English

Fujitsu Software

BS2000 OS DX
Utility Routines

User Guide

Valid for:
BS2000 V21.0A

Edition November 2022

Comments… Suggestions… Corrections…

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

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

Certified documentation according to DIN EN ISO 9001:2015

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

Copyright and Trademarks

Copyright © 2022 Fujitsu Technology Solutions GmbH.

All rights reserved.
Delivery subject to availability; right of technical modifications reserved.

All hardware and software names used are trademarks of their respective manufacturers.

Table of Contents

Utility Routines **15**
1 Preface **16**
1.1 Objectives and target groups of this manual **18**
1.2 Summary of contents **19**
1.3 Changes since the last edition of the manual **20**
1.4 Notational conventions **21**
2 DPAGE Outputting and modifying disk files **23**
2.1 Support for pubsets **24**
2.2 Starting the program run **25**
2.3 Statements **26**
2.3.1 BKPT - Interrupt DPAGE 27
2.3.2 DISPLAY - Output page to SYSOUT 28
2.3.3 EDT - Call EDT 30
2.3.4 END / HALT - Terminate DPAGE 31
2.3.5 MODIFY - Modify contents of page 32
2.3.6 OPEN - Open file 34
2.3.7 PRINT - Print page 35
2.3.8 READ - Read page 36
2.3.9 WRITE - Write internal work area back to file 37
2.4 DPAGE messages **38**
3 INIT Initializing (emulated) magnetic tapes **41**
3.1 Operating modes **43**
3.1.1 Normal mode 44
3.1.2 Console mode 45
3.2 Program run **46**
3.2.1 Program start 47
3.2.2 Initializing a magnetic tape (example) 48
3.2.3 Program termination 50
3.2.4 Problems in volume initialization 51
3.3 Statements **52**
3.3.1 INIT - Initialize magnetic tape 54
3.3.2 LIST - Read and output magnetic tape labels 60
3.3.3 OPTION - Activate and deactivate optional functions 62
3.3.4 END - Terminate INIT session 65
3.3.5 HELP - Outline description of INIT statements 66
3.4 Structure of the labels **67**
3.4.1 Volume label VOL1 for magnetic tapes 68

3.4.2 File label HDR1 for magnetic tapes 69
3.4.3 File label HDR2 for magnetic tapes 71
3.4.4 File label HDR3 for magnetic tapes 72
4 IORM Dynamic control of I/O resources 73
4.1 IOPT: I/O Priority Handling for Tasks 78
4.1.1 IOPT statements 80
4.1.1.1 Activating and deactivating IOPT 81
4.1.1.2 Defining and querying threshold values for I/O priority classes 82
4.1.1.3 Activating and deactivating disk devices for IOPT 83
4.1.1.4 Device groups 85
4.1.1.5 Defining threshold values for I/O priority LOW 87
4.1.1.6 Defining threshold values for I/O priority MEDIUM 92
4.1.1.7 Querying threshold values for I/O priorities 96
4.1.1.8 Querying utilization 98
4.1.1.9 Check mode 102
4.1.2 Typical applications 107
4.2 DPAV: Dynamic Parallel Access Volume 108
4.2.1 Statements 110
4.2.1.1 Activating and deactivating DPAV 111
4.2.1.2 Activating and deactivating alias devices for DPAV 112
4.2.1.3 Check mode 114
4.2.1.4 Activating and deactivating base devices for DPAV 118
4.2.2 Typical application 120
4.2.3 FastDPAV 121
4.3 DDAL: Optimized Load Balancing in ETERNUS CS operation 122
4.4 TCOM: Dynamic Tape Compression 123
4.5 IOLVM: I/O Limit for Virtual Machines 125
5 JMP Reconstruction of ENTER commands from the JMS job pool 126
5.1 Execution of JMP 127
5.2 Statements 128
5.2.1 Overview of JMP statements 129
5.2.2 Description of the statements 130
5.2.2.1 CREATE-PROCEDURE-FILE - Create SAM file with BS2000 procedure 131
5.2.2.2 END - Terminate statement input 132
5.2.2.3 OPEN-JOBPOOL-FILE - Open job pool file 133
5.2.2.4 SHOW-JOBPOOL-STATUS - Output information on the job pool 134
5.3 Notes on reconstructed attributes 141
5.4 JMP messages 150
6 JMU Creating and maintaining the SJMSFILE system file 151
6.1 Job management 152
6.2 Execution of JMU 154

6.3 Statements 156
6.3.1 Overview of JMU statements 157
6.3.2 Description of the statements 158
6.3.2.1 CREATE-PROCEDURE-FILE - Create SAM file containing BS2000 procedure 159
6.3.2.2 DEFINE-JOB-CLASS - Write job class definitions to SJMSFILE 161
6.3.2.3 DEFINE-JOB-STREAM - Write stream definitions to SJMSFILE 170
6.3.2.4 DELETE-JOB-CLASS - Delete class definitions 174
6.3.2.5 DELETE-JOB-STREAM - Delete stream definitions 175
6.3.2.6 END - Terminate statement input 176
6.3.2.7 GRANT-JOB-CLASS-ACCESS - Control access by user IDs to job class 177
6.3.2.8 MODIFY-JOB-CLASS - Modify job class definitions 178
6.3.2.9 MODIFY-JOB-STREAM - Modify stream definitions 180
6.3.2.10 REMOVE-USER - Prohibit access to private job classes 181
6.3.2.11 SET-JOB-CLASS-DEFAULT - Specify default classes for users 182
6.3.2.12 SET-MODIFICATION-MODE - Set modification mode 183
6.3.2.13 SET-POSIX-JOB-CLASS-DEFAULT - Specify POSIX default classes for users 184
6.3.2.14 SHOW-JOB-CLASS - List contents of class definitions or names of classes 185
6.3.2.15 SHOW-JOB-STREAM - List contents of stream definitions or names of streams 187
7 LMSCONV Generation and management of libraries 189
7.1 Libraries 194
7.1.1 Logical structure of a library 195
7.1.2 Input and output libraries 196
7.1.3 Multiple access to libraries 197
7.2 Members 198
7.2.1 Multiple access to members 199
7.2.2 Member type definition 200
7.2.3 Convention for member designations 202
7.2.4 Member designations in statements 203
7.2.5 Logging the member designations 204
7.2.6 Selectors for member designations 205
7.2.7 Constructors for member designations 206
7.2.8 Member attributes 208
7.2.9 Type dependencies 209
7.2.10 Version management 210
7.2.11 Data protection by overwriting 212
7.2.12 Auditing 213
7.2.13 Extended Host Code Support (XHCS) 214

7.3 LMSCONV functions 216
7.3.1 Starting LMSCONV 217
7.3.2 Assigning libraries 219
7.3.3 Processing members 220
7.3.4 Controlling the LMSCONV run 224
7.3.5 Disks without PAM key 227
7.3.6 NK4 disks 231
7.3.7 Handling alias names (ACS) 233
7.4 Statements 234
7.4.1 Overview of the LMSCONV statements 235
7.4.2 LMSCONV statements ADD-ELEMENT to MODIFY-ELEMENT 237
7.4.2.1 ADD-ELEMENT - Add member to library 238
7.4.2.2 CLOSE-LIBRARY - Close library 245
7.4.2.3 COPY-ELEMENT - Copy member 246
7.4.2.4 COPY-LIBRARY - Copy library 253
7.4.2.5 DELETE-ELEMENT - Logically delete member 255
7.4.2.6 END - Terminate LMSCONV 260
7.4.2.7 EXTRACT-ELEMENT - Output member to file 262
7.4.2.8 MODIFY-DEFAULTS - Modify defaults 269
7.4.2.9 MODIFY-ELEMENT - Modify member 285
7.4.3 Substatements of MODIFY-ELEMENT for member types R, C and L 292
7.4.3.1 ADD-REP-RECORD - Add REP records to object module 293
7.4.3.2 ADD-TEXT-MODIFICATION - Correct text records of an object module . 294
7.4.3.3 DELETE-RECORD-TYPE - Exclude record types from input member . . . 296
7.4.3.4 END-MODIFY - Terminate input of substatements 297
7.4.3.5 MODIFY-CSECT-ATTRIBUTES - Modify CSECT attributes 298
7.4.3.6 MODIFY-MODIFICATION-DEFAULTS - Specify global defaults 300
7.4.3.7 REMOVE-MODIFICATION - Cancel corrections 302
7.4.3.8 RENAME-SYMBOLS - Rename symbols 303
7.4.4 Substatements of MODIFY-ELEMENT for text members 304
7.4.4.1 ADD-RECORD - Add records 305
7.4.4.2 END-MODIFY - Conclude substatements 306
7.4.4.3 REMOVE-RECORD - Delete record or record area in member 307
7.4.5 LMSCONV statements MODIFY-ELEMENT-ATTRIBUTES to WRITE-COMMENT 308
7.4.5.1 MODIFY-ELEMENT-ATTRIBUTES - Modify member attributes 309
7.4.5.2 MODIFY-LOGGING-PARAMETERS - Modify logging settings 314
7.4.5.3 OPEN-LIBRARY - Open global library 317
7.4.5.4 SHOW-DEFAULTS - Output current default values 319
7.4.5.5 SHOW-ELEMENT - Display contents of member 321
7.4.5.6 SHOW-ELEMENT-ATTRIBUTES - Display member attributes 333

7.4.5.7 SHOW-LIBRARY-ATTRIBUTES - Display library attributes 343
7.4.5.8 SHOW-LIBRARY-STATUS - Display library status 344
7.4.5.9 SHOW-LOGGING-PARAMETERS - Display global LMSCONV parameters . 345
7.4.5.10 SHOW-TYPE-ATTRIBUTES - Display attributes of a member type 346
7.4.5.11 SHOW-USER-EXITS - Display LMSCONV version 348
7.4.5.12 WRITE-COMMENT - Write comments to output medium 349
7.5 Example: Modifying a link load module 350
7.6 Comparison between LMSCONV and LMS 351
8 MSGMAKER Processing of BS2000 Message Files 355
8.1 Execution of MSGMAKER 357
8.1.1 Starting the routine 358
8.1.2 Defining a monitoring job variable 359
8.1.3 MSGMAKER operating modes 360
8.1.4 Special character sets 361
8.1.5 Messages of MSGMAKER 362
8.2 Menu mode 363
8.2.1 Mask overview 364
8.2.2 Mask sequence 365
8.2.3 General mask format 367
8.2.4 Making entries in masks 371
8.2.5 Description of fields that occur frequently 372
8.2.6 Description of the masks 374
8.2.6.1 MENU mask - MSGMAKER main mask 375
8.2.6.2 MSG-FILE-ATTRIBUTES mask - Enter and modify message file attributes . . 381
8.2.6.3 COPY mask - Copy message units 383
8.2.6.4 MOVE mask - Copy and delete message units 387
8.2.6.5 SHOW mask - Display message file contents 392
8.2.6.6 SHOW-OUTPUT mask - Output message units and additional information . . 395
8.2.6.7 ADD-MSG mask - Add message unit 400
8.2.6.8 MODIFY-MSG mask - Modify message unit 404
8.2.6.9 MSG-TEXT mask - Add or modify message text 409
8.2.6.10 MEANING/RESPONSE mask - Add or modify meaning and response text . 412
8.2.6.11 INSERT-ATTRIBUTES mask - Add or modify insert attributes 415
8.2.6.12 DELETE-MSG mask - Delete message unit 418
8.2.6.13 ADD-DOCUMENTATION mask - Add documentation lines 422
8.2.6.14 MODIFY-DOCUMENTATION mask - Modify, add and delete documentation lines 425
8.2.6.15 DELETE-DOCUMENTATION mask - Delete documentation lines 427

8.3 Statements 429
8.3.1 Overview of statements 430
8.3.2 Description of the statements 431
8.3.2.1 ADD-DOCUMENTATION - Add documentation lines 432
8.3.2.2 ADD-MSG - Add message unit 434
8.3.2.3 COPY - Copy message unit 442
8.3.2.4 DELETE-DOCUMENTATION - Delete documentation lines 447
8.3.2.5 DELETE-MSG - Delete message unit 449
8.3.2.6 END - Terminate MSGMAKER 451
8.3.2.7 MERGE-MSG-FILES - Merge message files 452
8.3.2.8 MODIFY-DOCUMENTATION - Modify and delete documentation lines . . 454
8.3.2.9 MODIFY-MSG - Modify message unit 458
8.3.2.10 MODIFY-OPTION - Overwrite message unit 468
8.3.2.11 MOVE - Copy and delete message unit 469
8.3.2.12 OPEN-MSG-FILE - Open message file 475
8.3.2.13 SHOW - Display message file contents 478
8.3.3 Special features of statements in menu mode 482
8.3.3.1 ADD-DOCUMENTATION 484
8.3.3.2 ADD-MSG 485
8.3.3.3 COPY 487
8.3.3.4 DELETE-DOCUMENTATION 488
8.3.3.5 DELETE-MSG 489
8.3.3.6 END 490
8.3.3.7 GO-TO - Branch to specified mask 491
8.3.3.8 MERGE-MSG-FILES 492
8.3.3.9 MODIFY-DOCUMENTATION 493
8.3.3.10 MODIFY-MSG 494
8.3.3.11 MOVE 496
8.3.3.12 SHOW 497
8.3.4 Example 498
9 PAMCONV Conversion of file formats 503
9.1 Starting the program run 506
9.2 Functionality of PAMCONV 507
9.3 Conversion of file formats 515
9.3.1 Types of conversion 516
9.3.2 System environment requirements 522
9.3.3 Specifying source and target files 523
9.4 Reblocking 525
9.4.1 Explicit reblocking 526
9.4.2 Implicit reblocking 527
9.4.3 Reblocking PAM-DATA files without changing the file format 528

9.4.4 Problems when decreasing the blocking factor 529
9.5 Controlling conversion and reblocking 530
9.5.1 Special points relating to conversion 531
9.5.2 Further notes on conversion 536
9.6 Statements 537
9.6.1 Overview of PAMCONV statements 538
9.6.2 Description of the statements 539
9.6.2.1 CHANGE-TO-SYSTEM-MODE - Switch to system mode 540
9.6.2.2 CLASSIFY-FILE - Classify files according to their convertibility 541
9.6.2.3 CONVERT-FILE - Convert files 547
9.6.2.4 END - Terminate PAMCONV 555
9.6.2.5 MODIFY-CONVERT-FILE-DEFAULTS - Set default values for CONVERT-FILE statement 556
9.6.2.6 MODIFY-LOGGING-OPTIONS - Set logging values 563
9.6.2.7 SHOW-CONVERT-FILE-DEFAULTS - List current default values for CONVERT-FILE statement 564
9.6.2.8 SHOW-LOGGING-OPTIONS - List specified logging options 565
9.7 PAMCONV program execution 566
9.8 Error handling 568
9.9 PAMCONV messages 570
10 PASSWORD Encryption of passwords 571
10.1 Operation and execution 572
10.1.1 Passwords 573
10.1.2 Program execution 575
10.2 Statements 576
10.2.1 Overview of PASSWORD statements 577
10.2.2 Description of the statements 578
10.2.2.1 CONVERT - Encrypt most powerful password of a file 579
10.2.2.2 ENCPASS - Encrypt file password and enter in PASSWORD table 581
10.2.2.3 ENCRYPTD - Encrypt specified file password 582
10.2.2.4 ENCRYPTJ - Encrypt specified LOGON password 583
10.2.2.5 END - Terminate PASSWORD 584
10.2.2.6 HELP - List operands 585
10.2.2.7 JVCONV - Encrypt password for job variable 586
10.2.2.8 MODE - Select encryption routine 587
10.2.2.9 PASSWORD - Encrypt specified file password 588
11 PVSREN Renaming pubsets and volume sets 589
11.1 Prerequisites for PVSREN 590
11.1.1 Prerequisites for the execution of PVSREN 591
11.1.2 Prerequisites for renaming 592
11.1.3 Prerequisites for renaming mirror pubsets 594

11.2 PVSREN operation 595
11.3 Restrictions and reworking 598
11.4 Starting and stopping PVSREN 602
11.5 PVSREN messages 603
11.6 Statements of PVSREN 604
11.6.1 Overview of PVSREN statements 605
11.6.2 Description of the statements 606
11.6.2.1 CHECK-FILENAME-LENGTH - Check length of file and job variable names 607
11.6.2.2 MODIFY-JOINFILE - Modify default catalog ID in user catalog 609
11.6.2.3 MODIFY-LOGGING-OPTIONS - Modify default logging option values . . 610
11.6.2.4 RENAME-PUBSET-OR-VOLUME-SET - Convert pubset notation or rename SF or SM pubset or volume set 611
11.6.2.5 RESTART-RENAMING - Restart conversion or renaming 614
11.6.2.6 RESTORE-LABELS-OF-PUBSET - Restoring the names of a pubset from mirror pubsets in double-point notation 615
11.6.2.7 SHOW-LOGGING-OPTIONS - List current logging option values 616
12 RMS REP Mounting System 617
12.1 User groups and operations 620
12.2 RMS depot 621
12.3 Flexible use of RMS 622
12.4 RMS operation 623
12.4.1 Starting and exiting RMS 624
12.4.2 Inputs to RMS 625
12.5 Function overview 626
12.5.1 Addition of correction packets 627
12.5.2 Loader services 628
12.5.3 Correction management - Notebook 629
12.5.4 Information services 630
12.5.5 Editing 631
12.5.6 RMS management 632
12.5.7 Overview of functions found on the default menu 633
12.6 Interactive application 635
12.6.1 Addition of correction packets 636
12.6.2 Loader services 639
12.6.3 Correction management - Notebook 645
12.6.4 Information services 648
12.6.4.1 Correction - product and REP information 649
12.6.4.2 Correction - global list 650
12.6.4.3 Notebook - product and REP information 651
12.6.4.4 Notebook - global list 652

12.6.4.5 Comparing REP collections 653
12.6.4.6 Character-oriented search 654
12.6.5 Editing 655
12.6.6 RMS management 657
12.6.6.1 Installation 658
12.6.6.2 Startup procedure and basic parameters 659
12.6.6.3 Defining functionality 663
12.6.6.4 Example of a definition file 665
12.6.6.5 DEPOT maintenance 666
12.7 Typical RMS applications 668
12.7.1 Setting up a depot 669
12.7.2 Inputting a new delivery packet into the depot 670
12.7.3 Activating optional REPs 671
12.7.4 Building loaders in interactive mode 672
12.7.5 Building loaders using a batch file 673
12.7.6 Transferring a REP packet to the next user group 674
12.7.7 Rejecting an invalid REP 675
12.8 Statements 676
12.8.1 Generating and editing a software configuration 677
12.8.1.1 CREATE-SW-CONF - Create new software configuration 678
12.8.1.2 MODIFY-SW-CONF - Modify software configuration 680
12.8.1.3 DELETE-SW-CONF - Delete software configuration 682
12.8.2 Optional REP selection 683
12.8.2.1 CREATE-OPT-PACKET - Create optional REP selection packet 684
12.8.2.2 MODIFY-OPT-PACKET - Extend optional REP selection packet 685
12.8.2.3 DELETE-OPT-PACKET - Delete optional packet 686
12.8.3 Loader modifications 687
12.8.3.1 CREATE-MOD-PACKET - Create loader modification element 688
12.8.3.2 MODIFY-MOD-PACKET - Modify loader modification element 689
12.8.3.3 DELETE-MOD-PACKET - Delete loader modification element 690
12.8.4 REPFILE definitions 691
12.8.4.1 CREATE-REPFILE-DEFINITION - Create new loader definitions 692
12.8.4.2 MODIFY-REPFILE-DEFINITION - Modify loader definitions 694
12.8.4.3 DELETE-REPFILE-DEFINITION - Delete loader definition element 696
12.8.5 Creating the system loader file 697
12.8.5.1 SELECT-REPFILE-TO-BUILD - Select system loader definitions 698
12.8.5.2 BUILD-REPFILE - Create and install system loader 700
12.8.6 Transporting correction delivery packets 701
12.8.6.1 INPUT-DELIVERY-PACKET - Accept delivery packet 702
12.8.6.2 MODIFY-DELIVERY-PACKET - Record deviations (notebook) 703
12.8.6.3 CHECK-DELIVERY-PACKET - Check delivery packet 704

12.8.6.4 CREATE-DELIVERY-PACKET - Create delivery packet 705
12.8.7 Information functions 706
12.8.7.1 CREATE-MATRIX-LIST - Create matrix list 707
12.8.7.2 COMPARE-REPFILE - Compare REP files 708
12.8.7.3 CREATE-RMS-OPTIONS - Modify RMS settings 709
12.8.7.4 CALL-FUNCTION - Call module 711
12.8.8 Control statements 712
12.8.8.1 END - End inputs or exit RMS in batch mode 713
12.9 Reference information 714
12.9.1 Uses of the function keys 715
12.9.2 Uses of the P keys 716
12.9.3 JD notation 717
12.9.4 Naming conventions for input files 718
12.9.5 Output files 719
12.9.6 Module list 720
12.10 Statements of RMS before V7.0 722
12.10.1 Examples 724
12.10.2 Description of the statements 726
13 SIR Creation of pubsets 728
13.1 Creating pubsets with SIR 729
13.1.1 Volume sets 730
13.1.2 SM pubsets 732
13.1.3 SF pubsets 736
13.2 Pubset states 739
13.3 Overview of SIR functions and statements 741
13.4 Installing and operating SIR 746
13.5 Copying files with SIR 749
13.6 Statements 750
13.6.1 Overview of SIR statements 751
13.6.2 Input rules 752
13.6.3 Statement editor 754
13.6.4 Description of the statements 755
13.6.4.1 BEGIN-VOLUME-SET-DECLARATION - Set up a volume set 756
13.6.4.2 COPY - Copy files 761
13.6.4.3 CREATE-CATALOG - Define file catalog 765
13.6.4.4 CREATE-IPL-VOLUME - Convert disk to IPL disk 767
13.6.4.5 CREATE-PAGING-FILE - Create paging file 770
13.6.4.6 CREATE-SNAP-FILE - Create snapshot file 771
13.6.4.7 CREATE-VOLUME - Initialize disks 772
13.6.4.8 DECLARE-PUBSET - Define pubset and type of processing 776
13.6.4.9 DELETE-IPL-FACILITY - Delete IPL capability and IPL files of disk 782

13.6.4.10 END - Terminate SIR 783
13.6.4.11 END-VOLUME-SET-DECLARATION - Terminate statement sequence for volume set 784
13.6.4.12 INITIALIZE-PRIVATE-VOLUME - Initialize private disks 785
13.6.4.13 INITIALIZE-PUBLIC-VOLUME - Initialize public disks 787
13.6.4.14 MODIFY-IPL-VOLUME - Modifying an IPL disk 789
14 SMPGEN Conversion of SF pubsets to SM pubsets 792
14.1 Typical scenario 793
14.2 Requirements for operation of SMPGEN 795
14.3 Check function (consistency check) 797
14.3.1 What is checked? 798
14.3.2 Outputs 800
14.4 Creating new SM pubsets 805
14.4.1 Determining quotas and other attributes 808
14.4.2 Restrictions 810
14.4.3 Outputs 811
14.5 Extending an existing SM pubset 812
14.5.1 Restrictions 815
14.5.2 Outputs 817
14.6 Starting and stopping SMPGEN 818
14.7 Statements 819
14.7.1 Overview of SMPGEN statements 820
14.7.2 Description of the statements 821
14.7.2.1 CREATE-SYSTEM-MANAGED-PUBSET - Convert SF pubsets to SM pubset 822
14.7.2.2 MODIFY-SYSTEM-MANAGED-PUBSET - Extending an existing SM pubset 837
14.8 Error behavior 849
14.9 Outputs to screen masks 851
14.10 SMPGEN messages 863
14.11 Attributes of SM pubsets and volume sets 864
15 SPCCNTRL Checking and managing storage allocations on disks 867
15.1 Operating instructions 868
15.2 Statements 869
15.2.1 Overview of SPCCNTRL statements 870
15.2.2 Description of the statements 871
15.2.2.1 BKPT - Interrupt SPCCNTRL routine 872
15.2.2.2 CHECK - Perform allocation checks 873
15.2.2.3 DISPLAY - Direct output to SYSOUT 876
15.2.2.4 END - Terminate SPCCNTRL 883
15.2.2.5 HELP - Describe specified statement or list all SPCCNTRL statements . 884

15.2.2.6 LIST - Direct output to SYSLST 885
15.2.2.7 MODIFY - Modify default values 890
15.2.2.8 PURGE - Remove dead space and delete catalog entries 891
15.2.2.9 TRACE - Identify and output blocks and entries in system catalog and VTOC area of private disk 894
16 TPCOMP2 Comparison of data on tapes 898
16.1 Functions 899
16.2 Starting the program run 900
16.3 Statements 901
16.3.1 Overview of TPCOMP2 statements 902
16.3.2 Description of the statements 903
16.3.2.1 COM - Compare areas of input tapes 904
16.3.2.2 END - Terminate TPCOMP2 905
16.3.2.3 LIM - Terminate tape comparison after a certain number of blocks 906
16.3.2.4 POS - Position magnetic tapes 907
16.3.2.5 RCD - Specify processing of variable-length input records 908
16.3.2.6 STP - Stop TPCOMP2 in the event of discrepancies between magnetic tapes 909
16.4 Using /ADD-FILE-LINK 910
16.5 TPCOMP2 messages 912
17 VOLIN Initialization of disk storage units 914
17.1 VOLIN program execution 915
17.2 Generating BS2000 labels (label generation) 916
17.3 Program execution 918
17.4 Operating instructions 920
17.5 Statements 921
17.5.1 Overview of VOLIN statements 922
17.5.2 Description of the statements 923
17.5.3 Correspondence between ISP and SDF statements 926
18 Related publications 927

Utility Routines

1 Preface

This manual describes the following utility routines for controlling and monitoring the BS2000 operating system:

Name of utility routine	Privilege required	Description in section ...
DPAGE	For certain statements	"DPAGE Outputting and modifying disk files"
INIT	For certain statements	"INIT Initializing (emulated) magnetic tapes"
IORM	Yes	"IORM Dynamic control of I/O resources"
JMP	Yes	"JMP Reconstruction of ENTER commands from the JMS job pool"
JMU	Yes	"JMU Creating and maintaining the SJMSFILE system file"
LMSCONV	No	"LMSCONV Generation and management of libraries"
MSGMAKER	No	"MSGMAKER Processing of BS2000 Message Files"
PAMCONV	No	"PAMCONV Conversion of file formats"
PASSWORD	No	"PASSWORD Encryption of passwords"
PVSREN	Yes	"PVSREN Renaming pubsets and volume sets"
RMS	Yes	"RMS REP Mounting System"
SIR	For certain statements	"SIR Creation of pubsets"
SMPGEN	For certain statements	"SMPGEN Conversion of SF pubsets to SM pubsets"
SPCCNTRL	For certain statements	"SPCCNTRL Checking and managing storage allocations on disks"
TPCOMP2	No	"TPCOMP2 Comparison of data on tapes"
VOLIN	Yes	"VOLIN Initialization of disk storage units"

The utility routines handle three areas:

- volume processing
- file processing
- controlling of I/O resources (devices, controllers, channels, paths)

The utility routines can be assigned to these three areas as follows:

Utility routines for volume processing

DPAGE	Output and modification of data in PAM pages
INIT	Initialization of tapes
PVSREN	Renaming of pubsets and generating of pubsets from pubset mirrors
SMPGEN	Conversion of SF pubsets to SM pubsets
SIR	Creation of pubsets
SPCCNTRL	Checking and management of disk space allocation
TPCOMP2	Comparison of data on tapes
VOLIN	Initialization of disk storage units

Utility routines for file processing

DPAGE	Output and modification of data in PAM pages
JMP	Reconstruction of ENTER commands from the JMS job pool
JMU	Generation and maintenance of the file for stream and job class definitions
LMSCONV	Generation and management of libraries
MSGMAKER	Processing and generation of BS2000 message files
PAMCONV	Conversion of file formats
PASSWORD	Encryption of passwords
RMS	Management, documentation, delivery and mounting of REP packets and preliminary corrections

Utility routines for controlling I/O resources

IORM	Optimizing the workload of I/O resources

1.1 Objectives and target groups of this manual

This manual is intended both for privileged and nonprivileged BS2000 users.

1.2 Summary of contents

This manual describes the utility routines in alphabetical order.

Additional product information

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

1.3 Changes since the last edition of the manual

The following principal changes have been made since the last edition of the “Utility Routines” manual:

- For PAMCONV the CHECK-BLKCTRL-INDICATOR statement has been removed.
- Support for EMC VMAX disk systems was dropped with BS2000 OS DX V1. This means that the creationen of pubsets by means of disk mirror techniques is no longer supported and the CREATE-PUBSET-FROM-MIRROR and SET-NAME-OF-NEW-VOLUME-SET statements have now also been removed from the documentation for PVSREN.
- The FAST statement is no longer supported for the VOLIN utility.

