataManager selectable unit installed. These selectable units are available:

- DMR-SL4 TOTAL Generation
- DMR-SL5 IMS (DL/I) Generation
- DMR-SL6 ADABAS Generation
- DMR-SL9 System 2000/80 Generation
- DMR-SL10 Siemens Universal Database System Definition Generation
- DMR-SL11 Siemens SESAM Generation

In order to generate source data descriptions in Cobol, PL/1 or Assembler, or generate File Definition forms for input to the MARK IV File Management System you must have the appropriate DataManager selectable unit installed. These selectable units are available:

- DMR-SL1 Source Language Generation (COBOL)
- DMR-SL2 Source Language Generation (PL/I)
- DMR-SL3 Source Language Generation (Assembler)
- DMR-SL7 MARKIV Generation

Source Language Generation

The Source Language Generation facility enables a user to generate data descriptions in COBOL, PL/I or Assembler source language, or in the data description language of various database management systems, from encoded members of a dictionary. The ability to generate File Definition forms for input to the MARK IV File Management System is also provided.

The facility comprises one additional command: PRODUCE. The PRODUCE command generates data descriptions in the selected language, from encoded members of the dictionary, and optionally writes them to a user specified output file. The generation is performed from the members named in the command and from all members directly or indirectly contained by those members. Installation macros are provided by means of which the source code output can be specified to conform to user standards.

Additionally or alternatively, the PRODUCE command can generate record layouts. These are tables of the groups and items comprised in FILE or GROUP members named in the command, together with their offsets and formats. Again, installation macros enable the output of record layouts to be tailored to user standards.

The PRODUCE command format includes a number of optional clauses to control the generation, and other optional clauses to permit editing of the generated data names, or selection of data names appropriate to the source language from members' aliases.

The Source Language Generation manual fully describes the facility in respect of the generation of COBOL, PL/I or Assembler source data descriptions. The variations of the PRODUCE command to permit it to generate data descriptions in database management systems' data description languages are documented in the appropriate DataManager database interface manuals; see "Interfaces to Database Management Systems" on page 404.

The variation of the PRODUCE command to permit it to generate MARK IV File Definition forms is documented in the DataManager MARK IV Interface manual.

Interfaces to File Management Systems

An interface to the MARK IV File Management System is available. The interface facility (selectable unit DMR-SL7) provides the ability to define MARK IV files and segments to DataManager, and to generate MARK IV File Definition forms from encoded dictionary members. For this, the PRODUCE command of the Source Language Generation facility is used, an appropriate keyword being made available with that command. The MARK IV File Definition forms can be generated from members defining MARK IV files or conventional files, or from members defining IMS databases or segments.

The facility is documented in *ASG-DataManager MARK IV Interface Facility*. This must be read in conjunction with the *ASG-Manager Products Source Language Generation*, which contains the basic documentation of the PRODUCE command.

Manager Product Integrator

The Manager Product Integrator for use with DataManager (selectable unit DMR-MP0l) allows a DataManager user to utilize the functionality of any other integrated dictionary-driven products in the Manager Family which also have integrators installed.

In an Integrated Manager Product Installation, the individual products can be arranged so that they share the same dictionary or access separate dictionaries. In either case, any additional facilities which are available to one product and are not product-specific, become available to the other products by way of the integrators.

The Status Facilities

ControlManager offers Status facilities as two optional selectable units.

These are:

- Basic Status (selectable unit CMR-DD2)
- Advanced Status (selectable unit CMR-AD21).

Basic Status. Provides the capabilities necessary to partition the dictionary into several statuses and allow several versions of the same dictionary member to be recorded—one in each status. Status-related interrogation capabilities are provided to enable you to compare and analyze the contents of those statuses.

Advanced Status. Incorporates all Basic Status capabilities. In addition, any status may be updatable (this is not the case with Basic Status) and a Status Window feature is provided.

Note:

Although Basic Status is a prerequisite for the installation of Advanced Status, from an operational viewpoint they are mutually exclusive. You will be using either Advanced Status or Basic Status, not both.