1.4 Notational conventions

For the sake of simplicity and clarity, frequently used names are abbreviated as follows:

- **BS2000 servers** for the servers with /390 architecture and the servers with x86 architecture. These servers are operated with the relevant BS2000 operating system.
- **/390 server** for the Server Unit /390 of the Fujitsu Server BS2000 SE Series
- **x86-Server** for the Server Unit x86 of the Fujitsu Server BS2000 SE Series
- **SE servers** for the Fujitsu Server BS2000 SE Series (Server Units /390 and x86)

In the examples the strings `<date>`, `<time>` and `<version>` specify the current outputs for date, time and version of a software product when the examples are otherwise independent of date, time and version.

The following typographical elements are used in this manual:

For notes on particularly important information

This symbol designates special information that points out the possibility that data can be lost or that other serious damage may occur.

`[ ]` Related publications are referred to in short form throughout the manual. The full titles are listed in the “References” section at the back of this manual.

`Input` Inputs and system outputs in examples are shown in `typewriter font`.

Messages and their meaning

In this manual, messages are represented in most cases only by their message keys. The message key starts with 3 characters (message class) followed by 4 hexadecimal digits.

You obtain information on the meaning of the messages issued with the BS2000 command HELP-MSG-INFORMATION.

You will find the messages using an HTML application on our manual server (URL: *http://bs2manuals.ts.fujitsu.com*) under the current version of BS2000 OS DX instead of in the previous manual "System Messages".

All messages can be found in the respective message file using the software product MSGMAKER (see "MSGMAKER Processing of BS2000 Message Files").

Metasyntax for statements

SDF Format

For the description of the metasyntax in SDF format please refer to the manual „Commands“ [1 (Related publications)].

ISP format

Statement format in the utility routines DPAGE, INIT, PASSWORD, RMS, SPCCNTRL, TPCOMP2 and VOLIN.

ISP syntax representations have been changed to SDF format. However, additional metacharacters are used and declarations made, which are described below.

Representation	Function
WRPASS	Uppercase characters denote constants which must be input exactly as shown.
programname	Lowercase characters denote variables which are replaced with current values on input.
{ YES / NO }	Braces are used to indicate alternatives, i.e. a specification must be selected from the values shown in the brackets. The alternatives are written one below another.
[ ]	Square brackets indicate optional specifications that can be omitted. The entire specification in the brackets must be omitted.
( )	Parentheses are part of operands and must be input with them.
'BLANK'	Means a space (blank).
D2,ALL	Commas separating operands must also be entered.
(filename1),... (vsn1, vsn2, ...)	Three periods mean that the unit in front of the comma can be repeated (possibly only up to a specified maximum value).

Table 1: ISP metasyntax

2 DPAGE Outputting and modifying disk files

Version: **DPAGE V17.0A**

Privileges: **STD-PROCESSING** (for nonprivileged functions)

TSOS (to open a volume)

The DPAGE utility routine enables nonprivileged users, systems support and system programmers to perform the following functions:

- output files in PAM format to SYSOUT (i.e. the terminal in an interactive task, and a file in a batch task)
- output files in PAM format to SYSLST (for high-volume printing)
- modify data contained in a PAM page (2048 bytes) or in the PAM key (16 bytes)

Only systems support is permitted to process volumes.

Note

DPAGE requires each volume opened to have PAM pages format, otherwise the results are unpredictable.

It should be noted that public volumes contain IPL (Initial Program Load) and SVL (Standard Volume Label) records in physical pages 1, 2 and 3.

> ! **WARNING!** Modifying these pages may render the volume unusable. If the SHAREABLE volume is open, modifications on page 2 (SVL) are not accepted during WRITE.

2.1 Support for pubsets

In a system with several pubsets, each pubset has its own TSOSCAT file catalog. Each of these catalogs is uniquely identified by its catalog ID (catid). The catalog ID (catid) is part of the file name.

A file name has the format: :catid:$userid.filename

catid	Catalog identifier. This has a length of 1-4 characters and must be enclosed in colons.
userid	User identification under which the file is cataloged. The user identification must be preceded by a $ sign. It has a maximum length of 8 characters and must be concluded with a period.
filename	Name of the file as entered in TSOSCAT. The file name component, including all periods (designating partial qualification), must not exceed 41 characters.

A fully qualified file name, consisting of catalog ID, user ID and file name, may have a total length of up to 54 characters:

up to 4 characters	Catalog ID
2 character	Colons as delimiters before and after the catalog ID
1 character	$ signifying the beginning of the user ID
up to 8 characters	User ID
1 character	. (period) used as delimiter between user ID and file name
up to 41 characters	File name, including all periods for partial qualification; max. length depends on the length of the catid and userid

2.2 Starting the program run

The program is started with: `/START-DPAGE`

The BKPT statement, or alternatively a slash, can be used to interrupt DPAGE (i.e. to set a breakpoint). Control is

START-DPAGE	Alias: **DPAGE**
VERSION = ***STD** / <product-version>	
,MONJV = ***NONE** / <filename 1..54 without-gen-vers>	
,CPU-LIMIT = ***JOB-RE**ST / <integer 1..32767 *seconds*>	

Alternatively: `/START-EXECUTABLE-PROGRAM FROM-FILE=$DPAGE`

2.3 Statements

Overview of DPAGE statements

Operation	Function
`BKPT / /[bs-cmd]`	Interrupt DPAGE or enter a BS2000 command
`DISPLAY / D`	Output a specific portion of one or more PAM pages to SYSOUT
`EDT / @`	Call the file editor EDT as a subroutine
`HALT / H / END / E`	Terminate the DPAGE routine
`MODIFY / M`	Change the contents of a page or the key in an internal work area
`OPEN / O`	Open a file or a volume
`PRINT / P`	Output a specific portion of one or more PAM pages to SYSLST
`READ / R`	Read a page into an internal work area
`WRITE / W`	Write the page currently in the internal work area back to the file

Formats

The following terms are used in the description of the statements:

page	Decimal number of up to 10 digits in the range 1 - 2147483647.
byte,byte1,byte2	Decimal number of up to 4 digits in the range 1 - 2048. *Exception* After opening 4K-formatted volumes, a decimal number in the range 1-4096.
Kn	The letter K followed by an integer in the range 1-16.

The page number specified in the statements refers to the Physical Half Page Number (PHP = 2048 bytes).

The page number however refers to the logical 4k block number (contains 2PHP = 4096 bytes) in the following cases:

- a 4K-formatted volume has been opened
- a 4K file has been opened on a volume without a PAM key (NK2, NK4)

2.3.1 BKPT - Interrupt DPAGE

The BKPT statement, or alternatively a slash, can be used to interrupt DPAGE (i.e. to set a breakpoint). Control is returned to DPAGE via `/RESUME-PROGRAM`.

"/" and "bs-cmd" may be followed by any BS2000 command.

Format

{ /[bs-cmd] / BKPT }

Operands

bs-cmd
BS2000 command

2.3.2 DISPLAY - Output page to SYSOUT

The DISPLAY statement outputs one or more pages, or parts thereof, to SYSOUT.

Format

DISPLAY / D
page / page1-page2 / page-$ / * [,byte / ,byte1-byte2 / ,K]

Operands

page
Specifies the page of the file (or the volume) to be displayed.

page1-page2
Specifies a range of pages which is to be displayed (page1 < page2).

page-$
Specifies that all pages, starting at “page” and continuing through to the end of the file (or volume), are to be displayed.

*
Specifies that the page currently in the internal work area is to be displayed.

byte
Specifies the byte to be displayed.

byte1-byte2
Specifies the byte range to be displayed (byte1 < byte2). Up to 2048 bytes can be used for byte2. For 4K-formatted volumes up to 4096 bytes.

K
Only the PAM key is to be displayed.

Example

```
DISPLAY 1,1-224
*OPEN TEST
OPEN COMPLETED
*DISPLAY 1,1-224
 PAGE:0000000001            PAMKEY:  57739BDE  01000001  00000138  00010006
 001 --> (0001)  7CD7C602  02F8F5C1  D7C1D4C5  C4C9E340   @PF  90ATEST
 011 --> (0017)  000057E8  000057E8  F8F7F0F2  F2F50001      Y   Y870225
 021 --> (0033)  00000000  00000006  00000001  00000000
 031 --> (0049)  00010000  00000000  00000000  00000000
 041 --> (0065)  00D4010D  00000000  0000000C  027CD7C6    M           @PF
 051 --> (0081)  E5F9F04B  C1F0F040  40400000  00000000   V90.A00
 061 --> (0097)  00000000  00000000  D7C1D4C5  C4C9E340          TEST
=071 --> (0113)  40404040  40404040  40404040  40404040
 091 --> (0145)  40000000  00000000  00000000  00000000
 0A1 --> (0161)  00000000  00000000  40404040  40404040
 0B1 --> (0177)  40404040  F1F9F8F7  00000000  00000000       1987
 0C1 --> (0193)  00000000  00000000  00000000  E5F2F100              V21
 0D1 --> (0209)  00000000  00000000  6CD9D6D6  E3404040           %ROOT
```

¡ Output "=071"at the beginning of a line means that the following lines that are not explicitly output have the same contents as 071 and are also filled with blanks.

In the example, line 71 is highlighted ("=71") and line 81 is missing, i.e. line 81 is also filled with blanks.

2.3.3 EDT - Call EDT

The EDT statement calls the file editor EDT.
In interactive mode EDT full screen mode is set.
In batch mode the EDT statements are read from SYSDTA.

The EDT statement @RETURN or @HALT returns control to DPAGE – in full screen mode this is also the case when the K1 button is pressed.

The internal data of EDT (working files, variables) is retained for a possible further call of the EDT. It is released only when DPAGE is terminated.

Format

EDT / @

2.3.4 END / HALT - Terminate DPAGE

The END or HALT statement terminates DPAGE.

The open file is closed.

Format

HALT / H / END / E

> If any error occurred in DPAGE, e.g. syntax error during input of a command, DPAGE is terminated with TERM UNIT=STEP. In this case, error handling is possible in the procedures.

2.3.5 MODIFY - Modify contents of page

This statement modifies the contents of the page currently in the internal work area (see READ).
The original page in the file (on the volume) is not affected here (see WRITE).

Format

MODIFY / M
[byte1 / Kn1] [,X'hex-string' / ,nnX'hex-string' / ,'character-string' / ,nn'character-string' / ,C'character-string' / ,nnC'character-string'] [,byte2 / ,Kn2]

Operands

First operand

byte1
Specifies the location in the PAM page (1..2048) where the text specified in the second operand is to replace the original text. All bytes before byte 1 remain unchanged.

Kn1
Specifies the position in the PAM key (1-16) where the text specified in the second operand is to replace the original text. All bytes before Kn1 remain unchanged.

The default value is 1. The text specified in the second operand changes the original contents of the page, starting with the first byte.

Second operand

X'hex-string' / **nnX'hex-string'** / **'character-string'** / **nn'character-string'** / **C'character-string'** / **nnC'character-string'**
Specifies the text that is to replace the original text. If byte 2 is not specified, the new text replaces the old text for the length of the new text, starting with byte 1.
“nn” is an integer which specifies a repetition factor for the specified string.
No distinction is made between uppercase and lowercase if C'...' is specified.

Default value: no text is changed.

The text must not extend beyond the PAM page or PAM key.

Third operand

byte2
Specifies the location in the original PAM page where the text which is to be added to the new text begins, regardless of whether the new text is shorter, longer or equally long.

Kn2
Specifies the location in the original PAM key where the text which is to be added to the new text begins.

The locations which the first and thid operands specify must both be located on the PAM page or in the PAM key.

Modifications in the first 8 bytes of the PAM key will not be transcribed back to the file. Furthermore, in the case of files without a PAM key, the first 12 bytes of a page (control block field) will not be transferred to the file.

Example

The original page is assumed to contain: `'ABCCATDOGXX...'`
and the original key is assumed to contain: `'1234DEFGO...'`

```
MODIFY 5, 'OW', 7
```

Changes the entire page to: `'ABCCOWDOGXX....'`

```
MODIFY 5, 'OW'
```

Changes the page, like the instruction in example a), to: `'ABCCOWDOGXX...'`

```
MODIFY , 3X'C8C1', 10
```

Changes the page to: `'HAHAHAXX...'`

The page will be padded at the end with 3 bytes of hexadecimal zeros (X'000000').

```
M K1, 16x'00'
```

Sets the key to hexadecimal zero.
After the instruction WRITE only bytes 9..16 of the key are written back to the disk. Bytes 1..8 are not changed.

2.3.6 OPEN - Open file

The OPEN statement permits an open file or volume to be closed and another one to be opened. Only systems support is authorized to open a volume.

Format

OPEN / O
{ filename [,INOUT / ,INPUT/ ,PHYSICAL] / 'vsn' [,[device-type] [**E**XCLUSIVE / ,**S**HAREABLE]] }

Operands

Opening a file

filename
Fully qualified file name or name of a file generation group. The file protection offered by DMS takes effect when the file is opened (e.g. password, external access, read/write access).

INOUT
The file is opened for reading and writing.

INPUT
The file is opened for reading only. WRITE statements lead to a PAM write error.

PHYSICAL
The file is opened for reading only. WRITE statements lead to a PAM write error. If the file is encrypted, the subsequent output statements result in the file content being output in encrypted format.

Opening a volume

'vsn'
The volume serial number, up to six characters long, of a volume (either public or private) is enclosed in single quotes.

device-type
Defines the volume device type.
See the “System Installation” manual [7 (Related publications)] for possible specifications.

EXCLUSIVE
The volume is opened exclusively for one task.

SHAREABLE
The volume is opened as a public volume.

Example

```
OPEN 'ABCDEF'
```

Volume 'ABCDEF' is opened exclusively if it is available.

2.3.7 PRINT - Print page

The PRINT statement prints portions of a page or whole pages via the SYSLST file.

Format

PRINT / P
page / page1-page2 / page-$ / * [,byte / ,byte1-byte2 / ,K]

Operands

This statement is identical with the DISPLAY statement except for the fact that output is to SYSLST. The meanings of the operands are given on "DISPLAY - Output page to the SYSOUT file".

2.3.8 READ - Read page

The READ statement reads a page into the internal work area.

The data read in thus exists twice: once in the original PAM page and once in the internal work area.

The MODIFY statement can be used to modify the contents of the PAM page stored in the internal work area. With the WRITE statement, the data from the internal work area may be written back to the original area.

The internal work area is cleared by means of the HALT statement.

Format

READ / R
[page]

Operands

page
Specifies the logical page number to be read. If no page is specified, the next page is read.

Example

```
READ 7
READ 8
READ 3
```

This sequence of statements will cause the DPAGE routine to read pages 7, 8 and 3 consecutively. It should be noted that only one internal work area is provided.

2.3.9 WRITE - Write internal work area back to file

The WRITE statement writes the page currently in the internal work area back to the file (or volume).

The page may have been modified by a MODIFY statement.

Format

WRITE / W

Example

```
READ 1
MODIFY 10,X'FF'
WRITE
```

Page 1 is read to memory.
X'FF' is assigned to the tenth byte.
After modification, page 1 is written back to the file.

2.4 DPAGE messages

```
CLOSE ERROR, ERROR-CODE: xxxx
```

Meaning
Error on closing a file or disk. xxxx is the DMS error code.

Response
For an analysis of the error code xxxx, consult the manual "DMS Macros" or issue the BS2000 HELP command xxxx.

```
ERROR IN EDT-CALL
```

Meaning
Program error in the EDT call. EDT could not be started properly.

```
INVALID BS2000 command
```

Meaning
The specified BS2000 command is invalid, or an error occurred in the CMD macro call.

```
INVALID COMMAND
```

Meaning
Invalid command, no action.

```
INVALID OPERAND, COMMAND REJECTED
```

Meaning
Invalid operand specification, command ignored.

```
NO PAMKEY AVAILABLE
```

Meaning
The file/disk has no PAM key.

```
OPEN COMMAND MUST BE GIVEN FIRST, COMMAND REJECTED
```

Meaning
Wrong statement sequence.

Response
Enter OPEN first.

```
OPEN ERROR, ERROR-CODE: xxxx
```

Meaning
Error on opening a file or disk. xxxx is the DMS error code.

Response
For an analysis of the error code xxxx, consult the manual "DMS Macros" or issue the BS2000 HELP command xxxx.

```
OPEN VOLUME RESTRICTED TO SYSTEM ADMINISTRATOR
```

Meaning
Only systems support is allowed to open a disk.

```
PAM-READ ERROR, ERROR-CODE: xxxx
```

Meaning
Error on reading a PAM page. xxxx is the DMS error code.

Response
For an analysis of the error code xxxx, consult the manual "DMS Macros" or issue the BS2000 HELP command xxxx.

```
PAM-WRITE ERROR, ERROR-CODE: xxxx
```

Meaning
Error on writing a PAM page. xxxx is the DMS error code.

Response
For an analysis of the error code xxxx, consult the manual "DMS Macros" or issue the BS2000 HELP command xxxx.

```
READ COMMAND MUST BE GIVEN FIRST. PAGE NOT DISPLAYED.
```

Meaning
DISPLAY was entered, but no page had been read in.

Response
Enter READ first.

```
READ COMMAND MUST BE GIVEN FIRST. PAGE NOT MODIFIED.
```

Meaning
MODIFY was entered, but no page had been read in.

Response
Enter READ first.

```
READ COMMAND MUST BE GIVEN FIRST. PAGE NOT PRINTED.
```

Meaning
PRINT was entered, but no page had been read in.

Response
Enter READ first, or enter PRINT with page specification.

```
READ COMMAND MUST BE GIVEN FIRST. PAGE NOT WRITTEN
```

Meaning
WRITE was entered, but no page had been read in.

Response
Enter READ first.

```
REQUEST MEMORY ERROR
```

Meaning
Error on requesting memory via the REQM macro.

SEARCHED STRING NOT FOUND

Meaning
The desired string could not be found.

`VOLUME NOT FOUND`

Meaning
DPAGE could not find the specified volume.

Response
Enter correct volume serial number or contact the systems support.

`ERROR OCCURED IN DPAGE-RUN: TERM UNIT=STEP`

Meaning
An error has occurred during the DPAGE run. Jump to next STEP, ENDP, LOGOFF or to error processing (e.g. IF-BLOCK-ERROR).

Response
Check SYSOUT and SYSLST logs and modify and repeat procedure if necessary.

3 INIT Initializing (emulated) magnetic tapes

Version: **INIT V21.0A**

Privileges: **TAPE-PROCESSING** (for nonprivileged functions)
TAPE-PROCESSING (for privileged functions)

The INIT utility routine initializes magnetic tapes (magnetic tape cartridges, MTCs) and emulated magnetic tapes.

Initialization is the process of writing a volume label (VOL1) and possibly two file labels (HDR1 and HDR2 of a dummy file) at the logical start of the volume. The logical start of the volume is the position from which BS2000 can begin writing data. This is not necessarily the physical start of the volume.
In the case of magnetic tape cartridges, labels are never compressed when written, even if write access with compression has been specified (volume type TAPE-C4). The volume label (VOL1) is always written unencrypted, even when encrypted write mode is specified.

Up to 16 volumes can be initialized with one statement.

With all INIT or LIST functions, you can select a defined device that supports the corresponding volume type.

In order to prevent data being overwritten by mistake, each volume is checked for existing labels before new labels are created. The contents of existing labels are output to SYSOUT for scrutiny by the user. This means that the user has an opportunity to abort initialization.The read-before-write check is only not performed on magnetic tapes when the device management has already recognized the volume as empty and the INIT operand NEW has been specified.

If a volume is checked and a VOL1, HDR1 or HDR3 label is found to contain an access restriction (access pointer, release date, read or write password, “read-only” flag), a message to this effect is output.

The functions “read labels” (LIST statement) and “write new labels or tape marks” (INIT statement) are not executed unless the user has the “TAPE-ADMINISTRATION” privilege.

The INIT utility routine offers the following functions:

1. For magnetic tapes:
   - Write the volume header label VOL1 and the file header labels HDR1 and HDR2 (followed by two tape marks).

     Contents of tape: VOL1-HDR1-HDR2-TM-TM
   - Write only the volume header label VOL1 (followed by two tape marks).

     Contents of tape: VOL1-TM-TM
   - Write two tape marks at the logical beginning of the volume. No labels are written.

     Contents of tape: TM-TM
   - Dump the contents of VOL1, HDR1, HDR2 and HDR3 labels, if any, to SYSOUT or the console. New labels are not written.
   - Format the magnetic tape, if allowed by the volume and the tape device.
2. General:
   - Activating or deactivating special functions. The setting remains valid for all INIT and LIST statements for the duration of the program session or until reset.
   - Output an outline description of statements and operands.

- Switch input/output to the console after the program is started with an ENTER task.

3. Security functions:
   - Specifies a check volume serial number. A magnetic tape is not processed unless the VSN in its VOL1 label tallies with the check VSN.
   - Erase the entire contents of the volume before writing new labels (DSE = Data Security Erase).

     > ¡ In this context, erasure means that the volume is overwritten with a devicedependent deletion pattern. Several newer devices implement deletion only on a logical level by deleting the file management information on the volume. It may nevertheless still be possible to recover the information of earlier records using special devices and technically complex procedures. The only way to be absolutely certain that no unauthorized user reads any residual information left on the volume is to physically destroy the volume.

   - Abort initialization if a read-before-write check finds an access restriction.
   - Exclude special characters not defined in DIN 66003 from use in labels.

3.1 Operating modes

The functions of the INIT utility routine can be executed in two operating modes: Normal mode or console mode.

3.1.1 Normal mode

This is the default operating mode when the INIT program is started with `/START-INIT`.

START-INIT	Alias: **INIT**
VERSION = ***STD** / <product-version>	
,MONJV = ***NONE** / <filename 1..54 without-gen-vers>	
,CPU-LIMIT = ***JOB-RE**ST / <integer 1..32767 *seconds*>	

To guarantee compatibility with earlier versions, the follwoing command is still supported:

```
/START-EXECUTABLE-PROGRAM FROM-FILE=$INIT
```

Once the program is started, messages are output via SYSOUT. The system awaits inputs from SYSDTA.
The program runs as described in the section “Program run”.
Statements to the INIT routine can be:

- entered at the terminal in interactive mode
- contained within a command procedure (reassignment from SYSDTA to SYSCMD)
- contained in an ENTER job (SYSDTA reassigned by default to SYSCMD)

Examples

Command procedure:

```
/BEGIN-PROCEDURE
/ASSIGN-SYSDTA TO-FILE=*SYSCMD
/START-INIT
...    (INIT statements)
END
/ASSIGN-SYSDTA TO-FILE=*PRIMARY
/END-PROCEDURE
```

ENTER job:

```
/SET-LOGON-PARAMETERS
/START-INIT
...    (INIT statements)
END
/EXIT-JOB
```

3.1.2 Console mode

In this operating mode, the INIT utility is controlled by means of a dialog which takes place on the console. For that purpose an ENTER job must be sent from the console to start the INIT program and OPTION CONS used to direct outputs to the console.

```
/SET-LOGON-PARAMETERS
/START-INIT
OPTION CONS
/EXIT-JOB
```

When the job is started, all messages and inputs are handled via the console, with the task sequence number (TSN) being specified for each operation. The program runs as described in section “Program run”.

The OPTION CONS statement is valid only within a ENTER job; it is always rejected in interactive mode.

Example

```
<tsn>.INIT TAPE-C4,VSN=MBK001

<tsn>.LIST TAPE-C4
```

<tsn> is the task sequence number

> If you will install a SOLIS2 delivery in the actual HOME-PVS, the INIT must be terminated (if loaded in BATCH Mode) by <tsn>.END, before the installation will be started.

3.2 Program run

Typical logs are used below as examples to demonstrate how a magnetic tape is initialized.

Messages marked `(OUT)` are output to SYSOUT in normal mode. Those preceded by `(MSG)` are always output to the console.

3.2.1 Program start

Example

```
(IN)  /start-init                                          1.
(OUT) % ...                                                2.
(OUT) %  NVI0000 PROGRAM INIT READY                        3.
(OUT) %  NVI0001 ENTER COMMAND. (END = TERMINATE INIT)     4.
```

1. The INIT program is started.
2. Various load messages are output.
3. INIT signals ready ...
4. ... and requests statement input.

3.2.2 Initializing a magnetic tape (example)

```
(OUT) % NVI0001 ENTER COMMAND. (END = TERMINATE INIT)
(IN) INIT TAPE-C4,VSN=TAPE1,ERASE                                    1.
(MSG) %  NKVT013 MOUNT TAPE '*SCRAT' ON DEVICE 'T9';                 2.
                (USE='SPECIAL',WR='UNDEF',TYPE='TAPE-C4',
                INIT T-C4,VSN=TAPE1).
               '(ETX = YES; MN; N = NO )'?
(IN)  0ABC.T7
(OUT)  % NVI0003 LABELS ON TAPE (ISO7):                              3.
(OUT)  % NVI0004 VOL1 LABEL:
             'VOL1TAPE01                              '
               '                                       1'
(OUT)  % NVI0004 HDR1 LABEL:
                'HDR1SAMPLE.FILE   TAPE010001000100010'
                '0 92104 92104 000000BS2000              '
(OUT)  % NVI0004 HDR2 LABEL:
                'HDR2V8000102044 0                P     '
                '      .C..04                           '
(OUT)  % NVI0004 HDR3 LABEL:
                'HDR3TSOS    SAMPLE.FILE                 '
                '                .............00         '
(OUT)  %  NVI0007 OVERWRITE TAPE?                                    4.
                 REPLY (YES=YES N=NO)
(IN)   YES
(OUT)  %  NVI0208 DATA SECURITY ERASE STARTED                        5.
(OUT)  %  NVI0209 DATA SECURITY ERASE COMPLETED
(OUT)  %  NVI0210 INITIALIZATION OF TAPE 'TAPE1'                     6.
                 ON DEVICE 'T7' COMPLETED
(OUT)  %  NVI0001 ENTER COMMAND. (END = TERMINATE INIT)              7.
```

1. To initialize a volume, enter INIT with a unique valid volume type (in this case with ERASE is specified (data security erase)).
2. Message output via the console requests the operator to mount the volume. Once the volume is mounted (on device T7 in this case), the operator confirms by specifying the task sequence number (TSN). If the volume is mounted on the device specified in the MOUNT request (T9), the mnemonic device name (T7) can be omitted.
3. The volume is read and the labels, if any, are output. If the labels are written in ISO7 code, the output includes a message to this effect. In this example, the contents of the labels are spread over a number of lines and are delimited by quotation marks. In normal use, the contents of the labels are output as a single string; blanks at the end of the string are not included.

   If the volume has no volume label (VOL1) or the label is invalid, explanatory messages are issued instead of the labels.

   Empty volume:

   ```
   (OUT) % NVI0102 NO LABELS ON TAPE. TAPE EMPTY
   ```

   Tape marks at the beginning of the volume:

   ```
   (OUT) % NVI0103 NO LABELS ON TAPE. TAPE MARKS READ
   ```

   Error when reading VOL1 label :

   ```
   (OUT) % NVI0104 I/O-ERROR WHILE READING LABELS
   ```

4. This is followed by the question whether or not to overwrite the volume. This is the last opportunity to prevent the volume being overwritten. If the answer is affirmative, the word YES (for overwrite) must be typed in full. For safety's sake, abbreviations are rejected. All other entries (i.e. do not overwrite) are interpreted as negatives.
5. If data security erase was requested by the ERASE parameter, the data on the volume is erased. This procedure can take several minutes or hours, depending on the device and the volume; messages are output to indicate the start and end of the process.
6. After deletion, the new labels are written and a message is output indicating that initialization is completed. If initialization is unsuccessful, a message is output indicating the source of the problem and initialization is aborted.

   ```
   (OUT) % NVI0309 TAPE WITH VSN ' ' CANNOT BE USED.
                   INITIALIZATION ABORTED
   ```

7. The system prompts for the next statement.

3.2.3 Program termination

```
(OUT)  %  NVI0001 ENTER STATEMENT. (END = TERMINATE INIT)
(IN)   END                                                  1.
(OUT)  %  NVI0011 PROGRAM INIT TERMINATED NORMALLY          2.
```

1. The END statement is entered.
2. The INIT program is terminated and a message indicating correct termination is output.

3.2.4 Problems in volume initialization

If an error occurs in the read-before-write check that precedes label updating, when data is erased or when labels are updated, the device error handling system may output messages to the console. These messages contain useful information indicating the source of the error. If a message requires an answer, the possible answers are shown in the message text. The response must be entered via the console and must include the task sequence number. For example, if the answer required is „NO“, the following entry is required:

```
<tsn>.NO
```

<tsn> is the current task sequence number.

3.3 Statements

Entering INIT statements

INIT reads statements from the system file SYSDTA (by default, from the terminal in interactive mode) or from the console (in CONSOLE mode).

An INIT statement consists of the operation name (INIT, LIST, etc.) plus operands with operand values, as applicable. If a statement requires more than one operand, the operands are separated by commas. Blanks are not permissible between the operand and the operand value or values.

An input line cannot consist of more than 72 characters. Excess characters are truncated without warning. Leading blanks are ignored.

If necessary, an INIT statement can be continued on a continuation line or lines. A continuation character (hyphen: "-") must be set at some point in the statement to indicate the presence of continuation lines. INIT then prompts for continuation of statement input:

```
(OUT) % NVI0002 ENTER ADDITIONAL OPERANDS
```

All characters coming after a blank following the continuation character are ignored (line comment). If characters other than blanks come immediately after a hyphen, the hyphen is not recognized as the continuation character.

Leading blanks in a continuation line are ignored. The operands specified in a continuation line are appended to the preceding line at the point at which the continuation character occurs.

Even if it extends over a number of continuation lines, an INIT statement cannot consist of more than 240 characters (including permissible blanks, excluding leading blanks and character after a continuation character).

Example

Statement parts		Comment
(IN) INIT 3	TAPE-C4,-	INITIALIZE 3 MAGNETIC TAPE CARTRIDGES
(IN)	VSN=(VSN01,-	VSN 1
(IN)	VSN02,-	VSN 2
	VSN03),-	VSN 3
(IN)	UNIT=A1	
(IN)		

This is equivalent to the following input line:

```
INIT 3 TAPE-C4,VSN=(VSN01,VSN02,VSN03),UNIT=A1
```

If an asterisk (*) appears in the first column of an input line, the contents of the line in question are not evaluated. By this means, comments can be added to procedures in which the INIT utility routine is called or to a tracer listing.

Overview of INIT statements

Statement	Function
END	Terminate the program.
HELP	Output an outline description of statements and operands
INIT	Write the volume header label VOL1 and the file header labels HDR1 and HDR2 or write two tape marks. Optional formatting of the volume before writing new labels is possible if the volume type permits it. The system privilege TAPE-ADMINISTRATION is required.
LIST	Output the contents of existing VOL1, HDR1 , HDR2 and HDR3 labels. The system privilege TAPE-ADMINISTRATION is required.
OPTION	Activate and deactivate special functions; the setting remains valid for all INIT and LIST statements throughout the program session or until reset.

3.3.1 INIT - Initialize magnetic tape

The INIT statement writes volume labels to magnet tapes or magnet tape cartridges. The volume type and either the VSN to be written or the initialization mode in which only tape marks are written are specified.

The user can also choose to define:

- the number of volumes to be listed
- a particular volume
- a specific device
- the code for the labels (ISO7 or EBCDIC)
- the way in which the volume is handled after initialization

Before writing new labels, the INIT routine checks for existing labels. If labels can be read, they are output so that they can be checked. If OPTION CONS is set, the list is output to the console. If this option is not set, the list is output to SYSOUT.

> The INIT function is executed only if the task has the system privilege TAPE-ADMINISTRATION.

Format

INIT
[no] voltyp { ,VSN={ vsn / (vsn1,vsn2,...vsn16) / (initval) / *} [,OWN={ name / * }][,ZERO][,ISO7] / ,WTM } [,UNIT = mn] [,NEW / CHECK = vsn] [,REW / RUN] [,ERASE] [,FORMAT]

Operands

no
Number of volumes to be listed with this statement.
Permissible values: 1, 2, ..., 16.

> The operands “no” and “voltyp” must be separated by at least one blank.

voltyp
Defines the volume type. Defines the volume type, and thus also the recording mode.

You will find the volume types supported in the current in BS2000 version in the “System Installation” manual [7 (Related publications)]. For compatibility reasons, INIT also supports a few more volume types. You can ascertain the volume types accepted by INIT by means of the HELP statement (see "HELP - Outline description of INIT statements").

> ¡ The keywords for volume types can be abbreviated as long as they remain unique, e.g. T-C4 = TAPE-C4. As new volume types are introduced, however, an abbreviation which was originally unique may become ambiguous. There are no guaranteed abbreviations.

There is no default value for this operand; the value must always be specified by the user.

```
INIT 3 TAPE-C4,VSN=(VSN01,VSN02,VSN03),UNIT=A1
```

Volume serial number which is to be entered in the VOL1 label (no default).If two or more volumes are to be initialized with a single statement, the user must specify either a corresponding number of VSNs or an initial value.

VSN=vsn
Single volume serial number.
A single volume is to be initialized. Consequently, the “no” operand must be either 1 or not specified.
Permissible values: Max. 6 characters (A..Z, 0..9, #,$,@).

The special characters #, $, @ are permitted by INIT but must not be used in labels conforming to DIN 66029, DIN 66003. If OPTION DIN is active, these special characters are rejected.

VSN=(vsn1,vsn2,...,vsn16)
Two or more volumes are to be initialized with the specified VSNs. The number of VSNs must tally with the value specified for the “no” operand.
Permissible values: each value must be a valid VSN (see above).

VSN=(initval)
Two or more volumes, beginning with the VSN initval, are to be initialized. After every successful initialization, the VSN is automatically incremented by 1. The number of volumes must be specified in the “no” operand.

Permissible values:
initval must be a valid VSN and must contain at least one decimal digit. The number specified as the initial VSN is read from left to right. The initial value of the VSN to be generated is the first sequence of decimal digits found. If initval contains more than one sequence of digits separated by other characters (e.g. letters), only the string of digits furthest left is used as the initial value.

The number of digits in the initial value identified in this way must also be able to accept the highest VSN; if it cannot, the statement is rejected.

Examples

```
INIT 5 TAPE-C4,VSN=(A12B12)
```

The statement initializes volumes with the VSNs A12B12, A13B12, A14B12, A15B12 and A16B12.

```
INIT 16 TAPE-C4,VSN=(A1B)
```

The statement is rejected, because only single-digit decimal numbers can be formed.

VSN=*
The existing VSN is to be used.
If the volume does not have a readable VOL1 label with a valid VSN, a message to this effect is output and no new labels are written.

OWN
Valid only if WTM is not specified. Name of the owner to be entered in the VOL1 label.

OWN=name
Name to be entered in the VOL1 label.
Permissible values: Max. 8 characters (A...Z, 0...9, #, $, @, ., -) Default value:Blanks are entered if nothing is specified.

OWN=*
If an owner's name already exists in the VOL1 label, it is to be transferred to the new VOL1 label. If the volume does not have a legible VOL1 label with a valid owner's name (blanks are also a valid name), a message to this effect is output and no new labels are written.

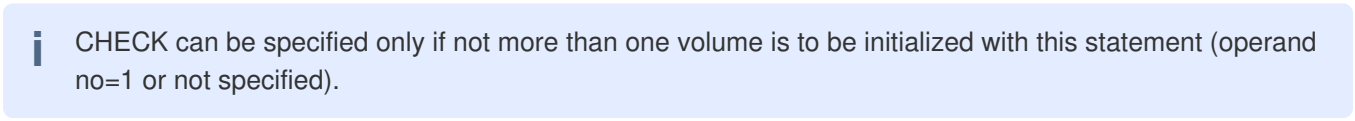
ZERO
In the VOL1 label (byte 11), the printable character 0 is entered as the access flag. If this operand is not specified, a blank is entered as the access flag.

ISO7
The labels are to be written in ISO 7-bit code.

WTM
Write two tape marks at the logical beginning of the volume. No labels are written.

UNIT=mn
Mnemonic device name for seizure of a particular device. A volume is to be mounted on the specified device. Unsuitable devices, i.e. devices that cannot process volumes of the specified type, are rejected.

CHECK=vsn
Specifies a check volume serial number.
A volume is not processed unless the VSN in the VOL1 label tallies with the specified VSN. This mechanism prevents the wrong volume from being overwritten by mistake.
If the CHECK operand is specified and the requested volume is already mounted on the correct device, no MOUNT request is sent to the operator and the volume is initialized immediately.
Permissible values: Max. 6 characters (A..Z, 0..9, #,$,@).
The special characters #, $, @ are permitted by INIT but must not be used in labels conforming to DIN 66029, DIN 66003. If OPTION DIN is active, these special characters are rejected.


CHECK can be specified only if not more than one volume is to be initialized with this statement (operand no=1 or not specified).

NEW
The effect of this operand depends on whether the volume already contains data or if it has been recognized as empty by the tape monitor of the operating system.
If the volume is empty, both the read-before-write check of the labels and the OVERWRITE query are suppressed.
If a volume already contains data, the read-before-write check of the labels is implemented without error handling.

If the NEW operand is not specified, INIT always attempts to read existing labels. This incorporates full error handling if the tape monitor of BS2000 has not already recognized the tape as empty during mounting. For safety's sake, INIT attempts to read without error handling in this event.

REW
Specifies that the magnetic tape or magnetic tape cartridge is to be rewound after listing but not unloaded. If a single volume is listed, REW is set by default.

RUN
Specifies that the magnetic tape or magnetic tape cartridge is to be rewound and unloaded after listing. If two or more volumes are to be listed with a single statement (operand
no=2..16), RUN is set by default.

ERASE
All the data on the volume is to be erased before the labels are updated.

> ¡ Depending on the device and the volume, this procedure may take several minutes or hours.

If this operand is not specified, only the new labels are written at the beginning of the volume and the logical end of volume is marked by means of two tape marks. The original contents of the volume behind these marks, however, is retained, and under certain circumstances it may still be readable (see note under “Security functions” on "INIT Initializing (emulated) magnetic tapes").
If INIT can read a valid VOL1 label, this label is retained after erasure. If not, everything from the beginning of the volume onward is erased.

FORMAT
The volume is formatted, if the volume type and the device permit this. If not, an explanatory message is displayed, the operand is ignored and the initialization procedure continues.

> ¡ It may take several minutes to format a volume.

Examples for writing labels

1. Initializing a volume (specifying minimum parameters):

   ```
   INIT TAPE-C4,VSN=VSN001
   ```

2. Initializing
   - three volumes
   - with one statement
   - with explicitly specified VSNs
   - with special characters in the VSNs:

   ```
   INIT 3 TAPE-C4,VSN=(VSN#00,VSN$00,VSN@00)
   ```

3. Initializing
   - 8 volumes
   - with one statement
   - with consecutive VSNs:

   ```
   INIT 8 TAPE-C4,VSN=(VSN001)
   ```

   The VSNs assigned to the volumes are VSN001, VSN002, ...,VSN008.

4. Initializing
   - a virgin volume
   - on a specified device (with mnemonic name M1):

   ```
   INIT TAPE-C4,VSN=MBK01,UNIT=M1,NEW
   ```

5. Initializing
   - a particular volume (with volume serial number VSN001)
   - on a particular device (with mnemonic name T1)
   - in ISO7 bit code
   - unloading it after initialization
   - with entry of an owner identifier
   - with data security erase
   - with entry of “0” as access code:

   ```
   INIT TAPE-C4,VSN=TAPE01,CHECK=VSN001,UNIT=T1,ISO7,RUN,-
   OWN=RZ#A0001,ERASE,ZERO
   ```

Examples for writing two tape marks

1. Writing two tape marks at the beginning of a volume, specifying minimum parameters:

   ```
   INIT TAPE-C4,WTM
   ```

2. Writing two tape marks at the start of
   - three virgin volumes
   - with one statement:

   ```
   INIT 3 TAPE-C4,WTM,NEW
   ```

3. Writing two tape marks at the start of
   - a particular volume
   - on a particular device (with mnemonic name M1)
   - and with data security erase:

   ```
   INIT TAPE-C4,WTM,UNIT=M1,ERASE
   ```

4. Writing two tape marks at the start of
   - a particular volume (with volume serial number VSN001)
   - on a particular device (with mnemonic name M1)
   - with data security erase
   - and remove after initialization:

   ```
   INIT TAPE-C4,WTM,UNIT=M1,CHECK=VSN001,RUN
   ```

3.3.2 LIST - Read and output magnetic tape labels

The LIST statement is used in the form described below to read and output volume labels.The user can choose

- the number of volumes to be listed
- a particular volume
- and a particular device.

If OPTION CONS is set, the list is output to the console. If this option is not set, the list is output to SYSOUT.

> The LIST function is executed only if the task has the system privilege TAPE-ADMINISTRATION.

Format

LIST
[no] voltyp [,VSN = vsn / (vsn1,vsn2,...,vsn16) / (initval)] [,UNIT = mn] [,REW / RUN]

Operands

no
Number of volumes to be listed with this statement.
Permissible values: 1, 2, ..., 16.

> The operands “no” and “voltyp” must be separated by at least one blank.

voltyp
Defines the volume (volume type). Defines the volume type, and thus also the recording mode.

You will find the volume types supported in the current in BS2000 version in the “System Installation” manual [7 (Related publications)]. For compatibility reasons, INIT also supports a few more volume types. You can ascertain the volume types accepted by INIT by means of the HELP statement (see "HELP - Outline description of INIT statements").

> The keywords for volume types can be abbreviated as long as they remain unique, e.g. T-C4 = TAPE-C4. As new volume types are introduced, however, an abbreviation which was originally unique may become ambiguous. There are no guaranteed abbreviations.

There is no default value for this operand; the value must always be specified by the user.

VSN
The labels of the volume with the specified volume serial number are to be output. To list two or more volumes with a single statement, the appropriate number of volume serial numbers or an initial value must be specified.

Default: If the VSN is not specified, a volume of type “voltyp” with any volume serial number is requested.
In this way, it is possible to read the labels of an unknown volume.

VSN=vsn
Single volume serial number. The labels of a single volume are to be output. If a single volume serial number is specified, the value of the “no” operand must be 1 or “no” must not be specified.

Permissible values: Max. 6 characters (A..Z, 0..9, #,$,@).
The special characters #, $, @ are permitted by INIT but must not be used in labels (DIN 66029, DIN 66003). If OPTION DIN is active, these special characters are rejected.

VSN=(vsn1,vsn2,...,vsn16)
The labels of the volumes with the specified volume serial numbers are to be output. The corresponding volumes are requested in consecutive order. The number of volumes must tally with the number specified for the “no” operand. See above and "INIT - Initialize magnetic tape" for a list of possible values.

VSN=(initval)
The labels of multiple volumes beginning with the volume serial number “initval” are to be output. Every time the labels of a volume are read successfully, the volume serial number is automatically incremented by 1. The number of volumes must be specified in the “no” operand. See above and "INIT - Initialize magnetic tape" for a list of possible values.

UNIT=mn
Mnemonic device name for seizure of a particular device. A volume is to be mounted on the specified device. Unsuitable devices, i.e. devices that cannot process volumes of the specified type, are rejected.

REW
Specifies that the magnetic tape or magnetic tape cartridge is to be rewound after listing but not unloaded.
If a single volume is listed, REW is set by default.

RUN
Specifies that the magnetic tape or magnetic tape cartridge is to be rewound and unloaded after listing.
If two or more volumes are to be listed with a single statement (operand no=2..16), RUN is set by default.

Examples

1. Outputting the labels of a single volume with any volume serial number (specifying minimum parameters):

   ```
   LIST TAPE-C4
   ```

2. Outputting the labels of three particular volumes with a single statement:

   ```
   LIST 3 TAPE-C4,VSN=(TAPE01,TAPE02,TAPE03)
   ```

3. Outputting the records of 8 particular volumes with a single statement:

   ```
   LIST 8 TAPE-C4,VSN=(VSN001)
   ```

   The volumes with the volume serial numbers VSN001, VSN002,...,VSN008 are requested one after the other.

4. Outputting the labels
   - of a particular volume (with the volume serial number VSN001),
   - on a particular device (with the mnemonic name M1),
   - and remove after initialization:

   ```
   LIST TAPE-C4,VSN=VSN001,UNIT=M1,RUN
   ```

3.3.3 OPTION - Activate and deactivate optional functions

The OPTION statement is used to activate and deactivate optional functions; the setting remains valid for all subsequent INIT or LIST operations. The options are enforced until the program is ended or until reset.

Format

OPTION
CONS / PROT / DIN / NOHDR / NONE

Operands

CONS
Diverts inputs/outputs to the console once the program has been started from the console by means of the ENTER job.

> CONS is valid only within an ENTER job and is rejected in interactive mode.

PROT
Aborts initialization of a volume if the read-before-write check discovers an access restriction.

Each of the following criteria is a valid access restriction:

- The access flag in the VOL1 volume label permits access to the volume only by the owner (byte 11 in the VOL1 label contains neither “0” nor “ ” (space)).
- The access flag in the file label of the first file permits access to the file only by the owner (byte 54 in the HDR1 label contains neither “0” nor “ ” (space)).
- The expiration date of the first file is not yet reached (bytes 48-53 in the HDR1 label).
- The first file is protected by a read and/or write password (bytes 57-64 in the HDR3 label).
- The first file is write-protected (byte 69 in the HDR3 label contains a “1”).

In all the cases listed above, the labels are read and output, followed by a message defining the situation and a prompt calling for another statement.
If the user wants to initialize the volume despite the access restriction, the OPTION NONE statement must be used to revoke the restriction.

DIN
Excludes special characters that do not conform to DIN 66003 from being used in labels. If the user wants to employ special characters (#, $, @) in labels nevertheless, OPTION NONE must be used to revoke the restriction.

NOHDR
Initializes a volume with the VOL1 volume label only. Once this option is activated, all subsequent initializations are implemented without HDR labels until the option is reset. In order to resume initialization with the HDR1 and HDR2 labels, OPTION NONE must be used to revoke the NOHDR setting.

NONE
Resets all options.

> Option CONS cannot be reset.

Examples

1. OPTION PROT

```
(OUT)  %  NVI0001 ENTER COMMAND (END = TERMINATE INIT)
(IN)   OPTION PROT                                                    a)
(OUT)  %  NVI0001 ENTER COMMAND (END = TERMINATE INIT)
(IN)   INIT TAPE-C4,VSN=TEST01                                        b)
(OUT)  %  NVI0003 LABELS ON TAPE       :                              c)
(OUT)  %  NVI0004 VOL1 LABEL:
                  'VOL1DAR15K                                 '
                  '                                          1'
(OUT)  %  NVI0004 HDR1 LABEL:
                 'HDR1V11.TEST-R.TXT.TEDAR15K0001000100010'
                 '0 92104 92107 000000BS2000                '
(OUT)  %  NVI0101 EXPIRATION DATE OF FIRST FILE NOT YET REACHED       d)
(OUT)  %  NVI0004 HDR2 LABEL:
                 'HDR2V8000102044 0                P     '
                 '      .C..04                             '
(OUT)  %  NVI0004 HDR3 LABEL:
                 'HDR3TSOS    V11.TEST-R.TXT.TESTFILE.T-C'
                 '4               ............00          '
(OUT)  %  NVI0201 TAPE PROTECTED.                                     e)
                   INITIALIZATION REQUEST REJECTED
(OUT)  %  NVI0001 ENTER COMMAND (END = TERMINATE INIT)
```

   a. OPTION PROT is set.
   b. A volume of type TAPE-C4 is to be initialized.
   c. The labels of the volume are output. The expiration date of the first file on the volume has not yet been reached (bytes 48-53 of the HDR1 label).
   d. A message to this effect is output.
   e. No new labels are written. Initialization is aborted.

2. OPTION DIN

```
(OUT)  %  NVI0001 ENTER COMMAND (END = TERMINATE INIT)
(IN)   OPTION DIN                                                     a)
(OUT)  %  NVI0001 ENTER COMMAND (END = TERMINATE INIT)
(IN)   LIST TAPE-C4,VSN=T#$@                                          b)
(OUT)  %  NVI0405 INVALID OPERAND VALUE ',VSN=T#'
(OUT)  %  NVI0001 ENTER COMMAND (END = TERMINATE INIT)
(IN)   INIT TAPE-C4,VSN=DAR10A,OWN=TEST#$@                            c)
(OUT)  %  NVI0405 INVALID VALUE ',OWN=TEST#'
(OUT)  %  NVI0001 ENTER COMMAND (END = TERMINATE INIT)
```

   a. OPTION DIN is set.
   b. The labels of a volume of type TAPE-C4, whose volume serial number contains special characters, are to be output. This statement is rejected.
   c. A volume of type TAPE-C4 is to be initialized and the owner’s name entered in the VOL1 label. The name in question includes special characters. This statement is rejected.

1. OPTION NOHDR

```
(OUT)  %  NVI0001 ENTER COMMAND (END = TERMINATE INIT)
(IN)   OPTION NOHDR                                                    a).
(OUT)  %  NVI0001 ENTER COMMAND (END = TERMINATE INIT)
(IN)  INIT TAPE-C4,VSN=TAPE01,CHECK=TAPE01                             b)
(OUT)  %  NVI0003 LABELS ON TAPE                                       c)
(OUT)  %  NVI0004 VOL1 LABEL: 'VOL1TAPE01...
(OUT)  %  NVI0004 HDR1 LABEL: 'HDR1      ...
(OUT)  %  NVI0004 HDR2 LABEL: 'HDR2U00001...
(OUT)  %  NVI0007 OVERWRITE TAPE ?                                     d)
                  REPLY (YES=YES  N=NO)
(IN)   YES
(OUT)  %  NVI0010 INITIALIZATION OF TAPE 'TAPE01'
          ON DEVICE 'TA' COMPLETED
(OUT)  %  NVI0001 ENTER COMMAND (END = TERMINATE INIT)
(IN)   LIST TAPE-C4,VSN=TAPE01                                         e)
(OUT)  %  NVI0003 LABELS ON TAPE       :                               f)
(OUT)  %  NVI0004 VOL1 LABEL: 'VOL1TAPE01...
(OUT)  %  NVI0001 ENTER COMMAND (END = TERMINATE INIT)
```

a. OPTION NOHDR is set.
b. A volume is to be initialized.
c. The VOL1, HDR1 and HDR2 labels on the volume are output.
d. The volume is re-initialized.
e. The labels are to be output.
f. The volume is initialized with a VOL1 label and no others.

3.3.4 END - Terminate INIT session

The END statement terminates the INIT utility routine. The INIT subsystem is released for unloading.
In command procedures, the INIT routine is ended even without an END statement as soon as a BS2000 command is read (beginning with /) or the end of the file is reached.

Format

Once the program is started, messages are output via SYSOUT. The system awaits inputs from SYSDTA.

END / E

3.3.5 HELP - Outline description of INIT statements

The HELP statement calls up outline information on all operations and operands in the INIT utility routine.
If OPTION CONS is set, the list is output to the console. If this option is not set, the list is output to SYSOUT.

Format

HELP / H

3.4 Structure of the labels

The general format of the labels complies with the DIN 66029 standard (magnetic tapes). These labels are also used for other volumes that are emulated as magnetic tapes or magnetic tape cartridges in BS2000.

Magnetic tape labels are 80 characters long.

Only the structure and contents of those magnetic tape labels read or written by INIT are described below.

The column headed “Field contents” contains the information that INIT writes into the field in question (or the possible contents in the case of the HDR3 label).

Key to symbols

x: Current value

_: Blank

The other fields contain fixed defaults.

3.4.1 Volume label VOL1 for magnetic tapes

Byte no.	Field name	Length	Field contents
1 - 3	Label name	3	VOL
4	Label number	1	1
5 - 10	Volume serial number <vsn>	6	xxxxxx
11	Access flag <flg>	1	_ or 0
12 - 37	-reserved -	26	_..._
38 - 51	Owner label <owner>	14	_ _ _ _xxxxxxxx_ _
52 - 79	-reserved -	28	_..._
80	Standard flag	1	1

V	O	L	1	vsn						flg	_	_	_	_	_
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
_	_	_	_	_	_	_	_	_	owner						
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	1
65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80

3.4.2 File label HDR1 for magnetic tapes

Byte no.	Field name	Length	Field contents
1 - 3	Label name	3	HDR
4	Label number	1	1
5 - 21	File name	17	_..._
22 - 27	Volume serial number <vsn>	6	xxxxxx
28 - 31	File section number	4	0001
32 - 35	File sequence number	4	0001
36 - 39	Generation number	4	0001
40 - 41	Version number	2	00
42 - 47	Creation date <credat>	6	xxxxxx
48 - 53	Expiration date <reldat>	6	xxxxxx
54	Access flag	1	_
55 - 60	Block counter	6	000000
61 - 73	System code	13	_..._
74 - 80	- reserved -	7	_..._

Format of the creation date and expiration date: cyyjjj

c Code for century:
blank = “20th century”; 0 = “21st century”

yy = Year

jjj = Julian date (number of the day in the year).

Examples

```
July 3, 1998:_98185

January 10, 2004:004010
```

H	D	R	1	_	_	_	_	_	_	_	_	_	_	_	_
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
_	_	_	_	_	vsn						0	0	0	1	0
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
0	0	1	0	0	0	1	0	0	credat						
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
reldat					_	0	0	0	0	0	0	_	_	_	_
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80

3.4.3 File label HDR2 for magnetic tapes

Byte no.	Field name	Length	Field contents
1 - 3	Label name	3	HDR
4	Label number	1	2
5	Record format	1	U
6 - 10	Block length	5	00001
11 - 15	Record length	5	00001
16 - 50	- reserved -	35	_..._
51 - 52	Buffer displacement	2	00
52 - 80	- reserved -	28	_..._

H	D	R	2	U	0	0	0	0	1	0	0	0	0	1	_
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
_	_	0	0	_	_	_	_	_	_	_	_	_	_	_	_
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80

3.4.4 File label HDR3 for magnetic tapes

The HDR3 label is never written as part of initialization. If it exists, however, it is output when the labels are read.

Byte no.	Field name	Length	Field contents
1 - 3	Label name	3	HDR
4	Label number	1	3
5 - 12	Owner ID <uid>	8	xxxxxxxx
13 - 56	File name <fnam>	44	xxx...xxx
57 - 60	Read password <rpass>	4	xxxx or X' 00000000'
61 - 64	Write password <wpass>	4	xxxx or X' 00000000'
65 - 68	Execution password <epass>	4	xxxx or X' 00000000'
69	Access type <acc>	1	0 = read + write 1 = read only
70 - 80	- reserved -	11	_..._

H	D	R	3	uid								fnam			
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
fnam															
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
fnam															
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
fnam								rpass				wpass			
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
epass				acc	_	_	_	_	_	_	_	_	_	_	_
65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80

4 IORM Dynamic control of I/O resources

Version: **IORM V21.0A**

Privileges: **TSOS** or **OPERATING**

The IORM utility routine is started using `/START-IORM` (TSOS or OPERATING privilege).

The IORM (**I**/**O** **R**esource **M**anagement) utility routine enhances the I/O properties of BS2000 in native and VM2000 modes.

The following functions are implemented in IORM for the autonomous, dynamic control of the I/O resources channel, controller, path and device:

- IOPT: I/O Priority Handling for Tasks
- DPAV (/390 server): Dynamic Parallel Access Volume
- DDAL: Optimized Dynamic Device Allocation in ETERNUS CS operation
- TCOM: Dynamic Tape Compression
- IOLVM (/390 server): I/O Limit for certain Virtual Machines

During ongoing operation, IORM collects data on the utilization of the I/O resources and controls I/O operation in accordance with specified threshold values.

The IORM functions IOPT, DPAV and IOLVM control disk devices.
The IORM functions DDAL and TCOM control tape devices, see "IORM Dynamic control of I/O resources".

When IORM is used in the monitor system and in the BS2000 guest systems involved in VM2000 mode, the IORM subsystems exchange I/O data and control information via an internal interface.

IORM can be used in native mode and under VM2000 on all BS2000 servers. IORM works on a cross-VM but not on a cross-server basis.

Installation of IORM

IORM is installed using the IMON installation monitor.

`SYSLNK.IORM.<version>`	Module library for /390 servers
`SKMLNK.IORM.<version>`	Module library for x86 servers
`SYSMES.IORM.<version>`	Message file
`SYSSDF.IORM.<version>`	SDF syntax file
`SYSSSC.IORM.<version>`	Subsystem catalog
`SYSDAT.IORM.<version>`	Parameter file
`SYSSII.IORM.<version>`	Structure information
`SYSNRF.IORM.<version>`	NOREF file
`SYSRMS.IORM.<version>`	RMS delivery packet

SYSRME.IORM.<version>.D/E	Readme file (German/English)
SYSFGM.IORM.<version>.D/E	Release Notice (German/English)

Table 2: Delivery components of IORM

Parameter file of IORM

In the parameter file $<user ID>.SYSDAT.IORM.<version> you can enter specifications for IORM which are processed the first time IORM is started. The parameter file consists of comment lines and value lines with statements for IORM. When supplied, the parameter file contains comment lines with sample statements for IORM.

IORM subsystem

Before the IORM utility routine is started for the first time, the IORM subsystem must be loaded using:

```
/START-SUBSYSTEM SUBSYSTEM-NAME=IORM
```

The subsystem IORM is terminated using:

```
/STOP-SUBSYSTEM SUBSYSTEM-NAME=IORM
```

If, at this time, the IORM utility routine is still in use, the subsystem remains in the IN DELETE / WAIT-DISCON state until usage comes to an end.