Audit and Security

The Audit and Security facility (selectable unit CMR-DD3) provides the following major benefits:

- A security system which gives the Controller complete control over who can use the dictionary and what users can do when they have accessed it. Where appropriate, users can be prevented from accessing, inserting, altering or removing all, or a defined group of, the members in the dictionary. For example, members developed by a particular project can be declared to be *owned* by that project so that access to those members by users outside the project can be controlled on a need-to-know basis.

  Flexible controls can be applied so that, for example, the Controller can allow a user to access a member but not alter or remove it from the dictionary. Users other than the Controller can be authorized to protect members or assign ownership to one or more members.

- An audit system for generating a comprehensive selection of reports from information held on the dictionary's log or log archive data set. Using the commands available with the facility the Controller can obtain information about:

  — Who has been using the dictionary and what they have done

  — What use was made of the dictionary at a particular time, if, for example, a security breach is suspected

  — Which updating commands issued during a given period were unsuccessful. If a batch job was run overnight, this information could be used to avoid manual checking of the batch output

  — What changes have been made to the dictionary and by whom.

- The ability to log non-updating as well as updating commands

- The ability to log selected Manager products messages or messages of particular severity levels, which can then be included in audit reports.

User Defined Syntax

The User Defined Syntax facility (selectable unit CMR-UD1) lets you define your own member types and attributes for use in your dictionary. Using the facility you can tailor the dictionary so that it precisely matches the specific requirements of your organization. For example:

- You can rename member types, or create new member types, to reflect the terminology in use in your organization. This feature is particularly valuable not only in non-English speaking countries but also at sites whose terminology does not match that in use in the standard ASG software products.
- You can create entirely new member types, outside the standard range of member types provided by ASG, to support your own dictionary requirements and computer applications
- You can define the relationships that can be supported between member types, within the general structure of relationships permitted by ASG
- You can create new attributes for inclusion in the members of the dictionary, specifying your own validation rules for the values of such attributes if you wish.

Index

A

ACCESS-AUTHORITY clause 395
ADD command 49
ADMINISTRATIVE-DATA clause 395
AFTER
    time-related-selection 309
ALIAS clause 396
ALIASES keyword
    GLOSSARY command 137
    LIST command 158
    PERFORM command 195
ALL-STATUSES keyword
    GLOSSARY command 137
    LIST command 159
    PRINT command 209
ALPHABETICALLY keyword
    BULK command 82–83
    GLOSSARY command 137, 139, 145
    LIST command 159, 163–164
    PERFORM command 195, 201–202
ALSO command 52
ALTER command 52
AMEND command 69
AND keyword
    ALSO command 157
    DROP command 127
    KEEP command 157
    time and user-related selections 312
AND, OR keywords
    name-related-selection 297
ANY keyword
    name-related-selection 297
AT keyword
    time-related-selection 311
Audit and Security facility 79
AUTHORITY command 78

B

basic concepts 9
Basic Status
    Status-related-selection keywords 300
BEFORE
    time-related-selection 309
BELONGS-TO clause
    WHAT command 254
    WHICH command 266
BETWEEN, AND
    time-related-selection 309
BULK ENCODE command 80
BULK PRINT command 91

C

catalog classifications
    WHAT command 257
    WHICH command 277
CHANGED
    status-related-selection 301
CLEAR-KEPT-DATA keyword
    PERFORM command 202
CLOSE command 113
CMS files
    READ and WRITE commands 218
COMMAND-STREAM
    member type definition 387
    PERFORM command 195
COMMENT clause 396
comparing statuses 293
condition
    WHAT command 253
    WHICH command 265
CONSTITUTES clause
    WHAT command 253
    WHICH command 266
CONTRAST command 115
conventions page xv
COPY command 117
COPYRIGHT clause 395
Corporate Dictionary/Repository
    overview 2
CURRENT keyword 300