Starting the IORM utility routine

The IORM utility routine is started using /START-IORM (TSOS or OPERATING privilege).

START-IORM	Alias: **IORM**
VERSION = ***STD** / <product-version>	
,MONJV = ***NONE** / <filename 1..54 without-gen-vers>	
,CPU-LIMIT = ***JOB-RE**ST / <integer 1..32767 *seconds*>	

The first time the IORM utility routine is started, the actions required for IORM operation are executed under the user ID where /START-IORM has been entered: configuring tables, connecting to the BS2000 I/O system, processing the statements from the IORM parameter file. IORM then terminates.
In VM2000 mode, IORM should be started in all BS2000 guest systems.

When IORM is started again, you can enter statements for IORM via SYSDTA. These statements control the execution of IORM. The statements can be entered directly in a dialog or in file form.

Example

```
/START-SUBSYSTEM SUBSYSTEM-NAME=IORM                                    1.
ESM0216 FUNCTION 'CREATE' STARTED FOR SUBSYSTEM 'IORM    /<version>'
/...
/start-iorm                                                             2.
BLS0523 ELEMENT 'IORM-TU', VERSION '<version>', TYPE 'L' FROM LIBRARY
        ':SBZ7:$TSOS.SYSLNK.IORM.<version>' IN PROCESS
BLS0524 LLM 'IORM-TU', VERSION '<version>' OF '<date> <time>' LOADED
/start-iorm                                                             3.
BLS0523 ELEMENT 'IORM-TU', VERSION '<version>', TYPE 'L' FROM LIBRARY
        ':SBZ7:$TSOS.SYSLNK.IORM.<version>' IN PROCESS
BLS0524 LLM 'IORM-TU', VERSION '<version>' OF '<date> <time>' LOADED
* iopt_set_on?
IOPT_SET_ON=NO                                                          4.
* ...
//END
/...
/ASSIGN-SYSDTA TO=SYSDAT.IORM.NIGHT                                    5.
/START-IORM
BLS0523 ELEMENT 'IORM-TU', VERSION '<version>', TYPE 'L' FROM LIBRARY
        ':SBZ7:$TSOS.SYSLNK.IORM.<version>' IN PROCESS
BLS0524 LLM 'IORM-TU', VERSION '<version>' OF '<date> <time>' LOADED
/...
/ASSIGN-SYSDTA TO=SYSDAT.IORM.DAY                                      6.
/START-IORM
BLS0523 ELEMENT 'IORM-TU', VERSION '<version>', TYPE 'L' FROM LIBRARY
        ':SBZ7:$TSOS.SYSLNK.IORM.<version>' IN PROCESS
BLS0524 LLM 'IORM-TU', VERSION '<version>' OF '<date> <time>' LOADED
/...
```

1. The IORM subsystem is started.
2. The IORM utility routine is started for the first time for initialization purposes.
3. The IORM utility routine is started again to enter statements.
4. The specifications in the parameter file apply.
5. During the night, the IORM subsystem that is already active obtains new specifications from the `SYSDAT.IORM.NIGHT` file.
6. The next morning, the IORM subsystem that is still active obtains new specifications from the `SYSDAT.IORM.DAY` file.

IORM statements

IORM statements consist of character and numeric strings without blanks.Comment lines begin with `*`.

IORM knows the following general statements:

END Terminates the IORM utility routine. The IORM subsystem and its functions remain active.

HELP Displays the IORM statements.

IORM_DUMP Creates IORM diagnostic documentation.

The other statements are function-specific and are described together with the function concerned.

Hardware dependencies of IORM

IORM supports the following hardware components:

IORM function	Hardware supported
IOPT	Disks on all BS2000 servers
DPAV	Disks connected to the Fibre Channel of /390 servers DPAV is not available on the Fibre Channel of x86 servers. IO parallelism is supported here for emulated disks (RSC).
DDAL	Tape devices on all BS2000 servers
TCOM	LTO tape devices connected to the Fibre Channel of all BS2000 servers
IOLVM	Disks on all BS2000 servers

Behavior in the event of an error, creation of diagnostic documentation

To permit IORM problems to be diagnosed and rectified, it is necessary that sufficient error documents should be created and stored at the earliest possible time. In the case of reproducible errors, a precise description should be provided of how an error can be generated.

In addition to the CONSLOG file and the IORM statements, a file with the IORM dump should always be created using the following statement sequence:

```
/ASSIGN-SYSLST TO=<file>
/START-IORM
* IORM_DUMP
* END
/ASSIGN-SYSLST TO=*PRIMARY
```

In the event of problems with DPAV or DDAL in VM2000 mode, these documents are also required from the monitor system and the guest systems involved.

4.1 IOPT: I/O Priority Handling for Tasks

An I/O-intensive application with low priority can hinder another, higher-priority application if these application execute I/Os on the same (logical) device. Operations can also be impaired if the I/Os are executed on different (logical) devices which are located on the same physical device or are connected via the same paths, are accessible via the same ports or connected to the same channels.

Enhanced priority handling for tasks using IOPT

The IOPT (**I**/**O** **P**riority Handling for **T**asks) function enables IORM to detect such conflict situations and to intervene in I/O operation for control purposes.

Here IOPT examines both the utilization level of the I/O units (devices, paths, ports and channels) and the I/O priorities of the tasks which use them.

I/O priorities for tasks

Three I/O priorities are available for tasks:

- HIGH (high I/O priority)
- MEDIUM (medium I/O priority)
- LOW (low I/O priority)

The I/O priority for tasks can be defined in two ways:

- With /`MODIFY-TASK-CATEGORIES`, `IO-PRIORITY=*HIGH/*MEDIUM/*LOW` operand. This is the procedure recommended.

  The command defines the I/O priority for the task categories.
  In addition to the four standard categories SYS, TP, DIALOG and BATCH, there may be also further categories whose names are defined in the job class definition.
- With the IORM statements IOPT_PRI_HIGH and IOPT_PRI_MED.
  However, these settings only have an effect if `IO-PRIORITY=*NONE` is set in `/MODIFY-TASK-CATEGORIES` (default value).

  The IORM statements define the I/O priority on the basis of the task priorities. The task priorites are defined in the job class definition and on a user-specific basis in the user catalog.

Specifications for the I/O units

The IORM statement IOPT_DEV_ADD enables disk devices to be defined on which IORM priority handling is to be used.

The IORM statements IOPT_LOW_xxx and IOPT_MED_xxx enable threshold values and I/O shares to be defined for tasks with the priority LOW or MEDIUM for the I/O units device (xxx=DEV), path (xxx=PTH), port (xxx=POR) and channel (xxx=CHN).

The default values set for this in IORM match the specifications in the `SYSDAT.IORM.<version>` parameter file which is supplied.

> The fact that the I/O load increases significantly more rapidly on a device than on a path, port or channel, in particular in the case of a multipath device connection, should be taken into account in the specifications for the I/O units.
> It can occur that IORM does not detect a conflict situation on a physical disk quickly enough because the utilization of the paths, ports, channels and logical devices is still below the specified threshold values. In this case it can make sense to combine all logical devices on the physical device to a device group in IORM (IOPT_GRP_ADD).

Execution of IOPT

At first IOPT is deactivated. After it has been activated (IOPT_SET_ON=YES), IOPT constantly collects utilization values, separated according to the I/O priorities HIGH / MEDIUM / LOW, for all known I/O units.

IOPT periodically checks whether tasks with the I/O priority LOW or MEDIUM are hindering higher-priority tasks (I/O priority MEDIUM or HIGH) in an I/O unit. If this is the case, only a limited utilization level is permitted on the I/O unit for tasks with the I/O priority concerned. This ensures that - in accordance with the utilization of the I/O unit - tasks with a lower I/O priority are “held back” from executing I/Os on devices which were activated beforehand for IOPT using IOPT_DEV_ADD.

IOPT only has a local effect, both in native mode and on a BS2000 guest system. Only disk devices are concerned here.
Tasks for FDDRL, ARCHIVE, VOLIN and PAGING are not held back.

4.1.1 IOPT statements

Default values in the syntax boxes are underscored.

4.1.1.1 Activating and deactivating IOPT

IOPT_SET_ON activates or deactivates IOPT. The current setting can also be queried.

IOPT_SET_ON
IOPT_SET_ON=YES IOPT_SET_ON=NO IOPT_SET_ON=CHK IOPT_SET_ON?

YES IOPT is activated on the system. In VM mode this setting is required on all systems on which IOPT is to be active.

NO IOPT is deactivated on the system. In VM mode this setting is required on all systems on which IOPT is to be deactivated.

CHK IOPT runs in check mode, i.e. IOPT does not intervene in I/O handling, but it is possible to ascertain which interventions IOPT would perform if it were activated.
This enables you to check whether the use of IOPT makes sense, see the section“Check mode”.

? The current setting is queried.

4.1.1.2 Defining and querying threshold values for I/O priority classes

IOPT_PRI_HIGH and IOPT_PRI_MED define the threshold values for the tasks with high, medium and low priority. Here the I/O priority of a task is derived from its task priority. The threshold values that are currently valid can be queried.

IOPT_PRI
IOPT_PRI_HIGH=x IOPT_PRI_MED=y IOPT_PRI_HIGH? IOPT_PRI_MED?

x Threshold value for the I/O priority HIGH. All tasks with a task priority which is less than or equal to x belong to the class HIGH (prio <= x).
Value range: 0 <= x <= 255
Default value: 155

y Threshold value for the I/O priorities MEDIUM and LOW. All tasks with a task priority which is greater than x and less than or equal to y belong to the class MEDIUM (x < prio <= y). All tasks with a priority which is greater than y belong to the class LOW (y < prio).
Value range: x < y <= 255
Default value: 205

? The threshold values for the classes HIGH and MEDIUM are queried.

The threshold values are effective only if the `IO-PRIORITY` operand in `/MODIFY-TASK-CATEGORIES` has the value `*NONE`. However, you are recommended to define the I/O priorities using `/MODIFY-TASK-CATEGORIES` because the I/O priority can then be assigned independently of the task priority. You may also want two tasks with the same task priority to have different I/O priorities because they differ in their I/O intensiveness.

4.1.1.3 Activating and deactivating disk devices for IOPT

Activating disk devices for IOPT

IOPT_DEV_ADD activates logical devices for IOPT. The devices are then monitored by IOPT.

Once the program is started, messages are output via SYSOUT. The system awaits inputs from SYSDTA.

IOPT_DEV_ADD
IOPT_DEV_ADD=ALL IOPT_DEV_ADD=D-R(mn1,mn2)

ALL All devices are activated for IOPT.

D-R(mn1,mn2)

All devices with mnemonic names from mn1 to mn2 are activated. If mn1 and mn2 are identical, only one device is activated.

Deactivating disk devices for IOPT

IOPT_DEV_REM deactivates logical devices for IOPT. The devices are then not (no longer) monitored by IOPT.

IOPT_DEV_REM
IOPT_DEV_REM=ALL IOPT_DEV_REM=D-R(mn1,mn2)

ALL All devices are deactivated for IOPT.

D-R(mn1,mn2)

All devices with mnemonic names from mn1 to mn2 are deactivated. If mn1 and mn2 are identical, only one device is deactivated.

Querying activated devices

IOPT_DEV_ADD? outputs a list of all devices which were activated for IOPT using IOPT_DEV_ADD.

IOPT_DEV_ADD?
IOPT_DEV_ADD?mask

mask Complete or masked specification of a mnemonic device name. The following specifications are possible:

*	All activated devices are listed.
n*	All activated devices whose names begin with n are listed.
nm*	All activated devices whose names begin with nm are listed.
nml*	All activated devices whose names begin with nml are listed.
nmlo	The device with the name nmlo is listed.

4.1.1.4 Device groups

Adding devices to a device group

IOPT_GRP_ADD adds logical devices to a device group. If a group with the specified name does not exist, it is configured. Only devices which are activated (either beforehand or subsequently) for IOPT using IOPT_DEV_ADD are also monitored by IOPT. Merely defining a device group is not enough to activate it.

IOPT monitors a device group like a device, in other words it examines the utilization of the entire group. For example, all logical devices which are assigned to a physical disk can form a device group. In this way IOPT detects I/O bottlenecks on the disk even if the utilization threshold values have not yet been exceeded for individual logical devices.

IOPT_GRP_ADD
IOPT_GRP_xxx_ADD=D-R(mn1.mn2)

xxx Three-digit numeric name of the device group.
Value range: 000 < x <= 255

D-R(mn1,mn2)

All devices with mnemonic names from mn1 to mn2 are added to the group. If mn1 and mn2 are identical, only one device is added to the group.

Removing devices from a device group

IOPT_GRP_REM removes logical devices from a device group. When the last device is removed from a group, the group is also deleted.

IOPT_GRP_REM
IOPT_GRP_xxx_REM=ALL IOPT_GRP_xxx_REM=D-R(mn1,mn2)

xxx Three-digit numeric name of the device group.
Value range: 000 < x <= 255

ALL All devices in the group are removed. The group is deleted.

D-R(mn1,mn2)

All devices with mnemonic names from mn1 to mn2 are removed from the group. If mn1 and mn2 are identical, only one device is removed from the group.

Querying devices of a device group

IOPT_GRP_ADD? displays a list of all the devices which belong to a device group.

IOPT_GRP_ADD?
IOPT_GRP_xxx_ADD?mask

xxx Three-digit numeric name of the device group.
Value range: 000 < x <= 255

mask Complete or masked specification of a mnemonic device name. The following specifications are possible:

*	All devices in the group are listed.
n*	All devices in the group whose names begin with n are listed.
nm*	All devices in the group whose names begin with nm are listed.
nml*	All devices in the group whose names begin with nml are listed.
nmlo	The device in the group with the name nmlo is listed.

4.1.1.5 Defining threshold values for I/O priority LOW

Defining threshold values for I/O priority LOW and channels

IOPT_LOW_CHN defines threshold values for channel utilization. As soon as the channel utilization by tasks with the priorities HIGH and MEDIUM reaches the threshold values, the proportion of tasks with the priority LOW is restricted.

IOPT_LOW_CHN
IOPT_LOW_CHN_1HM=20 / <integer 0...100>
IOPT_LOW_CHN_2HM=30 / <integer 0...100>
IOPT_LOW_CHN_3HM=40 / <integer 0...100>
IOPT_LOW_CHN_1QH=40 / <integer 0...100>
IOPT_LOW_CHN_2QH=30 / <integer 0...100>
IOPT_LOW_CHN_3QH=20 / <integer 0...100>
IOPT_LOW_CHN_1QM=55 / <integer 0...100>
IOPT_LOW_CHN_2QM=40 / <integer 0...100>
IOPT_LOW_CHN_3QM=30 / <integer 0...100>

IOPT_LOW_CHN_1HM, IOPT_LOW_CHN_2HM, IOPT_LOW_CHN_3HM

Threshold values 1, 2 and 3 for utilization by tasks with the priority HIGH or MEDIUM.

IOPT_LOW_CHN_1QH, IOPT_LOW_CHN_2QH, IOPT_LOW_CHN_3QH

Threshold values 1, 2 and 3 for tasks with the priority LOW in the event of a high proportion of tasks with the priority HIGH.

IOPT_LOW_CHN_1QM, IOPT_LOW_CHN_2QM, IOPT_LOW_CHN_3QM

Threshold values1, 2 and 3 for tasks with the priority LOW in the event of a low proportion of tasks with the priority HIGH.

The threshold values have the following effects:

- If utilization by tasks with the priority HIGH or MEDIUM is less than IOPT_LOW_CHN_1HM, the tasks with the priority LOW are not restricted.
- If utilization by tasks with the priority HIGH or MEDIUM is between IOPT_LOW_CHN_1HM and IOPT_LOW_CHN_2HM, then IOPT_LOW_CHN_1QH or IOPT_LOW_CHN_1QM restricts the proportion of tasks with the priority LOW:
  - IOPT_LOW_CHN_1QH if the proportion of tasks with the priority HIGH is high
  - IOPT_LOW_CHN_1QM if the proportion of tasks with the priority HIGH is low
- If utilization by tasks with the priority HIGH or MEDIUM is between IOPT_LOW_CHN_2HM and IOPT_LOW_CHN_3HM, then IOPT_LOW_CHN_2QH or IOPT_LOW_CHN_2QM restricts the proportion of tasks with the priority LOW:
  - IOPT_LOW_CHN_2QH if the proportion of tasks with the priority HIGH is high
  - IOPT_LOW_CHN_2QM if the proportion of tasks with the priority HIGH is low

- If utilization by tasks with the priority HIGH or MEDIUM is greater than IOPT_LOW_CHN_3HM, then IOPT_LOW_CHN_3QH or IOPT_LOW_CHN_3QM restricts the proportion of tasks with the priority LOW:
  - IOPT_LOW_CHN_3QH if the proportion of tasks with the priority HIGH is high
  - IOPT_LOW_CHN_3QM if the proportion of tasks with the priority HIGH is low

Defining threshold values for I/O priority LOW and ports

IOPT_LOW_POR defines threshold values for port utilization. As soon as the port utilization by tasks with the priorities HIGH and MEDIUM reaches the threshold values, the proportion of tasks with the priority LOW is restricted.

IOPT_LOW_POR
IOPT_LOW_POR_1HM=20 / <integer 0...100>
IOPT_LOW_POR_2HM=30 / <integer 0...100>
IOPT_LOW_POR_3HM=40 / <integer 0...100>
IOPT_LOW_POR_1QH=40 / <integer 0...100>
IOPT_LOW_POR_2QH=30 / <integer 0...100>
IOPT_LOW_POR_3QH=20 / <integer 0...100>
IOPT_LOW_POR_1QM=55 / <integer 0...100>
IOPT_LOW_POR_2QM=40 / <integer 0...100>
IOPT_LOW_POR_3QM=30 / <integer 0...100>

IOPT_LOW_POR_1HM, IOPT_LOW_POR_2HM, IOPT_LOW_POR_3HM

Threshold values 1, 2 and 3 for utilization by tasks with the priority HIGH or MEDIUM.

IOPT_LOW_POR_1QH, IOPT_LOW_POR_2QH, IOPT_LOW_POR_3QH

Threshold values 1, 2 and 3 for tasks with the priority LOW in the event of a high proportion of tasks with the priority HIGH.

IOPT_LOW_POR_1QM, IOPT_LOW_POR_2QM, IOPT_LOW_POR_3QM

Threshold values 1, 2 and 3 for tasks with the priority LOW in the event of a low proportion of tasks with the priority HIGH.

The threshold values have the following effects:

- If utilization by tasks with the priority HIGH or MEDIUM is less than IOPT_LOW_POR_1HM, the tasks with the priority LOW are not restricted.
- If utilization by tasks with the priority HIGH or MEDIUM is between IOPT_LOW_POR_1HM and IOPT_LOW_POR_2HM, then IOPT_LOW_POR_1QH or IOPT_LOW_POR_1QM restricts the proportion of tasks with the priority LOW:
  - IOPT_LOW_POR_1QH if the proportion of tasks with the priority HIGH is high
  - IOPT_LOW_POR_1QM if the proportion of tasks with the priority HIGH is low

- If utilization by tasks with the priority HIGH or MEDIUM is between IOPT_LOW_POR_2HM and IOPT_LOW_POR_3HM, then IOPT_LOW_POR_2QH or IOPT_LOW_POR_2QM restricts the proportion of tasks with the priority LOW:
  - IOPT_LOW_POR_2QH if the proportion of tasks with the priority HIGH is high
  - IOPT_LOW_POR_2QM if the proportion of tasks with the priority HIGH is low
- If utilization by tasks with the priority HIGH or MEDIUM is greater than IOPT_LOW_POR_3HM, then IOPT_LOW_POR_3QH or IOPT_LOW_POR_3QM restricts the proportion of tasks with the priority LOW:
  - IOPT_LOW_POR_3QH if the proportion of tasks with the priority HIGH is high
  - IOPT_LOW_POR_3QM if the proportion of tasks with the priority HIGH is low

Defining threshold values for I/O priority LOW and paths

IOPT_LOW_PTH defines threshold values for path utilization. As soon as the path utilization by tasks with the priorities HIGH and MEDIUM reaches the threshold values, the proportion of tasks with the priority LOW is restricted.

IOPT_LOW_PTH
IOPT_LOW_PTH_1HM=20 / <integer 0...100>
IOPT_LOW_PTH_2HM=30 / <integer 0...100>
IOPT_LOW_PTH_3HM=40 / <integer 0...100>
IOPT_LOW_PTH_1QH=40 / <integer 0...100>
IOPT_LOW_PTH_2QH=30 / <integer 0...100>
IOPT_LOW_PTH_3QH=20 / <integer 0...100>
IOPT_LOW_PTH_1QM=55 / <integer 0...100>
IOPT_LOW_PTH_2QM=40 / <integer 0...100>
IOPT_LOW_PTH_3QM=30 / <integer 0...100>

IOPT_LOW_PTH_1HM, IOPT_LOW_PTH_2HM, IOPT_LOW_PTH_3HM

Threshold values 1, 2 and 3 for utilization by tasks with the priority HIGH or MEDIUM.

IOPT_LOW_PTH_1QH, IOPT_LOW_PTH_2QH, IOPT_LOW_PTH_3QH

Threshold values 1, 2 and 3 for tasks with the priority LOW in the event of a high proportion of tasks with the priority HIGH.

IOPT_LOW_PTH_1QM, IOPT_LOW_PTH_2QM, IOPT_LOW_PTH_3QM

Threshold values 1, 2 and 3 for tasks with the priority LOW in the event of a low proportion of tasks with the priority HIGH.

The threshold values have the following effects:

- If utilization by tasks with the priority HIGH or MEDIUM is less than IOPT_LOW_PTH_1HM, the tasks with the priority LOW are not restricted.

- If utilization by tasks with the priority HIGH or MEDIUM is between IOPT_LOW_PTH_1HM and IOPT_LOW_PTH_2HM, then IOPT_LOW_PTH_1QH or IOPT_LOW_PTH_1QM restricts the proportion of tasks with the priority LOW:
  - IOPT_LOW_PTH_1QH if the proportion of tasks with the priority HIGH is high
  - IOPT_LOW_PTH_1QM if the proportion of tasks with the priority HIGH is low
- If utilization by tasks with the priority HIGH or MEDIUM is between IOPT_LOW_PTH_2HM and IOPT_LOW_PTH_3HM, then IOPT_LOW_PTH_2QH or IOPT_LOW_PTH_2QM restricts the proportion of tasks with the priority LOW:
  - IOPT_LOW_PTH_2QH if the proportion of tasks with the priority HIGH is high
  - IOPT_LOW_PTH_2QM if the proportion of tasks with the priority HIGH is low
- If utilization by tasks with the priority HIGH or MEDIUM is greater than IOPT_LOW_PTH_3HM, then IOPT_LOW_PTH_3QH or IOPT_LOW_PTH_3QM restricts the proportion of tasks with the priority LOW:
  - IOPT_LOW_PTH_3QH if the proportion of tasks with the priority HIGH is high
  - IOPT_LOW_PTH_3QM if the proportion of tasks with the priority HIGH is low

Defining threshold values for I/O priority LOW and devices

IOPT_LOW_DEV defines threshold values for device utilization. As soon as the device utilization by tasks with the priorities HIGH and MEDIUM reaches the threshold values, the proportion of tasks with the priority LOW is restricted.

IOPT_LOW_DEV
IOPT_LOW_DEV_1HM=15 / <integer 0...100>
IOPT_LOW_DEV_2HM=22 / <integer 0...100>
IOPT_LOW_DEV_3HM=30 / <integer 0...100>
IOPT_LOW_DEV_1QH=35 / <integer 0...100>
IOPT_LOW_DEV_2QH=25 / <integer 0...100>
IOPT_LOW_DEV_3QH=10 / <integer 0...100>
IOPT_LOW_DEV_1QM=50 / <integer 0...100>
IOPT_LOW_DEV_2QM=40 / <integer 0...100>
IOPT_LOW_DEV_3QM=30 / <integer 0...100>
IOPT_LOW_DEV_0SU=70 / <integer 0...100>
IOPT_LOW_DEV_0HM=1 / <integer 0...100>
IOPT_LOW_DEV_0RQ=25 / <integer 0...100>

IOPT_LOW_DEV_1HM, IOPT_LOW_DEV_2HM, IOPT_LOW_DEV_3HM

Threshold values 1, 2 and 3 for utilization by tasks with the priority HIGH or MEDIUM.

IOPT_LOW_DEV_1QH, IOPT_LOW_DEV_2QH, IOPT_LOW_DEV_3QH

Threshold values 1, 2 and 3 for tasks with the priority LOW in the event of a high proportion of tasks with the priority HIGH.

IOPT_LOW_DEV_1QM, IOPT_LOW_DEV_2QM, IOPT_LOW_DEV_3QM

Threshold values 1, 2 and 3 for tasks with the priority LOW in the event of a low proportion of tasks with the priority HIGH.

IOPT_LOW_DEV_0SU, IOPT_LOW_DEV_0HM, IOPT_LOW_DEV_0RQ

Threshold values for restricting the tasks with the priority LOW in the event of high utilization of the device with, at the same time, relatively low utilization by tasks with the priority HIGH or MEDIUM (i.e. IOPT_LOW_DEV_1HM is not reached).

The threshold values have the following effects:

- If utilization by tasks with the priority HIGH or MEDIUM is less than IOPT_LOW_DEV_1HM, the tasks with the priority LOW are not restricted.
- If utilization by tasks with the priority HIGH or MEDIUM is between IOPT_LOW_DEV_1HM and IOPT_LOW_DEV_2HM, then IOPT_LOW_DEV_1QH or IOPT_LOW_DEV_1QM restricts the proportion of tasks with the priority LOW:
  - IOPT_LOW_DEV_1QH if the proportion of tasks with the priority HIGH is high
  - IOPT_LOW_DEV_1QM if the proportion of tasks with the priority HIGH is low
- If utilization by tasks with the priority HIGH or MEDIUM is between IOPT_LOW_DEV_2HM and IOPT_LOW_DEV_3HM, then IOPT_LOW_DEV_2QH or IOPT_LOW_DEV_2QM restricts the proportion of tasks with the priority LOW:
  - IOPT_LOW_DEV_2QH if the proportion of tasks with the priority HIGH is high
  - IOPT_LOW_DEV_2QM if the proportion of tasks with the priority HIGH is low
- If utilization by tasks with the priority HIGH or MEDIUM is greater than IOPT_LOW_DEV_3HM, then IOPT_LOW_DEV_3QH or IOPT_LOW_DEV_3QM restricts the proportion of tasks with the priority LOW:
  - IOPT_LOW_DEV_3QH if the proportion of tasks with the priority HIGH is high
  - IOPT_LOW_DEV_3QM if the proportion of tasks with the priority HIGH is low
- Utilization by tasks with the priority LOW is restricted by IOPT_LOW_DEV_1QH or IOPT_LOW_DEV_1QM even if the following conditions are satisfied:
  - Total utilization of a device exceeds the value IOPT_LOW_DEV_0SU
  - Total utilization by tasks with the priorities HIGH and MEDIUM exceeds the value IOPT_LOW_DEV_0HM
  - Ten times the average number of I/O requests for the device is greater than the value IOPT_LOW_DEV_0RQ

  If the utilization by tasks with the priority HIGH exceeds the value IOPT_LOW_DEV_2HM or IOPT_LOW_DEV_3HM, the restrictions defined for these threshold values apply.
  If, despite considerably lower total utilization after a restriction has been introduced, the utilization by tasks with the priority HIGH or MEDIUM does not increase significantly, the restriction is canceled again.

4.1.1.6 Defining threshold values for I/O priority MEDIUM

Defining threshold values for I/O priority MEDIUM and channels

IOPT_MED_CHN defines threshold values for channel utilization. As soon as the channel utilization by tasks with the priority HIGH reaches the threshold values, the proportion of tasks with the priority MEDIUM is restricted.

IOPT_MED_CHN
IOPT_MED_CHN_1H=20 / <integer 0...100>
IOPT_MED_CHN_2H=30 / <integer 0...100>
IOPT_MED_CHN_3H=40 / <integer 0...100>
IOPT_MED_CHN_1Q=50 / <integer 0...100>
IOPT_MED_CHN_2Q=40 / <integer 0...100>
IOPT_MED_CHN_3Q=30 / <integer 0...100>

IOPT_MED_CHN_1H, IOPT_MED_CHN_2H, IOPT_MED_CHN_3H

Threshold values 1, 2 and 3 for utilization by tasks with the priority HIGH.

IOPT_MED_CHN_1Q, IOPT_MED_CHN_2Q, IOPT_MED_CHN_3Q

Threshold values 1, 2 and 3 for tasks with the priority MEDIUM.

The threshold values have the following effects:

- If utilization by tasks with the priority HIGH is less than IOPT_MED_CHN_1H, tasks with the priority MEDIUM are not restricted.
- If utilization by tasks with the priority HIGH is between IOPT_MED_CHN_1H and IOPT_MED_CHN_2H, then IOPT_MED_CHN_1Q restricts the proportion of tasks with the priority MEDIUM.
- If utilization by tasks with the priority HIGH is between IOPT_MED_CHN_2H and IOPT_MED_CHN_3H, then IOPT_MED_CHN_2Q restricts the proportion of tasks with the priority MEDIUM.
- If utilization by tasks with the priority HIGH is greater than IOPT_MED_CHN_3H, then IOPT_MED_CHN_3Q restricts the proportion of tasks with the priority MEDIUM.

Defining threshold values for I/O priority MEDIUM and ports

IOPT_MED_POR defines threshold values for port utilization. As soon as the port utilization by tasks with the priority HIGH reaches the threshold values, the proportion of tasks with the priority MEDIUM is restricted by limits.

IOPT_MED_POR

IOPT_MED_POR_1H=20 / <integer 0...100>
IOPT_MED_POR_2H=30 / <integer 0...100>
IOPT_MED_POR_3H=40 / <integer 0...100>
IOPT_MED_POR_1Q=50 / <integer 0...100>
IOPT_MED_POR_2Q=40 / <integer 0...100>
IOPT_MED_POR_3Q=30 / <integer 0...100>

IOPT_MED_POR_1H, IOPT_MED_POR_2H, IOPT_MED_POR_3H

Threshold values 1, 2 and 3 for utilization by tasks with the priority HIGH.

IOPT_MED_POR_1Q, IOPT_MED_POR_2Q, IOPT_MED_POR_3Q

Threshold values 1, 2 and 3 for tasks with the priority MEDIUM.

The threshold values have the following effects:

- If utilization by tasks with the priority HIGH is less than IOPT_MED_POR_1H, tasks with the priority MEDIUM are not restricted.
- If utilization by tasks with the priority HIGH is between IOPT_MED_POR_1H and IOPT_MED_POR_2H, then IOPT_MED_POR_1Q restricts the proportion of tasks with the priority MEDIUM.
- If utilization by tasks with the priority HIGH is between IOPT_MED_POR_2H and IOPT_MED_POR_3H, then IOPT_MED_POR_2Q restricts the proportion of tasks with the priority MEDIUM.
- If utilization by tasks with the priority HIGH is between IOPT_MED_POR_3H, then IOPT_MED_POR_3Q restricts the proportion of tasks with the priority MEDIUM.

Defining threshold values for I/O priority MEDIUM and paths

IOPT_MED_PTH defines threshold values for path utilization. As soon as the path utilization by tasks with the priority HIGH reaches the threshold values, the proportion of tasks with the priority MEDIUM is restricted.

IOPT_MED_PTH
IOPT_MED_PTH_1H=20 / <integer 0...100>
IOPT_MED_PTH_2H=30 / <integer 0...100>
IOPT_MED_PTH_3H=40 / <integer 0...100>
IOPT_MED_PTH_1Q=50 / <integer 0...100>
IOPT_MED_PTH_2Q=40 / <integer 0...100>
IOPT_MED_PTH_3Q=30 / <integer 0...100>

IOPT_MED_PTH_1H, IOPT_MED_PTH_2H, IOPT_MED_PTH_3H

Threshold values 1, 2 and 3 for utilization by tasks with the priority HIGH.

IOPT_MED_PTH_1Q, IOPT_MED_PTH_2Q, IOPT_MED_PTH_3Q

Threshold values 1, 2 and 3 for tasks with the priority MEDIUM.

The threshold values have the following effects:

- If utilization by tasks with the priority HIGH is less than IOPT_MED_PTH_1H, tasks with the priority MEDIUM are not restricted.
- If utilization by tasks with the priority HIGH is between IOPT_MED_PTH_1H and IOPT_MED_PTH_2H, then IOPT_MED_PTH_1Q restricts the proportion of tasks with the priority MEDIUM.
- If utilization by tasks with the priority HIGH is between IOPT_MED_PTH_2H and IOPT_MED_PTH_3H, then IOPT_MED_PTH_2Q restricts the proportion of tasks with the priority MEDIUM.
- If utilization by tasks with the priority HIGH is greater than IOPT_MED_PTH_3H, then IOPT_MED_PTH_3Q restricts the proportion of tasks with the priority MEDIUM.

Defining threshold values for I/O priority MEDIUM and devices

IOPT_MED_DEV defines threshold values for device utilization. As soon as the device utilization by tasks with the priority HIGH reaches the threshold values, the proportion of tasks with the priority MEDIUM is restricted.

IOPT_MED_DEV
IOPT_MED_DEV_1H=15 / <integer 0...100>
IOPT_MED_DEV_2H=22 / <integer 0...100>
IOPT_MED_DEV_3H=30 / <integer 0...100>
IOPT_MED_DEV_1Q=50 / <integer 0...100>
IOPT_MED_DEV_2Q=35 / <integer 0...100>
IOPT_MED_DEV_3Q=25 / <integer 0...100>
IOPT_MED_DEV_0S=70 / <integer 0...100>
IOPT_MED_DEV_0H=1 / <integer 0...100>
IOPT_MED_DEV_0R=25 / <integer 0...100>

IOPT_MED_DEV_1H, IOPT_MED_DEV_2H, IOPT_MED_DEV_3H

Threshold values 1, 2 and 3 for utilization by tasks with the priority HIGH.

IOPT_MED_DEV_1Q, IOPT_MED_DEV_2Q, IOPT_MED_DEV_3Q

Threshold values 1, 2 and 3 for tasks with the priority MEDIUM.

IOPT_MED_DEV_0S, IOPT_MED_DEV_0H, IOPT_MED_DEV_0R

Threshold values for restricting the tasks with the priority MEDIUM in the event of high utilization of the device with, at the same time, relatively low utilization by tasks with the priority HIGH (i.
e. IOPT_MED_DEV_1HM is not reached).

The threshold values have the following effects:

- If utilization by tasks with the priority HIGH is less than IOPT_MED_DEV_1H, tasks with the priority MEDIUM are not restricted.
- If utilization by tasks with the priority HIGH is between IOPT_MED_DEV_1H and IOPT_MED_DEV_2H, then IOPT_MED_DEV_1Q restricts the proportion of tasks with the priority MEDIUM.

- If utilization by tasks with the priority HIGH is between IOPT_MED_DEV_2H and IOPT_MED_DEV_3H, then IOPT_MED_DEV_2Q restricts the proportion of tasks with the priority MEDIUM.
- If utilization by tasks with the priority HIGH is greater than IOPT_MED_DEV_3H, then IOPT_MED_DEV_3Q restricts the proportion of tasks with the priority MEDIUM.
- Utilization by tasks with the priority MEDIUM is restricted by IOPT_MED_DEV_1Q even if the following conditions are satisfied:
  - Total utilization of a device exceeds the value IOPT_MED_DEV_0S
  - Total utilization by tasks with the priorities HIGH and MEDIUM exceeds the value IOPT_MED_DEV_0H
  - Ten times the average number of I/O requests for the device is greater than the value IOPT_MED_DEV_0R

  If the utilization by tasks with the priority HIGH exceeds the value IOPT_MED_DEV_2H or IOPT_MED_DEV_3H, the restrictions defined for these threshold values apply.If, despite considerably lower total utilization after a restriction has been introduced, the utilization by tasks with the priority HIGH does not increase significantly, the restriction is canceled again.

4.1.1.7 Querying threshold values for I/O priorities

Querying threshold values for the I/O priority LOW

IOPT_LOW? supplies the threshold values which currently apply.

IOPT_LOW?
IOPT_LOW? IOPT_LOW_xxx_iyy?

With no suffix

All threshold values and limits for the utilization of the channels, ports, paths and devices are displayed.

xxx Selects whether the threshold values and limits for channels, ports, paths or devices are displayed. The following specifications are possible:

CHN	Channels
POR	Ports
PTH	Paths
DEV	Devices

iyy Selects whether the first, second or third threshold values or limits are output. The following specifications are possible:

i 1, 2 or 3

yy HM (Threshold value for the tasks with the I/O priorities HIGH and MEDIUM)

QH (Limit for the tasks with the I/O priority LOW in the event of a high proportion of tasks with the priority HIGH)

QM (Limit for the tasks with the I/O priority LOW in the event of a low proportion of tasks with the priority HIGH)

Querying threshold values for the I/O priority MEDIUM

IOPT_MED? supplies the threshold values which currently apply.

IOPT_MED?
IOPT_MED? IOPT_MED_xxx_iy?

With no suffix

All threshold values and limits for the utilization of the channels, ports, paths and devices are displayed.

xxx Selects whether the threshold values and limits for channels, ports, paths or devices are displayed. The following specifications are possible:

CHN	Channels
POR	Ports
PTH	Paths
DEV	Devices

iy Selects whether the first, second or third threshold values or limits are output. The following specifications are possible:

i 1, 2 or 3

y H (Threshold value for the tasks with the I/O priorities HIGH)

Q (Limit for the tasks with the I/O priority MEDIUM)

4.1.1.8 Querying utilization

The IOPT_INF_... statements are used to query the percentage utilization of devices, channels, ports and paths by tasks with the I/O priorities HIGH, MEDIUM and LOW. The explicit and, if required, the implicit lock factors for tasks with the I/O priorities MEDIUM and LOW are also displayed.

The lock factor is the ratio of the duration of the IORM lock to the I/O duration.

Tasks with the I/O priority MEDIUM or LOW are explicitly locked for a channel (or a port, path or device) when the first threshold value for MEDIUM or LOW for channel utilization (or port, path or device utilization) is exceeded.

Tasks with the I/O priority MEDIUM or LOW are implicitly locked for a device when a channel, port or path to which the device is connected is explicitly locked.

Devices

IOPT_INF_DEV?
IOPT_INF_DEV?mask

mask Complete or masked specification of a mnemonic device name. The following specifications are possible:

- * The utilization of all devices is listed.
- n* The utilization of all devices whose names begin with n is listed.
- nm* The utilization of all devices whose names begin with nm is listed.
- nml* The utilization of all devices whose names begin with nml is listed.
- nmlo The utilization of the device with the name nmlo is listed.

Channels

IOPT_INF_CHN?
IOPT_INF_CHN?mask

mask Specifies a channel. The following specifications are possible:

- * The utilization of all channels is listed.
- nm* The utilization of the channel nm is listed.

Paths

IOPT_INF_PTH?
IOPT_INF_PTH?mask

mask Specifies a path. The following specifications are possible:

- * The utilization of all paths is listed.

nmlo/pq The utilization of the path from channel pq to controller nmlo is listed.

Ports

IOPT_INF_POR?
IOPT_INF_POR?mask

mask Specifies a port. The following specifications are possible:

* The utilization of all ports is listed.

portname The utilization of the port with the specified name is listed.
The WWPN (World Wide Port Name – 16 characters) of the controller port can be specified as the port name.

Example

I/Os are active for the devices 3801 and 3803 on the 3800 controller and on channel 50. The FC port connected to the controller has the WWPN 5006048448586C01.

Querying the utilization on channel 50

```
IOPT_INF_CHN?50
```

Output:

```
    UNIT NAME     | HIG MED LOW | EXPLICIT DELAY | IMPLICIT DELAY
                  |  %   %   %  |   MED     LOW  |   MED     LOW
------------------+-------------+----------------+---------------
50                |  24  19  13 |    0      0.01 |    0       0
```

Query for path 3800/50

```
IOPT_INF_PTH?3800/50
```

Output:

```
    UNIT NAME     | HIG MED LOW | EXPLICIT DELAY | IMPLICIT DELAY
                  |  %   %   %  |   MED     LOW  |   MED     LOW
------------------+-------------+----------------+---------------
3800/50           |  24  19  13 |    0      0.01 |    0       0
```

Query for port 5006048448586C01

```
IOPT_INF_POR?5006048448586C01
```

Output:

```
    UNIT NAME      | HIG MED LOW | EXPLICIT DELAY | IMPLICIT DELAY
                   |  %   %   %  |   MED     LOW  |   MED     LOW
-------------------+-------------+----------------+---------------
5006048448586C01   |  24  19  13 |    0      0.01 |    0       0
```

Query for devices 3801 and 3803:

```
IOPT_INF_DEV?380*
```

Output:

```
    UNIT NAME      | HIG MED LOW | EXPLICIT DELAY | IMPLICIT DELAY
                   |  %   %   %  |   MED     LOW  |   MED     LOW
-------------------+-------------+----------------+---------------
 3800              |  0   0   0  |    0       0   |    0       0
 3801              |  70  0   11 |    0      5.81 |    0       0
 3802              |  0   0   0  |    0       0   |    0       0
 3803              |  0   60  29 |    0      1.62 |    0       0
```

4.1.1.9 Check mode

Before IOPT is activated using IOPT_SET_ON=YES, IOPT means can be used to check whether it makes sense to use IOPT.

The following requirements apply for check mode:

- IOPT must be activated using the CHK option (IOPT_SET_ON=CHK, see "Activating and deactivating IOPT").
- I/O priority classes must be defined for tasks (`/MODIFY-TASK-CATEGORIES` or IOPT statement IOPT_PRI (see "Defining and querying threshold values for I/O priority classes")).
- The threshold values for the utilization by tasks with the various priority classes must be defined (see IOPT statements IOPT_LOW_... and IOPT_MED_...).

The IOPT_CHK_RESET statement defines the time at which the check should begin. You can subsequently (for example an hour or day later) use IOPT_CHK_...? statements to query how often IOPT has locked low-priority tasks.

The check period taken into account always begins with the IOPT_CHK_RESET statement and ends at the time a query is made.

Starting the check

IOPT_CHK_RESET
IOPT_CHK_RESET

Querying the locks for channels

IOPT_CHK_CHN?
IOPT_CHK_CHN?mask

mask Specifies a channel. The following specifications are possible:

- `*` The locks for all channels is listed.
- nm The locks for channel nm are listed.

Querying the locks for paths

IOPT_CHK_PTH?
IOPT_CHK_PTH?mask

mask Specifies a path. The following specifications are possible:

- `*` The locks for all paths are listed.
- nmlo/pq The lock for the path from channel pq to controller nmlo is listed.

Querying the locks for ports

IOPT_CHK_POR?
IOPT_CHK_POR?mask

mask Specifies a port. The following specifications are possible:

* The locks for all ports are listed.

portname The locks for the port with the specified name are listed.

Querying the locks for devices

IOPT_CHK_DEV?
IOPT_CHK_DEV?mask

mask Complete or masked specification of a mnemonic device name. The following specifications are possible:

* The locks for all devices are listed.

n* The locks for all devices whose names begin with n are listed.

nm* The locks for all devices whose names begin with nm are listed.

nml* The locks for all devices whose names begin with nml are listed.

nmlo The lock for the device with the name nmlo is listed.

Example

I/Os are active for the devices 3801 and 3803 on the 3800 controller and on channel 50. The FC port connected to the controller has the WWPN 5006048448586C01.

Query for channels

```
IOPT_CHK_CHN?*
```

Output:

```
    UNIT NAME     | FREQUENCY %
------------------+-------------
50                |      73
```

73 % of the lock conditions for channel 50 were satisfied.

Additional quantitative analysis for channel 50:

```
IOPT_INF_CHN?50
```

Output:

```
    UNIT NAME      | HIG MED LOW | EXPLICIT DELAY | IMPLICIT DELAY
                   |  %   %   %  |   MED     LOW  |   MED     LOW
-------------------+-------------+----------------+--------------
50                 |  22  22  44 |    0       0   |    0       0
```

Query for paths

```
IOPT_CHK_PTH?*
```

Output:

```
    UNIT NAME     | FREQUENCY %
------------------+-------------
3800/50           |      73
```

73 % of the lock conditions for path 3800/50 (controller/channel) were satisfied.

Additional quantitative analysis for path 3800/50:

```
IOPT_INF_PTH?3800/50
```

Output:

```
    UNIT NAME      | HIG MED LOW | EXPLICIT DELAY | IMPLICIT DELAY
                   |  %   %   %  |   MED     LOW  |   MED     LOW
-------------------+-------------+----------------+--------------
3800/50            |  22  22  44 |    0       0   |    0       0
```

Query for ports

```
IOPT_CHK_POR?*
```

Output:

```
    UNIT NAME     | FREQUENCY %
------------------+-------------
5006048448586C01  |      73
```

73 % of the lock conditions for port 5006048448586C01 were satisfied.

Additional quantitative analysis for port 5006048448586C01:

```
IOPT_INF_POR?5006048448586C01
```

Output:

```
    UNIT NAME      | HIG MED LOW | EXPLICIT DELAY | IMPLICIT DELAY
                   |  %   %   %  |   MED     LOW  |   MED     LOW
-------------------+-------------+----------------+---------------
5006048448586C01   |  22  22  44 |    0       0   |    0       0
```

Query for devices

```
IOPT_CHK_DEV?*
```

Output:

```
    UNIT NAME     | FREQUENCY %
------------------+-------------
3801              |       73
3803              |       73
```

73 % of the lock conditions for devices 3801 and 3803 were satisfied.

Additional quantitative analysis for devices 3801 and 3803:

```
IOPT_INF_DEV?380*
```

Output:

```
    UNIT NAME     | HIG MED LOW | EXPLICIT DELAY | IMPLICIT DELAY
                  |  %   %   %  |   MED     LOW  |   MED     LOW
------------------+-------------+----------------+---------------
 3800             |  0   0   0  |    0       0   |    0       0
 3801             |  48  0   48 |    0       0   |    0       0
 3802             |  0   0   0  |    0       0   |    0       0
 3803             |  0   48  48 |    0       0   |    0       0
```

With this utilization it makes sense to have the I/Os for devices 3801 and 3803 controlled by IOPT.

4.1.2 Typical applications

The following examples show the IOPT statements for two different scenarios.

Example 1

IOPT monitors all devices. The settings of `/MODIFY-TASK-CATEGORIES` or the default values of the IOPT statement IOPT_PRI (see "Defining and querying threshold values for I/O priority classes") apply for the I/O priority classes:

```
/START-IORM
IOPT_SET_ON=YES
IOPT_DEV_ADD=ALL
END
```

Here the IOPT statements can be entered in dialog mode or stored in the `SYSDAT.IORM.nnn` file.

Example 2

The logical devices 8800, 8801, 8810, 8820, 8821 and 8822 are located on a physical device. Applications with IO-PRIO=HIGH are active on 8800 and 8801, and applications with IO-PRIO=LOW on 8810, 8820, 8821 and 8822. The applications with IO-PRIO=LOW disturb the applications with IO-PRIO=HIGH. IORM does not recognize a clash on the logical devices because the utilization of the ports, paths and channels is still below the specified threshold values. Consequently a device group 001 is defined.

The settings of `/MODIFY-TASK-CATEGORIES` or the default values of the IOPT statement IOPT_PRI (see "Defining and querying threshold values for I/O priority classes") apply for the I/O priority classes:

```
/START-IORM
IOPT_GRP_001_ADD=D-R(8800,8801)
IOPT_GRP_001_ADD=D-R(8810,8810)
IOPT_GRP_001_ADD=D-R(8820,8822)
END
```

Here the IOPT statements can be entered in dialog mode or stored in the `SYSDAT.IORM.nnn` file.

4.2 DPAV: Dynamic Parallel Access Volume

Parallel disk access is offered on all BS2000 servers via Parallel Access Volumes (“static PAV”) as an alternative to single disk access (default), see the “Introduction to System Administration” [5 (Related publications)]. PAV enables the response times to be reduced in the case of high disk utilization.

Static PAV requires planning that anticipates the future device utilization, i.e. the heavily utilized devices must be assigned the right number of alias devices in advance. When hardware generation takes place in BS2000, it must be borne in mind that a separate address is required for each alias device. When an alias device is defined for each device, only 128 devices can be defined for a logical controller because a maximum of 256 devices can be connected to a logical controller.

I/O load balancing using DPAV

The IORM function DPAV (**D**ynamic **P**arallel **A**ccess **V**olume) is offered for disk devices connected to the Fibre Channel of /390 servers. DPAV reacts to peak loads by assigning alias devices, autonomously and dynamically, to the devices which profit most from this.

DPAV dynamically assigns alias devices which have been configured as “DPAV” devices to the heavily utilized devices. Not as many alias devices need to be generated in total any more. I/O bottlenecks caused by multiple jobs accessing the same disk are thus eased by automatically attaching alias paths.

DPAV supports Extended PAV (XPAV), see the “Introduction to System Administration” [5 (Related publications)].

> ¡ For Server Units that support FastDPAV, it is no longer recommended to use DPAV.

Configuring DPAV devices

DPAV uses only generated alias devices.

As the device number of a base device on a type FC channel must be less than the device number of the associated alias devices (see the DVC statement in the “System Installation” manual [7 (Related publications)]), it is advisable to generate alias devices in the “rear” area of a controller.

Example

> For a controller with 256 devices (device numbers 8000 through 80FF), the alias devices with the device numbers 80C0 through 80FF satisfy this condition for all base devices (device numbers 8000 through 80BF) that come into question.
>
> However, if, for example, 8010 was generated as an alias device for 8000, it cannot be switched to a base device between 8011 and 80FF.

Alias devices on the type FC channel are switched from a base device to another base device by dynamic modification of the I/O configuration. To do this, an alias device is temporarily removed from the I/O configuration (`/REMOVE-IO-UNIT`) and then entered again with the same logical unit number as the new base device (`/ADD-IO-UNIT`). The unit address (alias address) is retained.

Activating DPAV

At first DPAV is deactivated. The DPAV function is activated using DPAV_SET_ON=YES. DPAV_DEV_ADD is used to determine alias devices for DPAV use. All alias devices which have been generated but which are not intended for DPAV can (only) be used for the static PAV.

In VM2000 mode the function must be activated in the monitor system and on each BS2000 guest system on which DPAV is to be active. The actual switchover of alias devices is coordinated and executed by DPAV in the monitor system.

4.2.1 Statements

- Activating and deactivating DPAV
- Activating and deactivating alias devices for DPAV
- Check mode
- Activating and deactivating base devices for DPAV

4.2.1.1 Activating and deactivating DPAV

DPAV_SET_ON activates or deactivates DPAV. The current setting can also be queried.

DPAV_SET_ON
DPAV_SET_ON=YES DPAV_SET_ON=NO DPAV_SET_ON?

YES DPAV is activated. In VM mode this setting is required on all systems on which DPAV is to be active.

NO DPAV is deactivated on the system. In VM mode this setting is required on all systems on which DPAV is not to be active.

? The current setting is queried.

4.2.1.2 Activating and deactivating alias devices for DPAV

Activating alias devices for DPAV

DPAV_DEV_ADD activates alias devices for DPAV.

DPAV_DEV_ADD
DPAV_DEV_ADD=ALL[,vm-index] DPAV_DEV_ADD=D-R(mn1,mn2)[,vm-index]

ALL All alias devices are activated for DPAV.

D-R(mn1,mn2)

All alias devices with mnemonic names from mn1 to mn2 are activated. If mn1 and mn2 are identical, only one device is activated.

vm-index

Identifies a VM.

Special features in VM2000 operation:

- DPAV_DEV_ADD is entered in the monitor system only.
- When a VM index is specified, DPAV uses the alias devices only for the guest system selected.
- If no VM index is specified, DPAV uses the alias devices for all guest systems.

Querying activated devices

DPAV_DEV_ADD? displays a list of all alias devices which are activated for DPAV.

DPAV_DEV_ADD?
DPAV_DEV_ADD?mask

mask Complete or masked specification of a mnemonic device name. The following specifications are possible:

*	All activated alias devices are listed.
n*	All activated alias devices whose names begin with n are listed.
nm*	All activated alias devices whose names begin with nm are listed.
nml*	All activated alias devices whose names begin with nml are listed.
nmlo	The alias device with the name nmlo is listed.

Deactivating alias devices for DPAV

DPAV_DEV_REM deactivates alias devices for DPAV.

DPAV_DEV_REM
DPAV_DEV_REM=ALL DPAV_DEV_REM=D-R(mn1,mn2)

ALL All alias devices are deactivated for DPAV.

D-R(mn1,mn2)

All alias devices with mnemonic names from mn1 to mn2 are deactivated. If mn1 and mn2 are identical, only one alias device is deactivated.

Special features in VM2000 operation:

- DPAV_DEV_REM may only be specified on the monitor system.

4.2.1.3 Check mode

The potential benefit of DPAV can already be checked with the static PAV.
To do this, first of all DPAV is activated (DPAV_SET_ON=YES).
The DPAV_CHK_RESET statement then defines the time at which a check should begin. The following can be queried later (for example an hour or a day later):

- Query of all devices for which an additional alias device would have made sense (DPAV_CHK_DEV? statement)
- Query of the existing alias device and their I/O activity (DPAV_CHK_ALI? statement)

The check period taken into account always begins with the DPAV_CHK_RESET statement and ends at the time a query is made.

As the check takes place in normal DPAV mode, alias names can be activated immediately after a check.

Starting the check

DPAV_CHK_RESET
DPAV_CHK_RESET

Querying the devices for which an alias device would make sense

DPAV_CHK_DEV checks the devices for which an alias device would have made sense and displays the devices with the frequency > 0.

DPAV_CHK_DEV?
DPAV_CHK_DEV?mask

mask Complete or masked specification of a mnemonic device name. The following specifications are possible:

*	All devices are checked.
n*	All devices whose names begin with n are checked.
nm*	All devices whose names begin with nm are checked.
nml*	All devices whose names begin with nml are checked.
nmlo	The device with the name nmlo is checked.

Querying existing alias devices and their I/O activity

DPAV_CHK_ALI? lists existing alias devices and displays their I/O activity (average I/Os per second). The I/O activity relates only to the home system. In VM mode, additional I/Os could be active on other guest systems.

DPAV_CHK_ALI?
DPAV_CHK_ALI?mask

mask Complete or masked specification of a mnemonic device name. The following specifications are possible:

*	All alias devices are listed.
n*	All alias devices whose names begin with n are listed.
nm*	All alias devices whose names begin with nm are listed.
nml*	All alias devices whose names begin with nml are listed.
nmlo	The alias device with the name nmlo is listed.

Example

Two I/O requests are always pending for device 3801 which is connected to controller 3800 (devices 3800 ... 38FF); an alias device 3807 is assigned. Multiple I/O requests are often pending for device 3803; however, device 3803 is not assigned an alias device.

Querying all devices for which an additional alias device would make sense

```
DPAV_CHK_DEV?*
```

Output:

```
 UNIT NAME   | FREQUENCY %
-------------+------------
3803         |     100
```

The requirements for an additional device were satisfied 100% for device 3803.

Querying existing alias devices and their I/O activity

In the next step a check is made to see whether alias devices exist on controller 3800:

```
DPAV_CHK_ALI?38*
```

Output:

```
 UNIT NAME   | I/O PER SEC
-------------+------------
380A         |         0
380B         |         0
380C         |         0
380D         |         0
380E         |         0
380F         |         0
3804         |         0
3805         |         0
3806         |         0
3807         |      2569
3808         |         0
3809         |         0
```

2569 I/Os per second are active on alias device 3807; the remaining alias devices show no I/O activity on the home system.

Activating alias devices for DPAV

The following assignment of alias devices is made on the basis of the check that was performed:

```
DPAV_DEV_ADD=D-R(3804,3806)
```

```
DPAV_DEV_ADD=D-R(3808,380F)
```

Alias device 3807 remains permanently assigned to device 3801; the other alias devices on controller 3800 are activated for DPAV.

If no alias devices are generated on the controller, these can be configured (using IOGEN or /ADD-IO-UNIT).

Starting another check

The effectiveness of the new assignment of the alias devices is checked by starting a new check period:

```
DPAV_CHK_RESET
```

The devices for which an alias device would make sense are then queried again (after a few minutes):

```
DPAV_CHK_DEV?*
```

Output:

```
 UNIT NAME   | FREQUENCY %
-----------+------------
           |
```

Device 3803 is no longer listed.

4.2.1.4 Activating and deactivating base devices for DPAV

The following statements can be used to specify base devices for DPAV. In VM2000 mode, the statements have to be entered in the monitor system.

Directly after loading the IORM subsystem, all base devices for DPAV are allowed/activated.

Activating base devices for DPAV

DPAV_BAS_ADD activates base devices for DPAV.

DPAV_BAS_ADD
DPAV_BAS_ADD=ALL[,number][,vm-index] DPAV_BAS_ADD=D-R(mn1,mn2)[,number][,vm-index]

ALL All base devices are activated for DPAV.

D-R(mn1,mn2)

All base devices with mnemonic names from mn1 to mn2 are activated. If mn1 and mn2 are identical, only one base device is activated.

number

Maximum number of alias devices for a specified base device. Default value: 7.

vm-index

Identifies a VM.

Special features in VM2000 operation:

- When a VM index is specified, DPAV uses the base devices only for the guest system selected.
- If no VM index is specified, DPAV uses the base devices for all guest systems.