D

Database Management System Interface
    Facilities 407

introduction 407
DataManager
Database Management System
Interface Facilities 407
selectable unit codes 407
DBUF clause
DICTIONARY command 123
DESCRIPTION clause 395
DETAILS-ONLY keyword
GLOSSARY command 137
LIST command 159
DICTIONARY command 119
CONSITUTES clause 253
DOES command 126
GLOSSARY command 136
USES clause 253
WHAT command 253
DISPLAY-ONLY keyword
PERFORM command 196
DIVERGING keyword
status-related-selection 303
DOES command 126
DOWN-TO clause
REPORT command 231
DROP command 127

E

EFFECTIVE-DATE clause 398
ENCODE command 129
ENCODED-SOURCE keyword
COPY command 118
PRINT command 211
ERASE KEPT-DATA command 131
EXCEPT clause
BULK PRINT command 92
GLOSSARY command 137
EXISTING keyword
LOCK command 174

F

FETCH KEPT-DATA command 132
FILE
member type definition 357
FILE command 133
FOR clause
GLOSSARY command 136
FORMAT AS command 135
form-description
ITEM member definition 322
form-description keywords 324
FORMS clause
WHAT command 254
WHICH command 266
FROM keyword
DROP command 127
name-related-selection 297

G

GIVING clause
GLOSSARY command 307
GLOSSARY command 136
GROUP
member type definition 351

H

HIERARCHY keyword
REPORT command 231
HISTORY keyword
LIST command 159

I

IBUF clause
DICTIONARY command 123
IF clauses 343
CONTAINS clause
FILE definition 359, 371
GROUP member definition 353
operators 343
IMS (DLИ)Source Language
Generation 406
IN STATUS
time and user-related-selection 312
Index-names
definition 316
INDIRECTLY keyword
DOES command 126
GLOSSARY command 138
WHAT command 254
WHICH command 254
input line
PERFORM command 197
INSERT command 152
installation
Source Language Generation 406
INTRODUCED
time and user-related selections 308
IS clause
WHAT command 259

K

KEEP command 156
KEPT-DATA
BULK ENCODE command 81
BULK PRINT command 92
BULK REPORT command 100
GLOSSARY command 137

LIST command 158
PERFORM command 195
QUERY command 217
WHICH command 265

L

LENGTH
name-related-selection 296
LINES clause
PRINT command 209
LIST command 158
LOCALLY keyword
WHAT command 254
LOCK command 173
LOCKED keyword 82
BULK ENCODE command 81
BULK PRINT command 92
BULK REPORT command 100
GLOSSARY command 137
LIST command 158
PERFORM command 195
LOCK-INFORMATION keyword
LIST command 162
LOCK command 159
LOCKS keyword
LIST command 169
LONG-LITERAL
SHOW UDS command 245

M

member definition statements 322
member names
definition 92
member type
FILE 357
GROUP 317
Member Types
COMMAND-STREAM 318
member types
common clauses 395
introduction 317
ITEM 317
OBSOLETE-DEFINITION 318
SYSTEM/PROGRAM/MODULE 320
MEMBERS clause
BULK PRINT command 92
BULK REPORT command 100
GLOSSARY command 137
LIST command 158
PERFORM command 195
MODE command 230
MODIFY command 178
monitoring dictionary buffer steals
DICTIONARY command 119
MVS dataset
READ and WRITE commands 218

N

NAME clause
in ITEM member definition 324
Name-Related-Selection
syntax 298
NEW
status-related-selection 301
NEW keyword
LOCK command 173
NOPRINT keyword
ALTER command 55
INSERT command 155
NOSKIP keyword
REPORT command 235
NOTE clauses 391
NO-USAGE keyword
LIST command 159
NUMBER clause
ADD command 50
INSERT command 153
REPLACE command 228

O

OBSOLETE-DEFINITION
creating 391
examples 393
member type 389
processing 392
ONLY keyword
name-related-selection 294
OR keyword
time and user-related-selections 312
owner
WHAT command 258
WHICH command 266
ownership
WHO command 290