Querying activated base devices

DPAV_BAS_ADD? displays a list of all base devices which are activated for DPAV.

DPAV_BAS_ADD?
DPAV_BAS_ADD?mask

mask Complete or masked specification of a mnemonic device name. The following specifications are possible:

*	All activated base devices are listed.
n*	All activated base devices whose names begin with n are listed.
nm*	All activated base devices whose names begin with nm are listed.
nml*	All activated base devices whose names begin with nml are listed.

nmlo The base device with the name nmlo is listed.

Deactivating base devices for DPAV

DPAV_BAS_REM deactivates base devices for DPAV.

DPAV_BAS_REM
DPAV_BAS_REM=ALL DPAV_BAS_REM=D-R(mn1,mn2)

ALL All base devices are deactivated for DPAV.

D-R(mn1,mn2)

All base devices with mnemonic names from mn1 to mn2 are deactivated. If mn1 and mn2 are identical, only one base device is deactivated.

4.2.2 Typical application

In the monitor system

```
/START-IORM           ------------------------------------- (1)
```

(1) IORM is started and initialized in the monitor system.
The statements of `SYSDAT.IORM.<version>` in the monitor system are processed. These statements look like this:

```
DPAV_SET_ON=YES       ------------------------------------- (2)
DPAV_DEV_ADD=D-R(88D3,88D5) ----------------------------- (3)
DPAV_DEV_ADD=D-R(88E3,88E5),8  -------------------------- (4)
DPAV_DEV_ADD=D-R(88F3,88F5),10  ------------------------- (5)
END                ------------------------------------- (6)
```

(2) The DPAV function is activated in the monitor system.

(3) The alias devices 88D3, 88D4 and 88D5 may be used for DPAV in the monitor system and in all guest systems.

(4) The alias devices 88E3, 88E4 and 88E5 may be used for DPAV in the guest system with the VM index 8.

(5) The alias devices 88F3, 88F4 and 88F5 may be used for DPAV in the guest system with the VM index 10.

(6) All other generated alias devices are not permitted for DPAV. They remain statically assigned to the base devices.

In the guest systems

```
/START-SUBSYSTEM IORM ------------------------------------ (7)
/START-IORM           ------------------------------------- (8)
```

(7) The subsystem IORM is started in the guest system.

(8) IORM is started and initialized in the guest system.
The statements of `SYSDAT.IORM.`<version> in the guest system are processed. These statements look like this:

```
DPAV_SET_ON=YES    ------------------------------------------ (9)
END
```

(9) The DPAV function is activated in the guest system.

4.2.3 FastDPAV

The “FastDPAV” function, an optimized DPAV, is offered for Server Units SU /390 that support a modification of the logical unit number (LUN) for alias devices when starting an I/O. For these Server Units, DPAV is no longer recommended. For FastDPAV, also see the “System Administration” manual [5].

FastDPAV does not require monitoring of the device utilization and the switchover of alias devices by IORM, but it requires an activation using DPAV_SET_ON=YES.

Activation of FastDPAV alias devices (using DPAV_DEV_ADD) is not necessary.

However, querying the I/O rate for FastDPAV alias devices (using DPAV_CHK_ALI) is supported.

By default, all FastDPAV base devices are taken into account for using FastDPAV alias devices. The DPAV_BAS_REM statement can be used to exclude FastDPAV base devices from using FastDPAV alias devices. They can be included again, using DPAV_BAS_ADD. At the same time, the maximum number of aliases per disk can be modified. The default value is the same as the maximum value of 7.

For FastDPAV, the previously executed IORM statements are entered and executed in the respective local BS2000 system.

4.3 DDAL: Optimized Load Balancing in ETERNUS CS operation

BS2000 knows two procedures for selecting tape devices:

- Selection of a suitable tape device in the order of generation.
  If multiple tape devices are in use at the same time, this can result in unfavorable load balancing. In an ETERNUS CS, for example, a number of devices connected to an ICP (Integrated Channel Processor) can be in use, while at the same time no devices are active on other ICPs.
- Optimized device allocation for ETERNUS CS .
  The device management counts the number of active devices for each ICP and takes this counter into account for device allocation.
  The optimized device allocation is set using

  `/MODIFY-MOUNT-PARAMETER NEXT-TAPE-MOUNT=*BY-CONTROLLER.`

However, in both cases the device management knows only the device reservations within a native system or within a BS2000 guest system under VM2000.

Optimized device allocation with DDAL

In native mode the BS2000 device management implements optimized device selection for ETERNUS CS independently of DDAL.

In VM2000 mode on /390 servers the DDAL (**D**ynamic **D**evice **A**llocation) function implements optimized device allocation for ETERNUS CS over **all** BS2000 guest systems of a server.

To permit this, IORM must be started on the monitor system and on all BS2000 guest systems, and the DDAL function must be activated using `DDAL_SET_ON=YES`.

Internal communication ensures that IORM knows the allocation of the ICPs in all BS2000 guest systems of the server. When a device is allocated, IORM makes the global allocation counters available to the local device management.

Activating and deactivating DDAL

DDAL_SET_ON activates or deactivates DDAL. The current setting can also be queried.

DDAL_SET_ON
DDAL_SET_ON=YES DDAL_SET_ON=NO DDAL_SET_ON?

YES DDAL is activated. In VM mode this setting is required on all guest systems on which DDAL is to be active.

NO DDAL is deactivated on the system. In VM2000 operation this setting is required on all guest systems on which DDAL should not be active.

? The current setting is queried.

4.4 TCOM: Dynamic Tape Compression

To ensure optimum data backup on LTO tapes, a minimum data transfer rate must be achieved so that the tapes runs continuously (“streaming mode”).

If the minimum data transfer rate is not achieved, the tape is slowed by the device, rewound a little, and then repositioned behind the last data written. This procedure (start/stop mode) is not only time-consuming, but also reduces the service life of the tapes.

The required minimum data transfer rate can be achieved using ARCHIVE and data on high-speed disks and “large” Raid systems (see the “Introduction to System Administration” [5 (Related publications)]). It is not achieved with low-speed disks.

“In between” there is an area in which the minimum data transfer rate is achieved if compression on the device is switched off. The tape capacity is correspondingly lower when compression is switched off.

Selecting the compression using TCOM

Compression on the LTO device is switched on and off using the TCOM (Dynamic **T**ape **Com**pression) function. By default, compression (also without IORM) is switched on.

TCOM also enables compression to be switched on and off dynamically, i.e. in accordance with the data transfer rate. In this case

- compression is switched off when the data transfer rate without compression is above the minimum value required for streaming mode, but not if compression is used.
- compression is switched on when the data transfer rate is above the minimum value required for streaming mode even if compression is used.

In the case of /390 servers, the device notifies TCOM directly of the data volume transferred from the server to the device and from the device to tape. In the case of x86 servers, only the data volume transferred from the server to the device is known; the data volume transferred from the device to tape is calculated from the compression factor specified.

Controlling compression for LTO devices

TCOM_SET controls whether compression is switched on or off for LTO devices. The setting applies for all LTO devices in the home system. The current setting can also be queried.

TCOM_SET
TCOM_SET=ON TCOM_SET=OFF TCOM_SET=DYN TCOM_SET?

ON Switches compression on for LTO devices (default value).
This setting ensures that optimum use is made of the tape capacity. However, when the data transfer rate is low, the tape often switches to start/stop mode.

OFF Switches compression off for LTO devices.
When the data transfer rate is low, the number of start/stop events can be reduced; the tape capacity decreases in accordance with the compression rate for the data.

DYN Switches compression on or off for LTO devices in accordance with the data transfer rate. This setting combines the benefits of optimum tape capacity when the data transfer rate is high and a low number of start/stop operations when the data transfer rate is low.

? The current setting is queried.

Defining the compression factor

On x86 servers TCOM_FACTOR defines the compression factor for data which is to be saved for TCOM_SET=DYN. The current value can be queried.

TCOM_FACTOR
TCOM_FACTOR=n.m TCOM_FACTOR?

n.m Compression factor on x86 servers and when TCOM_SET=DYN.
Value range: 1.0 <= n.m <= 9.9
Default value: 2.0

? The current setting is queried.

4.5 IOLVM: I/O Limit for Virtual Machines

In VM2000 mode, less important guest systems which are I/O-intensive can hinder other, more important guest systems in I/O operations. This can occur when the I/O-intensive guest systems execute I/Os on the same (logical) device. It also occurs when I/Os are executed on different (logical) devices which are located on the same physical device or are connected over the same paths or can be reached via the same ports or are connected to the same channels.

IOLVM (**I**/**O** **L**imit for **V**irtual **M**achines) can detect conflict situations and specifically slows down I/O operations of the user's own guest system if IO resources that are used jointly (channel, port, path, disk) exceed the specific IO limit for the guest system. The IO limit is specified as a percentage value of the average I/O throughput of the jointly used IO resource.

IOLVM only applies for disk devices.

The I/O limit for IOLVM is defined as the maximum I/O performance utilization of the VM in the MAX-IO-UTILIZATION operand in the VM2000 commands `/CREATE-VM` or `/MODIFY-VM-ATTRIBUTES` and `/CREATE-VM-DEFINITION` or `/MODIFY-VM-DEFINITION`.

In the VM2000 inquiry commands `/SHOW-VM-ATTRIBUTES`, `/SHOW-VM-RESOURCES` and `/SHOW-VM-DEFINITION`, the `MAX-IO` column shows which value is set for the maximum I/O performance utilization of the VM.

5 JMP Reconstruction of ENTER commands from the JMS job pool

Version: JMP V2.0C

JMP (Jobpool Management Program) reconstructs ENTER-JOB commands from information which the JMS has saved in the job pool (system file SYSTEM.JOBPOOL) via accepted batch jobs. JMP writes the commands to a file. From this file they can be called, modified (if needed), and transferred back to the system. You can edit and print out the job pool information. Compared to previous warm start functions, this provides system administration with additional options for restarting the processing of batch jobs (after a change of version, for example).

Bear in mind that, as a rule, the reconstructed ENTER-JOB commands cannot be restarted by the job submitter /system administration without testing and modification. It is not possible to reconstruct all of the job submitters batch job attributes. Thus, you should read section “Notes on reconstructed attributes”, very carefully.

The program can run in batch mode and dialog mode. You must run it under the privileged user ID TSOS.

5.1 Execution of JMP

JMP opens the job pool file (which is in PAM format) and sets up a SAM file (referred to below as a procedure file) for the ENTER commands to be generated.
JMP either determines the file name of the job pool file via a standard link name (link name SJOBPOOL), or the file name is explicitly specified in the JMP statement `//OPEN-JOBPOOL-FILE`. When transferring the name of the job pool file using a link name, you must execute the command `/ADD-FILE-LINK LINK-NAME=SJOBPOOL, FILE-NAME=<filename>` prior to calling JMP.

JMP determines the name of the procedure file in the same way, either via the standard link name PJOBPOOL or via explicit specification in the JMP statement `//CREATE-PROCEDURE-FILE`. The procedure file can be either a newly created file or an existing file which is overwritten. The program is started with `/START-JMP`.

START-JMP	Alias: **JMP**
VERSION = ***STD** / <product-version>	
,MONJV = ***NONE** / <filename 1..54 without-gen-vers>	
,CPU-LIMIT = ***JOB-RE**ST / <integer 1..32767 *seconds*>	

JMP is controlled by means of control statements read from SYSDTA.

Notes on the job pool file

The job pool file named SYSTEM.JOBPOOL that is current during the session is located in the home pubset. It is opened during a session by the job pool task (system task, TSN JOBP) and can therefore not be accessed by the JMP program.

If the home pubset is imported as a data pubset (after a shutdown), JMP has unlimited access to the job pool file. However, the file is protected by a read password and must therefore be copied to an unprotected file using `/COPY-FILE ...,IGNORE-PROTECTION=*SOURCE`, before it can be processed by JMP.

The job pool contains information on all of the jobs currently being processed and the jobs yet to be run. These are current or interrupted batch jobs and all batch jobs that have not yet been started (with repeat jobs or calendar jobs, they are the versions that still need to be started). Immediately prior to a planned shutdown, the job pool file should therefore contain only jobs with a start time that is later than the planned session restart. When the system starts up again (STARTUP), a new job pool file is started and set up with job pool information from the previous session. All batch jobs that no longer need to be considered in the new session are removed.

5.2 Statements

A statement can span several lines. The hyphen is used as the continuation character. It announces an additional line. Only spaces are allowed between the hyphen and the end of the lines.

5.2.1 Overview of JMP statements

Statement	Function
CREATE-PROCEDURE-FILE	Create a SAM file that contains a BS2000 procedure with ENTER commands
OPEN-JOBPOOL-FILE	Open job pool file
SHOW-JOBPOOL-STATUS	Display information on the contents of a job pool file
END	End JMP

5.2.2 Description of the statements

- CREATE-PROCEDURE-FILE - Create SAM file with BS2000 procedure
- END - Terminate statement input
- OPEN-JOBPOOL-FILE - Open job pool file
- SHOW-JOBPOOL-STATUS - Output information on the job pool

5.2.2.1 CREATE-PROCEDURE-FILE - Create SAM file with BS2000 procedure

This statement creates a SAM file that contains a BS2000 procedure with reconstructed ENTER-JOB commands. The file contents and the format of the reconstructed commands are described below with the help of an example.

The statement can be used only after at least one `//OPEN-JOBPOOL-FILE` has been executed.

Format

CREATE-PROCEDURE-FILE
FILE-NAME = *STD-FILE-LINK / <filename 1..54 without-gen-vers> ,OVERWRITE = *NO / *YES

Operands

FILE-NAME = *STD-FILE-LINK / <filename 1..54 without-gen-vers>
Name of the procedure file to be written. The file must be open for write access.

FILE-NAME = *STD-FILE-LINK
The file name should be read from the Task File Table (TFT). The link name is PJOBPOOL; this name cannot be changed by the user.
The user can therefore specify the file name with `/ADD-FILE-LINK LINK-NAME=PJOBPOOL, FILE-NAME=<filename>` prior to running JMP.

FILE-NAME = <filename 1..54 without-gen-vers>
Fully qualified file name. Specification of a file generation or file generation group, or of a file name in “file(no)” format (no=version number) is not allowed.

OVERWRITE =
Causes or prevents overwriting of an existing file with the name specified in FILE-NAME=.

OVERWRITE = *NO
Prevents an existing file from being overwritten. The original file remains unchanged. No procedure file is created. The user sees the message `JMP0012`.

OVERWRITE = *YES
If a file with the same name already exists, it will be overwritten and a procedure file created.

5.2.2.2 END - Terminate statement input

This statement terminates the input of statements to the JMP program.

Format

END

5.2.2.3 OPEN-JOBPOOL-FILE - Open job pool file

This statement specifies the job pool to be reconstructed.

Format

OPEN-JOBPOOL-FILE
FILE-NAME = *STD-FILE-LINK / <filename 1..54 without-gen-vers>

Operands

FILE-NAME = *STD-FILE-LINK / <filename 1..54 without-gen-vers>

Name of the job pool to be reconstructed. The file must be open for read access.

FILE-NAME = *STD-FILE-LINK
The file name should be read from the Task File Table (TFT).
The link name is SJOBPOOL; this name cannot be changed by the user.
The user can therefore specify the file name with `/ADD-FILE-LINK LINK-NAME=SJOBPOOL, FILE-NAME=<filename>` prior to running JMP.

FILE-NAME = <filename 1..54 without-gen-vers>
Fully qualified file name. Specification of a file generation or file generation group, or of a file name in “file(no)” format (no=version number) is not allowed.

5.2.2.4 SHOW-JOBPOOL-STATUS - Output information on the job pool

This statement displays information on a subset of the job descriptions in the currently open job pool file.

The jobs are selected by means of the JOB-IDENTIFICATION operand on the basis of job attributes. You can set the number of attributes displayed by means of the INFORMATION operand. The output destination is set by means of the OUTPUT operand.

Format

```
SHOW-JOBPOOL-STATUS

JOB-IDENTIFICATION = *ALL / *NONE / *TSN(...) / *JOB-STATE (...) / *USER-IDENTIFICATION(...) /
                     *JOB-CLASS(...) / *JOB-NAME(...) / *PUBSET-OF-SYSCMD-FILE(...)
    *TSN(...)
        | TSN = <alphanum-name 1..4>
    *JOB-STATE(...)
        | JOB-STATE = *EXECUTING / *WAITING / *DORMANT / *REPEAT / *CALENDAR
    *USER-IDENTIFICATION(...)
        | USER-IDENTIFICATION = <alphanum-name 1..8>
    *JOB-CLASS(...)
        | JOB-CLASS = <alphanum-name 1..8>
    *JOB-NAME(...)
        | JOB-NAME = <alphanum-name 1..8>
    *PUBSET-OF-SYSCMD-FILE(...)
        | PUBSET-OF-SYSCMD-FILE = <alphanum-name 1..4>
,INFORMATION = *SUMMARY / *JOB-LIST / *FULL
,OUTPUT = list-poss(2): *SYSOUT / *SYSLST
```

Operands

JOB-IDENTIFICATION = *ALL / *NONE / *TSN(...) / *JOB-STATE (...) / *USER-IDENTIFICATION(...) / *JOB-CLASS(...) / *JOB-NAME(...) / *PUBSET-OF-SYSCMD-FILE(...)
Selects the jobs about which information is to be output on the basis of the specified parameters.

JOB-IDENTIFICATION = *ALL
Selects all the jobs in the job pool.

JOB-IDENTIFICATION = *NONE
If INFORMATION = *SUMMARY is selected, information on the current job pool file is displayed. Otherwise, there is no output.

JOB-IDENTIFICATION = *TSN(...)
Selects the job by means of the TSN.

TSN = <alphanum-name 1..4>
TSN of the job to be selected.

JOB-IDENTIFICATION = *JOB-STATE(...)
Selects the jobs on the basis of the state or type of the jobs from the viewpoint of JMS.

JOB-STATE = *EXECUTING
Selects all the active jobs for which there was already a task in the system.

JOB-STATE = *WAITING
Selects all the jobs that are waiting for their system startup under the control of a job stream.

¡ Waiting repeat job repetitions with a repetition count > 0 are not selected.

JOB-STATE = *DORMANT
Selects all the jobs with an inactive job stream or inactive repeat job repetitions.

JOB-STATE = *REPEAT
Selects all repeat jobs.

JOB-STATE = *CALENDAR
Selects all calendar jobs.

USER-IDENTIFICATION = *USER-IDENTIFICATION(...)
Selects all jobs to run under the specified user ID.

USER-IDENTIFICATION = <alphanum-name 1..8>
User ID under which the selected jobs are to run.

JOB-IDENTIFICATION = *JOB-CLASS(...)
Selects all jobs that belong to the specified job class.

JOB-CLASS = <alphanum-name 1..8>
Job class of the jobs that are to be selected.

JOB-IDENTIFICATION = *JOB-NAME(...)
Selects all jobs with the specified job name.

JOB-NAME = *NONE / <alphanum-name 1..8>
Name of the jobs to be selected. If *NONE is specified, jobs without a name are selected.

JOB-IDENTIFICATION = *PUBSET-OF-SYSCMD-FILE(...)
Selects all jobs whose command file is on the specified pubset.

PUBSET-OF-SYSCMD-FILE = <alphanum-name 1..4>
ID of the pubset containing the command file of the selected jobs.

INFORMATION = *SUMMARY / *JOB-LIST / *FULL
Specifies the amount of information to be output on the selected jobs.

INFORMATION = *SUMMARY
Limits the information that is output to the name and version of the current job pool file and the number of selected jobs.
If JOB-INFORMATION = *NONE is selected, the number of jobs is not specified (see example 1).

INFORMATION = *JOB-LIST
Displays a header line, a line with 9 attributes for each selected job and a line that summarizes the number of these jobs (see example 2).

INFORMATION = *FULL
Displays the most important job attributes for each selected job in up to 12 output lines. The output is concluded with a line with the number of jobs (see example 3).

OUTPUT = list-poss(2): *SYSOUT / *SYSLST
Defines the output destination.

OUTPUT = *SYSOUT
Output is to be to SYSOUT.

OUTPUT = *SYSLST
Output is to be to SYSLST.

Example 1

```
%//show-jobpool-status *none,*summary
JOBPOOL :  :2OS6:$YLA.SYSTEM.JOBPOOL.V160
VERSION :  OSD V3.0 OR HIGHER
```

> The JOBPOOL file shown can be used by OSD versions >= V3.0.

Example 2

```
%//show-jobpool-status *job-class(jcjs2xsb),*job-list
TSN  JNAME    USER     JCLASS   CAT   TYPE  START        REPEAT LTSN
0AVX          TSOS     JCJS2XSB G     2     SOON
0AVZ          TSOS     JCJS2XSB G     1 WT  A040809.2300
0AV0          TSOS     JCJS2XSB G     1 WT  E041010.2222
0AV2          TSOS     JCJS2XSB G     2     W0345
0AV3          TSOS     JCJS2XSB G     1 WT  A000809.1200 CALNDR
0AV4          TSOS     JCJS2XSB G     1 DO  STUP         STUP
0AV6 R1       TSOS     JCJS2XSB G     1 DO  A040810.1400 DAILY  0AV5
0AV5 R1       TSOS     JCJS2XSB G     1 WT  A040809.1400 DAILY
0AV8          TSOS     JCJS2XSB G     1 DO  SOON         0234   0AV7
0AV7          TSOS     JCJS2XSB G     2     SOON         0234
0AWB          TSOS     JCJS2XSB G     1 WT  A000809.1600 CALNDR
            11 JOB(S) DISPLAYED
```

Explanations

The JOBPOOL file shown contains 11 job descriptions in the JCJS2XSB job class. The job attributes are output after a header linen:

`TSN:` The jobs are output in the order in which they appear in the job pool file. The TSNs are therefore generally not sorted.

`JNAME:` Job name

`USER:` User ID

`JCLASS:` Job class

`CAT:` Catalog ID of the pubset containing the command file.

TYPE:	1 WT:	The job waits to be started in the job stream. This corresponds to the selection criterion JOB-STATE=*INACTIVE.
	1 DO:	waiting repeat job successor (*DORMANT).
	1 HO:	The stop was suspended with HOLD-JOB.
	1 HOC:	The job is in resource HOLD because the calendar file could not be accessed at scheduling.
	2 :	The job is active (*ACTIVE).
START:	The START attribute in the following formats:	
	SOON	
	IMME	IMMEDIATELY
	STUP	AT-STREAM-STARTUP
	Whhmm	WITHIN(HOURS=hh,MINUTES=mm)
	Ayymmdd.hhmm	AT(DATE=yyyy-mm-dd,TIME=hh:mm)
	Eyymmdd.hhmm	EARLIEST(.. see AT ..)
	Lyymmdd.hhmm	LATEST(.. see AT ..)
REPEAT:	The REPEAT attribute in the following formats:	
	DAILY, WEEKLY	
	STUP	AT-STREAM-STARTUP
	hhmm	PERIOD(HOURS=hh,MINUTES=mm)
	CALENDR	The job is a calendar job.
LTSN:	Only for repeat job repetitions whose predecessor was also in the job pool: TSN of the predecessor (“last” TSN).	

- If the job pool file in question was created with MOVE-JOBS, only the version (of the maximum of two versions) with the highest count was accepted for repeat jobs (the NEXT image in the type 1 DO). It contains the entire repeat information. In this case, there are generally no jobs with the TSN <LTSN>.
  In the case of IMPORT or a warm start, the NEXT image is changed back into its predecessor, which is then given the TSN <LTSN> (if it is still available).
- Information cannot be obtained on repeat jobs that are made available by /SHOW-JOB-STATUS. In particular, the link from the predecessor to the successor repeat job, which is reflected in the START and NTSN output parameters, cannot be obtained in the job pool file. The link is only possible from the successor repeat job to the predecessor repeat job via its TSN (LTSN output parameter).

Example 3

```
%//show-jobpool-status *t(0avx),*full
 TSN:     0AVX
 JOBNAME:            TYPE:    2     USER-ENTER                          1.
 USERID:  TSOS        ACCNB:   ADMINSTR   JCLASS:  JCJS2XSB
 PRI:     8 230       CPU-MAX: 200        START:   SOON
 RERUN:   NO          FLUSH:   NO         SPOOLIN: <date>.<time>
 CREATOR: TSOS        PROTECT: *NONE      LOGON:   <date>.<time>          2.
 ORIGFILE::G:$TSOS.OS232.E
 CMD-FILE:*SAME
          1 JOB(S) DISPLAYED
//show-jobpool-status *t(0awb),*full
 TSN:     0AWB
 JOBNAME:            TYPE:    1 WT  USER-ENTER
 USERID:  TSOS        ACCNB:   ADMINSTR   JCLASS:  JCJS2XSB
 PRI:     8 230       CPU-MAX: 200        START:    A<date>.<time>
 RERUN:   NO          FLUSH:   NO         SPOOLIN:  <date>.<time>
 CREATOR: TSOS        PROTECT: *NONE      LOGON:
 REPEAT:  CALENDAR    COUNT:   0          LIMIT:   *STD                    3.
 SYMDAT:  SYM.16.00
 CAL-NAME::G:$TSOS.ULTIMATIVER.CALENDAR
 ORIGFILE::G:$TSOS.S.199.0AV7.E
 CMD-FILE::G:$TSOS.S.E.0AV7.<date>.<time>
 MONJV:   :G:$TSOS.JV
          1 JOB(S) DISPLAYED
//show-jobpool-status *t(0av6),*full
 TSN:     0AV6
 JOBNAME:            TYPE:    1 DO  USER-ENTER
 USERID:  TSOS        ACCNB:   ADMINSTR   JCLASS:  JCJS2XSB
 PRI:     8 230       CPU-MAX: 200        START:   A<date>.<time>
 RERUN:   NO          FLUSH:   NO         SPOOLIN:  <date>.<time>
 CREATOR: TSOS        PROTECT: *NONE      LOGON:
 REPEAT:  DAILY       COUNT:   1          LTSN:    0AV5                    4.
 RTIME:   <date>.<time>
 ORIGFILE::G:$TSOS.OS232.E
 CMD-FILE::G:$TSOS.S.E.0AV2.<date>.<time>
          1 JOB(S) DISPLAYED
```

Explanations

Most of the attributes output are based on the output of `/SHOW-JOB-STATUS` but differ in their arrangement. The essential differences are as follows:

1. After the `TYPE` output as shown in example 2, the origin of the job is specified:

   `USER-ENTER:` Created with ENTER-JOB or ENTER-PROCEDURE by a user task.
   You can recognize ENTER-PROCEDURE from the name of the CMD-FILE: S.E.<tsn>.<yyyy-mm-dd>.<hh.mm.ss>.

   `PRIVILEGED-ENTER:` Created by the system with a privileged ENTER (e.g. the job scheduler tasks).

   `CONSOLE-ENTER:` Created by a user task with OPERATING privilege using the DEFAULT-FROM-FILE = *YES operand or from a console.

2. `CREATOR:` The ID under which the creating task ran.The job can also be managed from this ID.

   `PROTECT:` `*CANCEL` : The job was started with the ENTER operand PROTECTION=*CANCEL.

   `*NONE` : The job was started with the ENTER operand PROTECTION=*NONE.

3. `REPEAT:` Calendar jobs are displayed here in addition to the original repeat attributes.

   `LIMIT:` Only output with calendar jobs. Has the same meaning as SHOW-JOB-STATUS.

4. `LTSN:` Refers in repeat jobs to the predecessor (see example 2)

5.3 Notes on reconstructed attributes

The following example contains notes on reconstructing individual job attributes and on problem cases. JMP writes the following commands to the procedure file:

```
/"REJECTED:    TSN:  0EY4  TYPE:   1 DO       REASON: NEXT RPT. IMAGE "  1.
/SET-JOB-STEP " ------------------------------------------- "           2.
/"IMPORTED:    TSN:  0EY3  TYPE:   1 DO       ORIGIN: USER-ENTER      "  3.
/" CALLER:     TSN:  0EY2  USERID: TSOS       HOST:                   "
/" REPETITION: REPCNT: 0   START: <date>.<time>"
/ENTER-JOB                                                              - 4.
/  FROM-FILE = :2BV:$TSOS.OS27.E                                        - 5.
/ ,PROC-ADMI = *PAR(                                                    - 6.
/    USER-ID = TSOS                                                     -
/   ,ACCOUNT = ADMINSTR                                                 -
/   ,PASS    = '????????')                                             - 7.
/ ,FILE-PASS = *NONE                                                    - 7.
/ ,DELETE    = *NO                                                      -
/ ,JOB-CLASS = JCBTSOS                                                  -
/ ,JOB-NAME  = OS27KF                                                   -
/ ,MONJV     = *NONE                                                    -
/ ,JV-PASS   = *NONE                                                    -
/ ,JOB-PRIO  = 5                                                        -
/ ,RERUN-AFTER-CRASH    = *NO                                           -
/ ,FLUSH-AFTER-SHUTDOWN = *NO                                           -
/ ,SCHEDULING-TIME = *PAR(                                              -
/    START   = *AT       (DATE = *TODAY    , TIME = 11:00)             - 8.
/   ,REP-JOB = *DAILY )                                                 -
/ ,LIMIT =     *STD                                                     -
/ ,RESOURCES = *PAR(                                                    -
/    RUN-PRIO   =  210                                                  -
/   ,CPU-LIMIT  =  20000                                                -
/   ,SYSLST-LIM =  *NO                                                  -
/   ,SYSOPT-LIM =  *NO )                                                -
/ ,LOGGING = *PAR(                                                      -
/    LISTING = *NO )                                                    -
/ ,PROTECTION= *NONE
/"REJECTED:    TSN:  0EYR  TYPE:   2          REASON: PRIVILEGED ENTER"
/"REJECTED:    TSN:  0EYS  TYPE:   2          REASON: PRIVILEGED ENTER"
/"REJECTED:    TSN:  0EYT  TYPE:   2          REASON: PRIVILEGED ENTER"
/"REJECTED:    TSN:  0EYU  TYPE:   2          REASON: PRIVILEGED ENTER"
/SET-JOB-STEP " ------------------------------------------- "
/"IMPORTED:    TSN:  0EYZ  TYPE:   2          ORIGIN: USER-ENTER      "
/" CALLER:     TSN:  0EYX  USERID: TSOS       HOST:                   "
/ENTER-JOB                                                               -
/  FROM-FILE = :2BV:$TSOS.SYSENT.TCP-IP-AP.031.FTPD                      -
/ ,PROC-ADMI = *PAR(                                                     -
/    USER-ID = TSOS                                                     -
/   ,ACCOUNT = ADMINSTR                                                 -
/   ,PASS    = '????????')                                             -
/ ,FILE-PASS = *NONE                                                     -
/ ,DELETE    = *NO                                                       -
/ ,JOB-CLASS = JCBTSOS                                                   -
/ ,JOB-NAME  = FTPSR                                                     -
/ ,MONJV     = *NONE                                                     -
/ ,JV-PASS   = *NONE                                                     -
/ ,JOB-PRIO  = 5                                                         -
/ ,RERUN-AFTER-CRASH    = *NO                                            -
```

```
/ ,FLUSH-AFTER-SHUTDOWN = *NO                                          -
/ ,SCHEDULING-TIME = *PAR(                                             -
/    START   = *IMMEDIATELY                                            - 9.
/   ,REP-JOB = *NO )                                                   -
/ ,LIMIT =     *STD                                                    -
/ ,RESOURCES = *PAR(                                                   -
/    RUN-PRIO   =  120                                                 -
/   ,CPU-LIMIT  =  *NO                                                 -
/   ,SYSLST-LIM =  *NO                                                 -
/   ,SYSOPT-LIM =  *NO )                                               -
/ ,LOGGING = *PAR(                                                     -
/    LISTING = *NO )                                                   -
/ ,PROTECTION= *NONE
/SET-JOB-STEP " ------------------------------------------ "
/"IMPORTED:    TSN:  0EYW  TYPE:   1 DO       ORIGIN: USER-ENTER      "
/" CALLER:     TSN:  0AAU  USERID: TSOS       HOST:                   "
/" REPETITION: REPCNT: 51  START: 0000-00-00.0000"
/ENTER-JOB                                                             -
/  FROM-FILE = :2BV:$SYSPRIV.SYSENT.HOLD-SAT                           -
/ ,PROC-ADMI = *PAR(                                                   -
/    USER-ID = SYSPRIV                                                 -
/   ,ACCOUNT = SYSACC                                                  -
/   ,PASS    = '????????')                                             -
/ ,FILE-PASS = *NONE                                                   -
/ ,DELETE    = *NO                                                     -
/ ,JOB-CLASS = JCBSTD                                                  -
/ ,JOB-NAME  = HOLDSAT                                                 -
/ ,MONJV     = *NONE                                                   -
/ ,JV-PASS   = *NONE                                                   -
/ ,JOB-PRIO  = 9                                                       -
/ ,RERUN-AFTER-CRASH    = *NO                                          -
/ ,FLUSH-AFTER-SHUTDOWN = *NO                                          -
/ ,SCHEDULING-TIME = *PAR(                                             -
/    START   = *AT-STREAM-STARTUP                                      -
/   ,REP-JOB = *AT-STREAM-STARTUP )                                    -
/ ,LIMIT =     *STD                                                    -
/ ,RESOURCES = *PAR(                                                   -
/    RUN-PRIO   =  220                                                 -
/   ,CPU-LIMIT  =  32000                                               -
/   ,SYSLST-LIM =  *NO                                                 -
/   ,SYSOPT-LIM =  *NO )                                               -
/ ,LOGGING = *PAR(                                                     -
/    LISTING = *NO )                                                   -
/ ,PROTECTION= *NONE
/SET-JOB-STEP " ------------------------------------------ "
/"IMPORTED:    TSN:  0EYY  TYPE:   2          ORIGIN: USER-ENTER      "
/" CALLER:     TSN:  0EYX  USERID: TSOS       HOST:                   "
/ENTER-JOB                                                             -
/  FROM-FILE = :2BV:$TSOS.SYSENT.TCP-IP-AP.031.TELNETD                 -
/ ,PROC-ADMI = *PAR(                                                   -
/    USER-ID = TSOS                                                    -
/   ,ACCOUNT = ADMINSTR                                                -
/   ,PASS    = '????????')                                             -
/ ,FILE-PASS = *NONE                                                   -
/ ,DELETE    = *NO                                                     -
/ ,JOB-CLASS = JCBTSOS                                                 -
/ ,JOB-NAME  = TELSR                                                   -
/ ,MONJV     = *NONE                                                   -
/ ,JV-PASS   = *NONE                                                   -
```

```
/ ,JOB-PRIO  = 5                                                    -
/ ,RERUN-AFTER-CRASH    = *NO                                       -
/ ,FLUSH-AFTER-SHUTDOWN = *NO                                       -
/ ,SCHEDULING-TIME = *PAR(                                          -
/    START   = *IMMEDIATELY                                         -
/   ,REP-JOB = *NO )                                                -
/ ,LIMIT =     *STD                                                 -
/ ,RESOURCES = *PAR(                                                -
/    RUN-PRIO   =  120                                              -
/   ,CPU-LIMIT  =  *NO                                              -
/   ,SYSLST-LIM =  *NO                                              -
/   ,SYSOPT-LIM =  *NO )                                            -
/ ,LOGGING = *PAR(                                                  -
/    LISTING = *NO )                                                -
/ ,PROTECTION= *NONE
/"REJECTED:    TSN:  0EY6  TYPE:   1 DO       REASON: NEXT RPT. IMAGE "
/SET-JOB-STEP " ------------------------------------------ "
/"IMPORTED:    TSN:  0EY5  TYPE:   1 DO       ORIGIN: USER-ENTER      "
/" CALLER:     TSN:  0EY2  USERID: TSOS       HOST:                   "
/" REPETITION: REPCNT: 0   START: <date>.<time>"
/ENTER-JOB                                                          -
/  FROM-FILE = :2BV:$TSOS.OS27.E                                    -
/ ,PROC-ADMI = *PAR(                                                -
/    USER-ID = TSOS                                                 -
/   ,ACCOUNT = ADMINSTR                                             -
/   ,PASS    = '????????')                                          -
/ ,FILE-PASS = *NONE                                                -
/ ,DELETE    = *NO                                                  -
/ ,JOB-CLASS = JCBTSOS                                              -
/ ,JOB-NAME  = OS27KF                                               -
/ ,MONJV     = *NONE                                                -
/ ,JV-PASS   = *NONE                                                -
/ ,JOB-PRIO  = 5                                                    -
/ ,RERUN-AFTER-CRASH    = *NO                                       -
/ ,FLUSH-AFTER-SHUTDOWN = *NO                                       -
/ ,SCHEDULING-TIME = *PAR(                                          -
/    START   = *AT       (DATE = *TODAY    , TIME = 11:00)          -
/   ,REP-JOB = *WEEKLY )                                            -
/ ,LIMIT =     *STD                                                 -
/ ,RESOURCES = *PAR(                                                -
/    RUN-PRIO   =  210                                              -
/   ,CPU-LIMIT  =  20000                                            -
/   ,SYSLST-LIM =  *NO                                              -
/   ,SYSOPT-LIM =  *NO )                                            -
/ ,LOGGING = *PAR(                                                  -
/    LISTING = *NO )                                                -
/ ,PROTECTION= *NONE
/"REJECTED:    TSN:  0EY8  TYPE:   1 DO       REASON: NEXT RPT. IMAGE "
/SET-JOB-STEP " ------------------------------------------ "
/"IMPORTED:    TSN:  0EY7  TYPE:   1 DO       ORIGIN: USER-ENTER      "
/" CALLER:     TSN:  0EY2  USERID: TSOS       HOST:                   "
/" REPETITION: REPCNT: 0   START: <date>.<time>"
/ENTER-JOB                                                          -
/  FROM-FILE = :2BV:$TSOS.OS27.E                                    -
/ ,PROC-ADMI = *PAR(                                                -
/    USER-ID = TSOS                                                 -
/   ,ACCOUNT = ADMINSTR                                             -
/   ,PASS    = '????????')                                          -
/ ,FILE-PASS = *NONE                                                -
```

```
/ ,DELETE    = *NO                                                    -
/ ,JOB-CLASS = JCBTSOS                                                -
/ ,JOB-NAME  = OS27KF                                                 -
/ ,MONJV     = *NONE                                                  -
/ ,JV-PASS   = *NONE                                                  -
/ ,JOB-PRIO  = 5                                                      -
/ ,RERUN-AFTER-CRASH    = *NO                                         -
/ ,FLUSH-AFTER-SHUTDOWN = *NO                                         -
/ ,SCHEDULING-TIME = *PAR(                                            -
/    START   = *SOON                                                  -
/   ,REP-JOB = *PERIOD (HOURS = 01, MINUTES = 02))                    -
/ ,LIMIT =     *STD                                                   -
/ ,RESOURCES = *PAR(                                                  -
/    RUN-PRIO   =  210                                                -
/   ,CPU-LIMIT  =  20000                                              -
/   ,SYSLST-LIM =  *NO                                                -
/   ,SYSOPT-LIM =  *NO )                                              -
/ ,LOGGING = *PAR(                                                    -
/    LISTING = *NO )                                                  -
/ ,PROTECTION= *NONE
/SET-JOB-STEP " ------------------------------------------ "
/"IMPORTED:    TSN:  0EY9  TYPE:   1 WT       ORIGIN: USER-ENTER      "
/" CALLER:     TSN:  0EY2  USERID: TSOS       HOST:                   "
/ENTER-JOB                                                             -
/  FROM-FILE = :2BV:$TSOS.OS27.E                                       -
/ ,PROC-ADMI = *PAR(                                                   -
/    USER-ID = TSOS                                                    -
/   ,ACCOUNT = ADMINSTR                                                -
/   ,PASS    = '????????')                                             -
/ ,FILE-PASS = *NONE                                                   -
/ ,DELETE    = *NO                                                     -
/ ,JOB-CLASS = JCBTSOS                                                 -
/ ,JOB-NAME  = OS27KF                                                  -
/ ,MONJV     = *NONE                                                   -
/ ,JV-PASS   = *NONE                                                   -
/ ,JOB-PRIO  = 5                                                       -
/ ,RERUN-AFTER-CRASH    = *NO                                          -
/ ,FLUSH-AFTER-SHUTDOWN = *NO                                          -
/ ,SCHEDULING-TIME = *PAR(                                             -
/    START   = *EARLIEST (DATE = <date>, TIME = <time>)                -
/   ,REP-JOB = *NO )                                                   -
/ ,LIMIT =     *STD                                                    -
/ ,RESOURCES = *PAR(                                                   -
/    RUN-PRIO   =  210                                                 -
/   ,CPU-LIMIT  =  20000                                               -
/   ,SYSLST-LIM =  *NO                                                 -
/   ,SYSOPT-LIM =  *NO )                                               -
/ ,LOGGING = *PAR(                                                     -
/    LISTING = *NO )                                                   -
/ ,PROTECTION= *NONE
/SET-JOB-STEP " ------------------------------------------ "
/"IMPORTED:    TSN:  0EZA  TYPE:   2          ORIGIN: USER-ENTER      "
/" CALLER:     TSN:  0EY2  USERID: TSOS       HOST:                   "
/" REPETITION: REPCNT: 0   START: <date>.<time>"
/ENTER-JOB                                                             -
/  FROM-FILE = :2BV:$TSOS.OS27.E                                       -
/ ,PROC-ADMI = *PAR(                                                   -
/    USER-ID = TSOS                                                    -
/   ,ACCOUNT = ADMINSTR                                                -
```

```
/   ,PASS    = '????????')                                                -
/ ,FILE-PASS = *NONE                                                      -
/ ,DELETE    = *NO                                                        -
/ ,JOB-CLASS = JCBTSOS                                                    -
/ ,JOB-NAME  = OS27KF                                                     -
/ ,MONJV     = *NONE                                                      -
/ ,JV-PASS   = *NONE                                                      -
/ ,JOB-PRIO  = 5                                                          -
/ ,RERUN-AFTER-CRASH    = *NO                                             -
/ ,FLUSH-AFTER-SHUTDOWN = *NO                                             -
/ ,SCHEDULING-TIME = *BY-CALENDAR(                                        -
/    CALENDAR-NAME =      :2BV:$TSOS.OS27.CALENDAR                        -
/   ,SYMBOLIC-DATE =      HEMUL                )                          -
/ ,LIMIT =     *STD                                                       -
/ ,RESOURCES = *PAR(                                                      -
/    RUN-PRIO   =  210                                                    -
/   ,CPU-LIMIT  =  20000                                                  -
/   ,SYSLST-LIM =  *NO                                                    -
/   ,SYSOPT-LIM =  *NO )                                                  -
/ ,LOGGING = *PAR(                                                        -
/    LISTING = *NO )                                                      -
/ ,PROTECTION= *NONE
/SET-JOB-STEP " ------------------------------------------- "
/"IMPORTED:    TSN:  0EZB  TYPE:   1 WT       ORIGIN: USER-ENTER      "
/" CALLER:     TSN:  0EY2  USERID: TSOS       HOST:                   "
/" REPETITION: REPCNT: 0   START: <date>.<time>"
/ENTER-JOB                                                                -
/  FROM-FILE = :2BV:$TSOS.OS27.E                                          -
/ ,PROC-ADMI = *PAR(                                                      -
/    USER-ID = TSOS                                                       -
/   ,ACCOUNT = ADMINSTR                                                   -
/   ,PASS    = '????????')                                                -
/ ,FILE-PASS = *NONE                                                      -
/ ,DELETE    = *NO                                                        -
/ ,JOB-CLASS = JCBTSOS                                                    -
/ ,JOB-NAME  = OS27KF                                                     -
/ ,MONJV     = *NONE                                                      -
/ ,JV-PASS   = *NONE                                                      -
/ ,JOB-PRIO  = 5                                                          -
/ ,RERUN-AFTER-CRASH    = *NO                                             -
/ ,FLUSH-AFTER-SHUTDOWN = *NO                                             -
/ ,SCHEDULING-TIME = *BY-CALENDAR(                                        -
/    CALENDAR-NAME =      :2BV:$TSOS.OS27.CALENDAR                        -
/   ,SYMBOLIC-DATE =      ELCH                 )                          -
/ ,LIMIT =     *STD                                                       -
/ ,RESOURCES = *PAR(                                                      -
/    RUN-PRIO   =  210                                                    -
/   ,CPU-LIMIT  =  20000                                                  -
/   ,SYSLST-LIM =  *NO                                                    -
/   ,SYSOPT-LIM =  *NO )                                                  -
/ ,LOGGING = *PAR(                                                        -
/    LISTING = *NO )                                                      -
/ ,PROTECTION= *NONE
/SET-JOB-STEP " ------------------------------------------- "
/"IMPORTED:    TSN:  0EZC  TYPE:   2          ORIGIN: USER-ENTER      "
/" CALLER:     TSN:  0EY2  USERID: TSOS       HOST:                   "
/ENTER-JOB                                                                -
/  FROM-FILE = :2BV:$TSOS.OS27.E                                          -
/ ,PROC-ADMI = *PAR(                                                      -
```

```
/    USER-ID = TSOS                                                    -
/   ,ACCOUNT = ADMINSTR                                                -
/   ,PASS    = '????????')                                             -
/ ,FILE-PASS = *NONE                                                   -
/ ,DELETE    = *NO                                                     -
/ ,JOB-CLASS = JCB02000                                                -
/ ,JOB-NAME  = OS27KF                                                  -
/ ,MONJV     = :2BV:$TSOS.OS27.JV.1                                    -
/ ,JV-PASS   = *NONE                                                   -
/ ,JOB-PRIO  = 9                                                       -
/ ,RERUN-AFTER-CRASH    = *YES                                         -
/ ,FLUSH-AFTER-SHUTDOWN = *YES                                         -
/ ,SCHEDULING-TIME = *PAR(                                             -
/    START   = *SOON                                                   -
/   ,REP-JOB = *NO )                                                   -
/ ,LIMIT =     *STD                                                    -
/ ,RESOURCES = *PAR(                                                   -
/    RUN-PRIO   =  210                                                 -
/   ,CPU-LIMIT  =  2000                                                -
/   ,SYSLST-LIM =  *NO                                                 -
/   ,SYSOPT-LIM =  *NO )                                               -
/ ,LOGGING = *PAR(                                                     -
/    LISTING = *NO )                                                   -
/ ,JOB-PAR =                                                           -
/C'WRTLPRMFT'                                                          -
/ ,PROTECTION= *NONE
/SET-JOB-STEP " ------------------------------------------- "
/"IMPORTED:    TSN:  0EZD  TYPE:   2          ORIGIN: USER-ENTER      "
/" CALLER:     TSN:  0EY2  USERID: TSOS       HOST:                   "
/ENTER-JOB                                                              -
/  FROM-FILE = :2BV:$HEMUL.S.IN.0EZD.040323.0913.C                      - 10.
/ ,PROC-ADMI = *PAR(                                                   -
/    USER-ID = HEMUL                                                   -
/   ,ACCOUNT = HEMUL                                                   -
/   ,PASS    = '????????')                                             -
/ ,FILE-PASS = *NONE                                                   -
/ ,DELETE    = *YES                                                    -
/ ,JOB-CLASS = JCB02000                                                -
/ ,JOB-NAME  = HOPPLA                                                  -
/ ,MONJV     = :2BV:$TSOS.OS27.JV.2                                    -
/ ,JV-PASS   = *NONE                                                   -
/ ,JOB-PRIO  = 9                                                       -
/ ,RERUN-AFTER-CRASH    = *YES                                         -
/ ,FLUSH-AFTER-SHUTDOWN = *YES                                         -
/ ,SCHEDULING-TIME = *PAR(                                             -
/    START   = *SOON                                                   -
/   ,REP-JOB = *NO )                                                   -
/ ,LIMIT =     *STD                                                    -
/ ,RESOURCES = *PAR(                                                   -
/    RUN-PRIO   =  230                                                 -
/   ,CPU-LIMIT  =  2000                                                -
/   ,SYSLST-LIM =  *NO                                                 -
/   ,SYSOPT-LIM =  *NO )                                               -
/ ,LOGGING = *PAR(                                                     -
/    LISTING = *NO )                                                   -
/ ,JOB-PAR =                                                           -
/C'OCHGOTTCHEN'                                                        -
/ ,PROTECTION= *CANCEL
/SET-JOB-STEP " ------------------------------------------- "
```

```
/"IMPORTED:    TSN:  0EZE  TYPE:   2          ORIGIN: USER-ENTER      "
/" CALLER:     TSN:  0EY2  USERID: TSOS       HOST:                   "
/" WARNING:    COPY OF SYSCMD-FILE NOT POSSIBLE; COPY-RC: 0501"          11.
/ENTER-JOB                                                             -
/  FROM-FILE = :2BV:$HEMUL.S.IN.0EZE.040323.0929                       -
/ ,PROC-ADMI = *PAR(                                                   -
/    USER-ID = HEMUL                                                   -
/   ,ACCOUNT = HEMUL                                                   -
/   ,PASS    = '????????')                                             -
/ ,FILE-PASS = *NONE                                                   -
/ ,DELETE    = *YES                                                    -
/ ,JOB-CLASS = JCB00050                                                -
/ ,JOB-NAME  = OS27KF                                                  -
/ ,MONJV     = *NONE                                                   -
/ ,JV-PASS   = *NONE                                                   -
/ ,JOB-PRIO  = 9                                                       -
/ ,RERUN-AFTER-CRASH    = *YES                                         -
/ ,FLUSH-AFTER-SHUTDOWN = *NO                                          -
/ ,SCHEDULING-TIME = *PAR(                                             -
/    START   = *EARLIEST (DATE = *TODAY    , TIME = 08:00)             -
/   ,REP-JOB = *NO )                                                   -
/ ,LIMIT =     *STD                                                    -
/ ,RESOURCES = *PAR(                                                   -
/    RUN-PRIO   =  233                                                 -
/   ,CPU-LIMIT  =  100                                                 -
/   ,SYSLST-LIM =  2222                                                -
/   ,SYSOPT-LIM =  3333     )                                          -
/ ,LOGGING = *PAR(                                                     -
/    LISTING = *YES)                                                   -
/ ,PROTECTION= *NONE
```

Explanations

1. Not all jobs contained in the job pool are reconstructed. The example already contains the 'next image' for a repeat job. If this job were reconstructed to an ENTER command and started, it would result in two repeat job versions. Attention is drawn to such unreconstructed jobs in the form of a command comment with information about the job and a list of the reasons. Example:
   - Next Repeat Image
   - Privileged ENTER
2. Start of a reconstructed batch job.

3. Notes on the reconstructed batch job and attributes that cannot be specified with ENTER command operands. The information is generally analogous to that of the STATUS command (see the “Commands” manual [1 (Related publications)]).

   In this example:

   - IMPORTED: Indication that the job which was accepted with the TSN `0EY3` has been reconstructed.
   - CALLER: Indicates the initiator of the batch job (TSN, user ID, host name).
   - REPETITION: Repeat counter and designated start time (only for repeat jobs and calendar jobs).

   > Scheduled/repeat jobs are reconstructed without regard to the start time. If, for example, the start time is prior to the reconstruction run and the reconstructed ENTER command is activated without modification, the command is considered incorrect and will be rejected when accepted again.

4. The jobs are always reconstructed in the form of the SDF command

   `/ENTER-JOB` (regardless of whether the job was started with the ISP command `/ENTER` or via `/ENTER-PROCEDURE`).

5. The command file does not have to be the original file specified by the initiator of the ENTER command. See also note (10).
6. The access control attributes specified with the PROCESSING-ADMISSION operand (user ID, accounting number, password) are only obligatory if the job submitter ID and runtime user ID are different. If both are the same, specification is optional. The operand always appears in the reconstructed command, regardless of whether or not it was specified.

   Note that a reconstructed ENTER-JOB command will be returned by the same user ID as the original one whenever possible. Only in this case will ID-specific default settings continue to be valid. Furthermore, if the job is to run under a different user ID, it can be administered by the same ID as the original one (commands `/CANCEL-JOB`, `/CHANGE-TASK-PRIORITY`, `/SHOW-JOB-STATUS`).
7. Neither the LOGIN password nor any passwords for the ENTER file or monitoring job variables are transferred to the reconstructed ENTER command. This is not possible for technical reasons, since passwords are not generally stored in the JMS data structures. When restarting the ENTER command, you must insert the passwords in non-encrypted format.
8. The original values (including *TODAY, for example) are entered without checking. This becomes important when the original contains fixed dates or attributes such as *WITHIN and *LATEST. The reconstructed values may be wrong as far as the intended repeat time is concerned.

   With repeat jobs, the start attribute may have changed from the original user specification (see also the descriptions for `/ENTER-JOB` and `/ENTER-PROCEDURE` in the “Commands” manual [1 (Related publications)]). In this case, the program attempts to generate a logical start attribute from the repeat attribute and the start time of the next repeat (the start attribute determines only the start of the first run of a repeat job).
9. As described under (8), JMP does not update start times.

10. S.IN/S.E files: JMS creates auxiliary files with this prefix under the followingcircumstances (see also the description for `/ENTER-JOB` and `/ENTER-PROCEDURE` in the “Commands" manual [1 (Related publications)]):

    When `/ENTER-PROCEDURE` is called, an S.E file is created. When `/ENTER-JOB` is called, the original command file is copied to an S.IN intermediate file if the system assumes that the original file will no longer be accessible when the batch job is completed (e.g. if a temporary file was specified as the original command file). This intermediate file created by JMS is protected with a random password. When the reconstructed ENTER command is called again, the intermediate file originally created by JMS can therefore not be accessed. JMP therefore copies the intermediate file into a new intermediate file with no password and the name `<original s.in/s.e filename>.C`.

11. If no copy of the S.IN or S.E file can be created (see note (10)), a command line with the DMS return code (COPY) is displayed.

5.4 JMP messages

The messages of the JMP utility routine have the message class JMP.

See also the section “Messages and their meaning” (Notational conventions).

6 JMU Creating and maintaining the SJMSFILE system file

Version: **JMU V21.0A**

Privileges: **TAPE-PROCESSING** (for nonprivileged functions)
TSOS (changes in ongoing operation and for $TSOS.SJMSFILE)

The JMU (Job Management Utility) program allows you to create and manage the SJMSFILE system file. The SJMSFILE contains the stream and job class definitions, which are stored in an internal table format. The program can run in batch or interactive mode.

At system initialization (BS2000 startup), the SJMSFILE is read and the job class and stream definitions are copied to the system.

In addition, JMU can be used to modify specific JMS data while the system is running:

- You can modify access rights with immediate effect.
- You can assign suitable job classes to new users.
- You can modify, delete and create job classes and job streams.

6.1 Job management

The function of job management is to manage jobs until they are started. The job scheduling system, based on job classes, allows an administrative strategy to be defined for the data center in order to classify users and the system load.

Jobs that share certain characteristics are assigned to the same job class. This applies to jobs in both batch and interactive mode. The relevant characteristics are specified by system administration on defining the classes and determine which user IDs are to be served by a given class.
It is also possible to define default classes, intended for users who have not explicitly specified a class.
By setting a limit for each class and defining class priorities, the data center can improve control over access to the system and can achieve an optimum mix of jobs, e.g. shortrunning vs long-running jobs, at any time of day.

By means of job classes it is possible to classify jobs, for example on the basis of CPU time required, so as to favor short-running jobs over long-running ones. It is also possible for system administration to assign privileges to certain users, such as the right to start scheduled or repeat jobs.

A job management utility (JMU) is available for creating and maintaining the file for stream and job class definitions. For a description of job streams, see the “Introduction to System Administration” [5 (Related publications)].