P

password format
AUTHORITY command 78
PERFORM command 195
picture
form-description 326
PICTURE clause
alphabetic 331
alphanumeric 331
alphanumeric edited 332

floating symbols 330
ITEM member definition 324
numeric 333
numeric edited 334
symbols 328
PLURAL-LITERAL
SHOW UDS command 245
PRINT command 208
PRINT FIRST keyword
PERFORM command 196
PRINT keyword
ALTER command 52
INSERT command 153
PERFORM command 196
PROTECT command 213
protecting members
PROTECT command 214

Q

QUERY clause 395
QUERY MODE command 177

R

READ command 219
read-only opening mode
DICTIONARY command 121
Reference Count
WHICH command 281
REFERENCES keyword
GLOSSARY command 136
REFERRING-TO clause
WHICH command 266
REINSTATE command 222
RELEASE keyword
LOCK command 175
REMOVE command 223
remove security level
PROTECT command 215
removing ownership
PROTECT command 216
RENAME command 224
renaming a member
RENAME command 224
RENEW keyword
LOCK command 175
RENUMBER keyword
ALTER command 54
MODIFY command 178
REPLACE command 226
REPORT command 230
reporting user-defined attributes
SHOW UDS command 246
REVERIFIED
status-related-selection 300
RFILE command 236

S

SBUF clause
DICTIONARY command 123
secondary selection
name
BULK ENCODE command 81
BULK PRINT command 92
BULK REPORT command 100
GLOSSARY command 137
LIST command 159
PERFORM command 195
WHICH command 266
status
BULK ENCODE command 81
BULK PRINT command 92
BULK REPORT 100
GLOSSARY command 137
LIST command 159
PERFORM command 195
WHICH command 266
time
BULK ENCODE command 81
BULK PRINT command 92
BULK REPORT command 100
GLOSSARY command 137
LIST command 159
PERFORM command 195
WHICH command 266
user
BULK ENCODE command 81
BULK PRINT command 92
BULK REPORT command 100
GLOSSARY command 137
LIST command 159
PERFORM command 195
WHICH command 266
Secondary Selection clauses syntax
status-related-selection 308
time and user-related-selection 314
security levels 18
access 128, 157
alter 128, 157
remove 169
specifying 216
SECURITY-CLASSIFICATION clause 399
selection
based on status 293
based on time and user 293
Selection based
on name 293
SHOW ALIAS-TYPES command 237

SHOW MEMBER-TYPE command 238
SHOW PRODUCE-OPTIONS
    command 242
SHOW UDS command 244
SIGNIFICANT keyword
    GLOSSARY command 137
SKIP keyword
    REPORT command 231
SOURCE-FILE command 249
SOURCES keyword
    BULK ENCODE command 81
    BULK PRINT command 95
    LIST command 159
    PERFORM command 195
SPECIFIED keyword
    WHICH command 266
STANDARD-LITERAL
    SHOW UDS command 245
Status Window
    WHAT command 254
status-related-selection 299
STORE KEPT-DATE command 250
SYSNAME clause
    DICTIONARY command 122
SYSTEM member type 380

T

time and user-related-selection syntax 314
TO keyword
    name-related-selection 295
TOTALS-ONLY keyword
    LIST command 169

U

UDS tables
    SHOW UDS command 247
UNVERIFIED
    status-related-selection 300
UPDATE command 251
update opening mode
    DICTIONARY command 121
usage count
    WHICH command 281
USAGE keyword
    GLOSSARY command 136
usage tables 13
USED-BY clause
    WHICH command 266
user-defined attributes
    SHOW UDS command 246
user-related-selection 293
USES clause
    WHAT command 253
USES keyword
    WHICH command 266

V

VALIDATE command 252
VERIFIED
    time and user-related-selection 308
VIA keyword
    WHICH command 266

W

WBDA command 177
WHAT command 253
WHICH command 265
WHO command 290
WHOSE command 291
WIPE-OUT keyword
    ALTER command 52
    MODIFY command 178
WRITE command 218