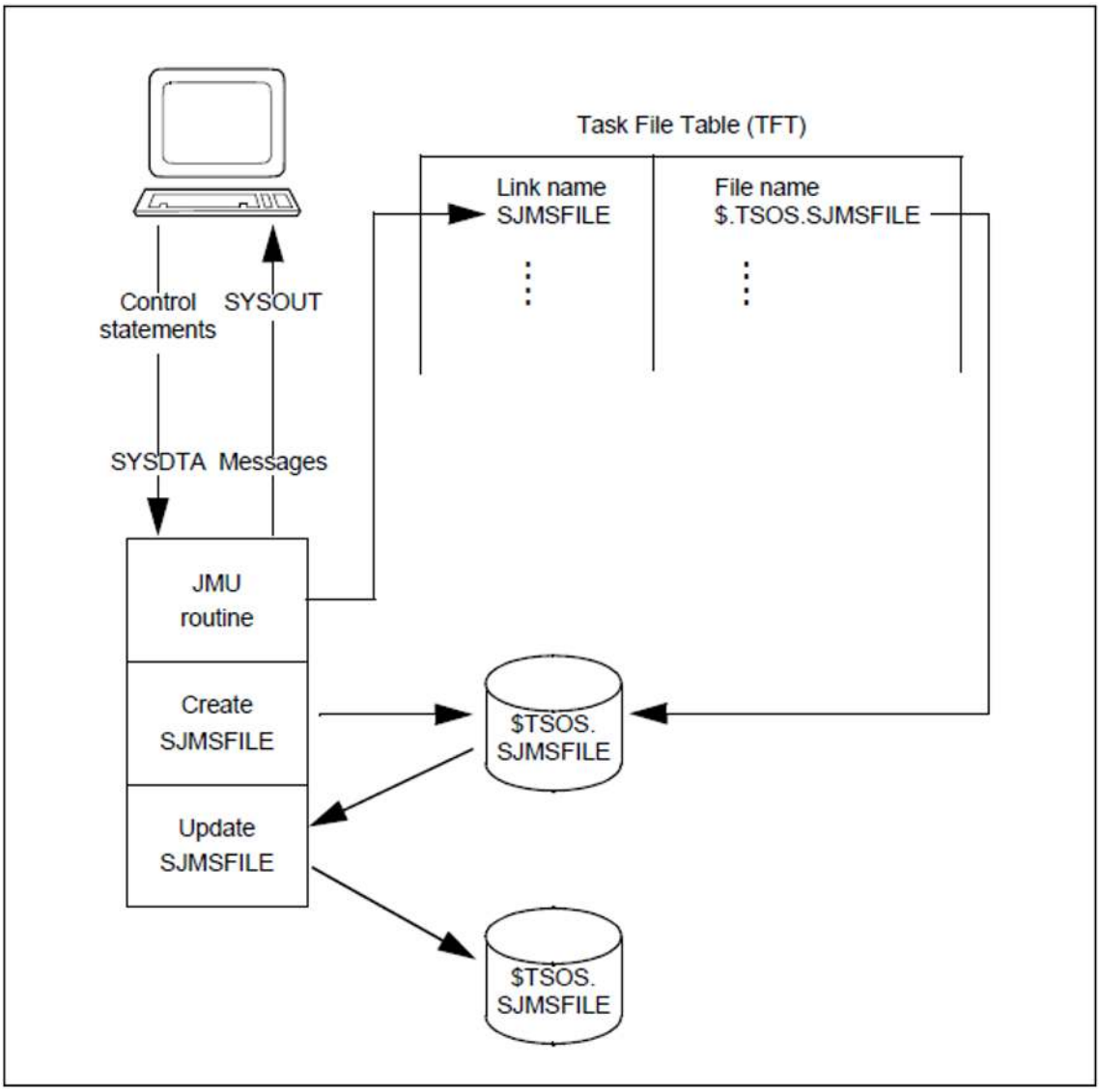


Figure 1: Creating and updating the SJMSFILE

6.2 Execution of JMU

JMU creates the ISAM file SJMSFILE. The file name is ascertained from the task file table (link name SJMSFILE) and can be defined with. If the file already exists, JMU updates it:

```
/ADD-FILE-LINK LINK-NAME=SJMSFILE,FILE-NAME=<filename>
```

The file need not necessarily exist before `/ADD-FILE-LINK` is executed, or it can exist and be empty. In both cases, it is created by JMU.

`/ADD-FILE-LINK` is not mandatory. If the link name SJMSFILE has not been assigned, JMU processes the file with the file name SJMSFILE as before and automatically assigns the link name SJMSFILE to it.

The user must not assign the link name SJMSFILE to a file which is not to be processed by JMU.

The program is started using `/START-JMU`.

START-JMU	Alias: **JMU**
VERSION = ***STD** / <product-version> **,MONJV** = ***NONE** / <filename 1..54 without-gen-vers> **,CPU-LIM**IT = ***JOB-RE**ST / <integer 1..32767 *seconds*>	

JMU is controlled by means of control statements read from SYSDTA.

JMU uses the dialog interface SDF (see the “Commands” manual [1 (Related publications)] and the “SDF Dialog Interface” manual [20 (Related publications)]). Syntax errors in interactive mode cause a correction dialog to be initiated with the user. A correction dialog is not possible in batch or procedure mode, but SDF does not allow the statement to be simply ignored. All the statements following an incorrect statement are skipped until the system encounters a STEP or END statement. Processing then continues with the statement (or command) following the STEP or END statement. Message `CMD0230` is output to inform the user that statements have been omitted.

The SJMSFILE file is updated in the same session in which the control statements were given. However, the updates do not have any effect until the next session.
The desired changes should be entered in a copy of the SJMSFILE, and the updated file used in a session only when the changes have been verified as correct

It is advisable to maintain a copy of the SJMSFILE, or a procedure for reconstructing it.

> There is a risk that it will not be possible to process the SJMSFILE file using the job management systems of operating system versions later than the one used to create the file. It is therefore recommended that a copy of the JMU control statements be preserved for reconstructing the file in a new version of the operating system.

Use of link names by JMU

JMU evaluates the link names SJMSFILE and SJMUPROC. If the SJMSFILE link name is already defined in the task file table, the corresponding file is processed by JMU. Otherwise, SJMSFILE is used as the file name.

When the CREATE-PROCEDURE-FILE statement is processed, the link name SJMUPROC is evaluated and assigned to a file where necessary. This link name should therefore only be used in the cases described under the CREATE-PROCEDURE-FILE statement.

Compatibility

Inconsistencies may occur when a SJMSFILE is processed by an earlier version of JMU than that under which the SJSMFILE was created.

6.3 Statements

A statement can span several lines. The hyphen is used as the continuation character. It announces an additional line. Only spaces are allowed between the hyphen and the end of the lines.

6.3.1 Overview of JMU statements

Statement	Function
CREATE-PROCEDURE-FILE	Create a SAM file containing a BS2000 procedure.
DEFINE-JOB-CLASS	Write a new job class definition.
DEFINE-JOB-STREAM	Write a new stream definition into the SJMSFILE.
DELETE-JOB-CLASS	Delete an existing job class definition.
DELETE-JOB-STREAM	Delete an existing stream definition.
END	Terminate the routine.
GRANT-JOB-CLASS-ACCESS	Grant or prohibit access to a job class for one or more users.
MODIFY-JOB-CLASS	Modify an existing job class definition.
MODIFY-JOB-STREAM	Modify an existing stream definition.
REMOVE-USER	Prohibit access to private job classes.
SET-JOB-CLASS-DEFAULT	Define default classes for users.
SET-MODIFICATION-MODE	Change the modification mode.
SET-POSIX-JOB-CLASS-DEFAULT	Define POSIX default classes for users.
SHOW-JOB-CLASS	List the contents of job class definitions.
SHOW-JOB-STREAM	List the contents of stream definitions.

6.3.2 Description of the statements

- CREATE-PROCEDURE-FILE - Create SAM file containing BS2000 procedure
- DEFINE-JOB-CLASS - Write job class definitions to SJMSFILE
- DEFINE-JOB-STREAM - Write stream definitions to SJMSFILE
- DELETE-JOB-CLASS - Delete class definitions
- DELETE-JOB-STREAM - Delete stream definitions
- END - Terminate statement input
- GRANT-JOB-CLASS-ACCESS - Control access by user IDs to job class
- MODIFY-JOB-CLASS - Modify job class definitions
- MODIFY-JOB-STREAM - Modify stream definitions
- REMOVE-USER - Prohibit access to private job classes
- SET-JOB-CLASS-DEFAULT - Specify default classes for users
- SET-MODIFICATION-MODE - Set modification mode
- SET-POSIX-JOB-CLASS-DEFAULT - Specify POSIX default classes for users
- SHOW-JOB-CLASS - List contents of class definitions or names of classes
- SHOW-JOB-STREAM - List contents of stream definitions or names of streams

6.3.2.1 CREATE-PROCEDURE-FILE - Create SAM file containing BS2000 procedure

This statement can be used to create a SAM file containing a BS2000 procedure. The procedure contains a START command for the JMU program.
When JMU is called during execution of the procedure, a new SJMSFILE system file is written. This replaces and corresponds to the SJMSFILE being processed before the procedure started.

This statement can be used to save the status of an open SJMSFILE during processing by means of BS2000 procedures.

The CREATE-PROCEDURE-FILE statement can be used to update the format of an SJMSFILE. If an existing SJMSFILE is processed with a different version of JMU, its format does not change. An SJMSFILE is only formatted according to the JMU version used when it is created as a new file.

However, as a result of the functional enhancements to JMU, conversion of the SJMSFILE format is only supported if the same JMU version is used to execute both the CREATE-PROCEDURE-FILE statement and the BS2000 procedure.

For this reason the JMU version used to create the procedure is recorded for documentation purposes in a REMARK command. If a different kind of conversion is to be made, the JMU statements in the procedure created may have to be altered in line with the operating instructions for the JMU version to be called.

Format

CREATE-PROCEDURE-FILE
FILE-NAME = *STD-FILE-LINK / <filename 1..54 without-gen-vers> ,OVERWRITE = *NO / *YES

Operands

FILE-NAME =
Name of the procedure file to be written. The file must be open for write access.

FILE-NAME = *STD-FILE-LINK
The file name should be read from the Task File Table (TFT). The link name, which must not be changed by the user, is SJMUPROC. The user can thus define the file name before calling JMU using

```
/ADD-FILE-LINK LINK-NAME=SJMUPROC,FILE-NAME=<filename>
```

If SJMUPROC is not defined as the link name, JMU uses the name SJMUPROC as the file name.

FILE-NAME = <filename 1..54 without-gen-vers>
A fully qualified file name. Specification of a file generation or file generation group, or of a file name in “file(no)” format (no=version number) is not allowed.

OVERWRITE =
Overwriting an existing file with the name specified in the FILE-NAME operand can be initiated or prevented.

OVERWRITE = *NO
Overwriting an existing file is prevented. The original file remains unchanged. The user receives the message `JMU0114`.

No procedure file is created.

OVERWRITE = *YES
If a file with the same name already exists, it will be overwritten and a procedure file created.

Structure of the created BS2000 procedure

The file name of the SJMSFILE to be created and the JMU load module to be called can be specified as procedure parameters.

Meaning of the parameters:

&SJMSFILE	stands for the file name of the SJMSFILE to be created. The file named &SJMSFILE is deleted within the procedure so that JMU can create a new one. The default value is SJMSFILE.
&JMU	stands for the JMU load module called in the procedure. The default value is $.JMU

BS2000 commands within the procedure

Immediately after the procedure header there are a number of /REMARK commands which contain information about the SJMSFILE to be created: the SJMSFILE file name defined by the CREATE-PROCEDURE-FILE statement, the date and time this statement was executed, the JMU version used and various characteristics of the SJMSFILE.

Then follow the commands and JMU statements required to run the procedure.

JMU statements within the procedure

The JMU statements DEFINE-JOB-STREAM, DEFINE-JOB-CLASS, GRANT-JOB-CLASS-ACCESS, SET-JOB-CLASS-DEFAULT, SET-POSIX-JOB-CLASS-DEFAULT and END are used to create the SJMSFILE.

The statements must be specified in a particular sequence within the procedure: First of all, the stream definitions should be specified in alphabetical order. A DEFINE-JOB-STREAM statement is required for each of these. Next, all job classes are defined, also in alphabetical order. Here, the statements GRANT-JOB-CLASS-ACCESS, SET-JOB-CLASS-DEFAULT and SET-JOB-CLASS-DEFAULT may be required in addition to the DEFINE-JOB-CLASS statement to define the user’s access rights for these classes. The JMU run is terminated by the END statement.

6.3.2.2 DEFINE-JOB-CLASS - Write job class definitions to SJMSFILE

This statement is used to write a new job class definition to the SJMSFILE or JMS database and define its characteristics.

Format

```
DEFINE-JOB-CLASS

 NAME = <name 1..8>

 ,STREAM = *DEFAULT-STREAM / <name 1..8>

 ,CLASS-LIMIT = <integer 0..4095>

 ,CLASS-WEIGHT = <integer 1..9>

 ,CLASS-OPTIMUM = 0 / <integer 0..4095>

 ,JOB-PRIORITY = *NO / *PARAMETERS(...)

     *PARAMETERS(...)

         |   DEFAULT = <integer 1..9>

         |   ,MAXIMUM = *NO / <integer 1..9>

 ,JOB-TYPE = *BATCH / *DIALOG

 ,TP-ALLOWED = *NO / *YES(...)

     *YES(...)

         |   CATEGORY = TP / <name 1..7>

 ,DIALOG-ALLOWED = *NO / *YES(...)

     *YES(...)

         |   CATEGORY = DIALOG / <name 1..7>

 ,BATCH-ALLOWED = *NO / *YES(...)

     *YES(...)

         |   CATEGORY = BATCH / <name 1..7>

 ,START-ATTRIBUTE = *BATCH / *DIALOG / *TP

 ,RUN-PRIORITY = *PARAMETERS(...)

     *PARAMETERS(...)

         |   DEFAULT = <integer 30..255>

         |   ,MAXIMUM = *NO / <integer 30..255>
```

```
,NO-CPU-LIMIT = *NO / *YES
,CPU-LIMIT = *PARAMETERS(...)
    *PARAMETERS(...)
        |   DEFAULT = *NO-LIMIT / <integer 1..32767>
        |   ,MAXIMUM = *NO / <integer 1..32767>
,SYSLST-LIMIT = *PARAMETERS(...)
    *PARAMETERS(...)
        |   DEFAULT = *NO-LIMIT / <integer 0..999999>
        |   ,MAXIMUM = *NO / *NO-LIMIT / <integer 0..999999>
,START = *NO / *PARAMETERS(...)
    *PARAMETERS(...)
        |   DEFAULT = *SOON / *WITHIN(...)
        |       *WITHIN(...)
        |           |   HOURS = 0 / <integer 0..23>
        |           |   ,MINUTES = 00 / <integer 0..59>
        |   ,ALLOWED = list-poss(7): *AT-STREAM-STARTUP / *AT / *EARLIEST / *SOON /
        |                   *LATEST / *WITHIN / *IMMEDIATELY
,REPEAT-JOB = *NO / *PARAMETERS(...)
    *PARAMETERS(...)
        |   DEFAULT = *NO / *AT-STREAM-STARTUP / *WEEKLY / *DAILY / *PERIOD(...)
        |       *PERIOD(...)
        |           |   HOURS = 0 / <integer 0..23>
        |           |   ,MINUTES = 00 / <integer 0..59>
        |   ,ALLOWED = list-poss(5): *NO / *AT-STREAM-STARTUP / *DAILY / *WEEKLY /
        |                   *PERIOD
,JOB-PARAMETER = *NO / <c-string 0..127>
```

Operands

NAME = <name 1..8>
Name of the new job class definition to be written to the SJMSFILE. The name may consist of between 1 and 8 alphanumeric characters. The first character must be a letter from the set A through Z or the character @ or #.

> ¡ The statement is rejected if a job class with the same name already exists.

STREAM = *DEFAULT-STREAM
The default stream defined in the DEFINE-JOB-STREAM statement.

> ¡ The default stream in the SHMSFILE must not be identical with that of the system.

STREAM = <name 1..8>
Name of the stream under which the job scheduler to which the job class is assigned runs. The name must not be $SYSJS. The name of the stream must already have been defined using the DEFINE-JOB-STREAM statement.

CLASS-LIMIT = <integer 0..4095>
Maximum number of jobs that shall be started in the class.

> ¡ It is not advisable to specify a value of 0 except in order to prevent jobs from being started following system startup. Otherwise, the job scheduler's performance could be impaired. The specified value is an absolute limit, but it may be exceeded by express jobs.

CLASS-WEIGHT = <integer 1..9>
Determines the start priority of the class relative to other classes whose jobs are waiting to be started.
1 is the lowest and 9 the highest weight.

CLASS-OPTIMUM =
Specifies the optimum number of jobs that should run in the job class in order to achieve a balanced job distribution within the system.
CLASS-OPTIMUM influences the sequence in which the class scheduler selects the job classes in order to start the jobs.

CLASS-OPTIMUM = 0 / <integer 0..4095>
The number of jobs. Values from 0 up to the value specified in the CLASS-LIMIT operand can be specified: 0 <= CLASS-OPTIMUM <= CLASS-LIMIT <= 4095.

JOB-PRIORITY =
Specifies the job scheduling priority for batch jobs; this determines the priority of a job relative to other jobs of the same class.

JOB-PRIORITY = *NO
This specification is required by the statement format; it has no significance, but must be given if JOB-TYPE=DIALOG was specified.

JOB-PRIORITY = *PARAMETERS(...)

DEFAULT = <integer 1..9>
Default priority for the job class.
1 is the highest and 9 the lowest priority.

If the user has not specified a priority for a job, DEFAULT=<integer...> is used.
If the user has specified a priority no higher than MAXIMUM=<integer...>, the priority specified by the user in `/ENTER-JOB` applies.
The DEFAULT operand in the DEFINE-JOB-CLASS statement must not specify a priority higher than that given in MAXIMUM, otherwise the statement is rejected with a syntax error.

MAXIMUM = NO / <integer 1..9>
Maximum permitted priority for the job class.
If MAXIMUM=NO is specified, the job is given the priority specified for DEFAULT, irrespective of the priority given by the user in `/ENTER-JOB` .

JOB-TYPE =
Defines the type of job class.

JOB-TYPE = *BATCH
Specifies that the job class is to be an interactive job class. This means that a job belonging to this class must not be initiated by `/SET-LOGON-PARAMETERS` in interactive mode.

JOB-TYPE = *DIALOG
Specifies that the job class is to be an interactive job class. A job belonging to this class must not be initiated by `/ENTER-JOB` .

> In the case of a POSIX default class (see the SET-POSIX-JOB-CLASS-DEFAULT statement) the JOB-TYPE is evaluated only when the user attempts to start normal batch or interactive jobs in this class. The specification has no effect on POSIX-FORK tasks.

TP-ALLOWED = *NO / *YES(...)
Specifies whether the task attribute TP is permitted in the job class.

TP-ALLOWED = *NO
Means that the task attribute TP is not allowed in the job class. Jobs in this class must not be started under this task attribute; switching to this task attribute by means of the TINF macro is forbidden, unless it is permitted by user catalog entry.

TP-ALLOWED = *YES(...)

CATEGORY = TP / <name 1..7>
A category name may be assigned to the task attribute TP.
It may be the default category name (TP) or a name freely defined by the user. The standard category name SYS is not allowed.
Up to twelve further category names may be defined in addition to the four predefined names.
If the database is updated, the default category name TP is used if the specified category is unknown to the system.
(See “Note for all 3 task attributes:”.)

DIALOG-ALLOWED = *NO / *YES(...)
Specifies whether the task attribute DIALOG is permitted in the job class.

DIALOG-ALLOWED = *NO
Means that the task attribute DIALOG is not permitted in the job class. Jobs in this class must not be started under this task attribute; switching to this task attribute by means of the TINF macro is forbidden, unless it is permitted by user catalog entry.

DIALOG-ALLOWED = *YES(...)

CATEGORY = DIALOG / <name 1..7>
A category name may be assigned to the task attribute DIALOG.
It may be the default category name (DIALOG) or a name freely defined by the user. The standard category name SYS is not allowed.
Up to twelve further category names may be defined in addition to the four predefined names.
If the database is updated, the default category name DIALOG is used if the specified category is unknown to the system.
(See “Note for all 3 task attributes:”.)

BATCH-ALLOWED = *NO / *YES(...)
Specifies whether the task attribute BATCH is permitted in the job class.

BATCH-ALLOWED = *NO
Means that the task attribute BATCH is not permitted in the job class. Jobs in this class must not be started under this task attribute; switching to this task attribute by means of the TINF macro is forbidden, unless it is permitted by user catalog entry.

BATCH-ALLOWED = *YES(...)

CATEGORY = BATCH / <name 1..7>
A category name may be assigned to the task attribute BATCH.
It may be the default category name (BATCH) or a name freely defined by the user. The standard category name SYS is not allowed.
Up to twelve further category names may be defined in addition to the four predefined names.
If the database is updated, the default category name BATCH is used if the specified category is unknown to the system.

> Note for all 3 task attributes:
> The dependencies on the value of the START-ATTR operand must be taken into consideration.
>
> A category name may not be assigned to two different task attributes, e.g. the following specification is ambiguous and therefore not allowed:
> BATCH-ALLOWED=*YES(CATEGORY=HUGO) and DIALOG-ALLOWED=*YES(CATEGORY=HUGO)

START-ATTRIBUTE = *BATCH / *DIALOG / *TP
Defines the task attribute for the job. At the same time the corresponding task attribute must be specified in the TP-, DIALOG- or BATCH-ALLOWED operand, e.g. START-ATTRIBUTE=TP and TP-ALLOWED=*YES(...).

RUN-PRIORITY =
Specifies the run priority with which a job is started.

RUN-PRIORITY = *PARAMETERS(...)

DEFAULT = <integer 30..255>
Default value for the job class.
DEFAULT must not give a priority higher than MAXIMUM.

MAXIMUM = *NO / <integer 30..255>
Specifies the maximum permitted priority for the job class.
Means that a job is given the priority specified by the user, providing it does not exceed the maximum permitted priority. If, however, the user catalog entry indicates that a higher priority is permitted than that specified in the MAXIMUM operand, the job may exceed the value of MAXIMUM.

MAXIMUM=*NO means that no maximum task priority is defined. A job is given the priority assigned by the user providing it does not exceed the value in the user catalog entry.

NO-CPU-LIMIT = *NO / *YES
Specifies whether jobs in this class may run without a time limit (NTL).
NO means that jobs in this class must not run without a time limit (see /SET-LOGON-PARAMETERS). If NO-CPU-LIMIT=*YES is specified in the user catalog entry for a user, that user can run jobs without a time limit even when NO-CPU-LIMIT=*NO applies to the job class.

CPU-LIMIT =
CPU time that a job in this class may utilize.

CPU-LIMIT = *PARAMETERS(...)

DEFAULT = *NO-LIMIT
The default setting in this job class is that the jobs run without a time limit. The value is only permitted when NO-CPU-LIMIT=*YES and MAXIMUM=32767.

DEFAULT = <integer 1..32767>
Default value for the job class.

MAXIMUM = <integer 1..32767>
This is the maximum CPU time that can be explicitly requested for a job of this job class.This value cannot be exceeded if a numerical value is specified for CPU-LIMIT when a job is created with ENTER-JOB, ENTER-PROCEDURE or SET-LOGON-
PARAMETERS. The maximum CPU time that can be used for an account number depends on the CPU entry in the user catalog.
The value specified by DEFAULT must not exceed the value specified for MAXIMUM. In the case of DEFAULT=*NO-LIMIT the highest possible value (32767) must be assigned to MAXIMUM and NO-CPU-LIMIT=*YES must be specified. If one of these conditions is violated, the statement is rejected due to a syntax error.

MAXIMUM = *NO
The job may utilize the CPU time specified in DEFAULT, irrespective of the CPU time the user requested.

SYSLST-LIMIT =
Defines the number of lines for a job when output takes place via SYSLST.

SYSLST-LIMIT = *PARAMETERS(...)

DEFAULT = *NO-LIMIT / <integer 0..999999>
Default number of lines for the job class. *NO-LIMIT means that the number of lines is not limited. The value specified for DEFAULT must not exceed that specified for MAXIMUM, otherwise the DEFINE-JOB-CLASS statement is rejected with a syntax error.

MAXIMUM =
Maximum number of lines permitted for the job class.

MAXIMUM = *NO
The job is assigned the permitted number of lines specified in DEFAULT, irrespective of the number actually requested by the user.

MAXIMUM = *NO-LIMIT
There is no limit on the number of lines for a job in this class. The number specified by the user always applies.

MAXIMUM = <integer 0..999999>
Means that the job is assigned the number of lines specified by the user, providing it does not exceed the value specified for MAXIMUM.

START =
Assigns appropriate start options to job start requests.

START = *NO
This is a formal entry with no significance, except that it is required if the operand JOB-TYPE=*DIALOG is specified.

START = *PARAMETERS(...)

DEFAULT =
This is the default value assumed if a user has not requested a specific start type in `/ENTER-JOB`. The value defined for DEFAULT need not be listed under ALLOWED (see below).

DEFAULT = *SOON
The job should be started as soon as possible. If several jobs have requested SOON, the job priority determines which starts first.

DEFAULT = *WITHIN(...)
The job should be started within the time specified in hours and minutes.

HOURS = 0 / <integer 0..23>
Is a value in the range 0 to 23 hours.

MINUTES = 00 / <integer 0..59>
Is a value in the range 0 to 59 minutes.

ALLOWED =
Defines those values which the user may specify for the job class concerned in the START operand of `/ENTER-JOB`.

ALLOWED = *IMMEDIATELY
A job in this job class may be started immediately, even if it delays other jobs having higher priority that were supposed to be started at this time.

ALLOWED = *SOON
Same meaning as DEFAULT=*SOON, see above.

ALLOWED = *AT
The job may be started on the specified date and precisely at the specified time (hour, minutes) if possible.

ALLOWED = *LATEST
A job may be started at the latest by the specified date and time.

ALLOWED = *EARLIEST
A job may be started at the earliest at the specified date and time.

ALLOWED = *WITHIN
A job may be started within the specified time.

ALLOWED = *AT-STREAM-STARTUP
A job may be started at the time the job scheduler is started.

> If an option is not specified under ALLOWED, it is not allowed unless the entry concerned is *SOON or *WITHIN, defined under DEFAULT as the default value.
> A further exception is ALLOWED=*IMMEDIATELY: If the user has specified START=*IMMEDIATELY in /SET-LOGON-PARAMETERS, /ENTER-JOB or /MODIFY-JOB, and START-IMMEDIATE=*YES is specified in the user catalog entry, the job will be started immediately, even if ALLOWED=*IMMEDIATELY is not specified in the user's job class.

REPEAT-JOB =
Controls the frequency of repeat jobs at specific time intervals. This is ignored if JOB-TYPE =*DIALOG is specified.

REPEAT-JOB = *NO
This is a formal entry with no significance, except that it is required if the operand JOB-TYPE=*DIALOG is specified.

REPEAT-JOB = *PARAMETERS(...)

DEFAULT =
This is the default value assumed if the user has made no entry for the frequency of job repetition in /ENTER-JOB or /SET-LOGON-PARAMETERS, i.e. has omitted the REPEAT-JOB or REPEAT operand, or has specified REPEAT-JOB=STD or REPEAT=STD.

DEFAULT = *NO
Means that the job is not repeated.

DEFAULT = *AT-STREAM-STARTUP
Jobs belonging to this class are run again following each start of the job scheduler, provided the user has requested this in /SET-LOGON-PARAMETERS or /ENTER-JOB.

DEFAULT = *WEEKLY
Jobs in this class are started weekly. The exact start time depends on the START operand value in /ENTER-JOB.

DEFAULT = *DAILY
Jobs in this class are started daily. The exact start time depends on the START operand value in /ENTER-JOB.

DEFAULT = *PERIOD(...)
Jobs are repeated each time the specified time interval has elapsed.

HOURS = 0 / <integer 0..23>
The time interval in hours may have a value between 0 and 23.

MINUTES = 00 / <integer 0..59>
A value between 0 and 59 minutes may be given.
The total time interval must be greater than 0. The exact start time depends on the START operand value in /ENTER-JOB.

ALLOWED =
Specifies the values the user may give in the REPEAT-JOB or REPEAT operand of /ENTER-JOB or /SET-LOGON-PARAMETERS.

ALLOWED = *NO
Repetition of jobs in this class is not possible, unless a value other than NO has been specified for DEFAULT.

ALLOWED =*AT-STREAM-STARTUP
Jobs may be repeated if required each time the job scheduler is started.

ALLOWED = *DAILY
Jobs in this class may be repeated daily. The exact starting time depends on the START operand value in /ENTER-JOB.

ALLOWED = *WEEKLY
Jobs in this class may be repeated weekly. The exact starting time depends on the START operand value in /ENTER-JOB.

ALLOWED = *PERIOD
Jobs can be repeated each time the specified time interval has elapsed.

JOB-PARAMETER =
Additional class attributes, evaluated by system exit 32.

JOB-PARAMETER = *NO
No additional class attributes are defined.

JOB-PARAMETER = <c-string 1..127>
This operand defines additional class attributes in free syntax. Here, system-specific information can be stored in each job class definition.

This operand is not evaluated by the system or by the predefined job scheduler. If it is to be evaluated, systems support must:

1. create an exit routine to compare what the user specifies in the JOB-PARAMETER operand of /SET-LOGON-PARAMETERS, /ENTER-JOB or /MODIFY-JOB with what was specified in JOB-PARAMETER and to confirm its validity or
2. define the scheduling algorithm of the scheduler which is responsible for the job class accordingly.

The exit routine is called during the processing of these commands entered by the user.

Notes

- The CLASS-LIMIT and CLASS-WEIGHT operands are evaluated by the operating system's class scheduler, which is independent of the job schedulers, in order to control the job-related portion of the system load (see the "Introduction to System Administration" [5 (Related publications)]).
- The significance of the JOB-PRIORITY, START and JOB-PARAMETER operands depends on the job scheduling algorithm used by the stream, as defined in the STREAM operand.
- The number of job classes is unlimited.

6.3.2.3 DEFINE-JOB-STREAM - Write stream definitions to SJMSFILE

This statement writes a new stream definition to the SJMSFILE file or the JMS database.

Format

DEFINE-JOB-STREAM

```
NAME = <name 1..8>
,FILE = <filename 1..54> / *LIBRARY-ELEMENT(...)
      *LIBRARY-ELEMENT(...)
            |     LIBRARY = <filename 1..41>
            |     ,ELEMENT = <name 1..8>
,RUN-PRIORITY = 65 / <integer 30..255>
,DEFAULT = *NO / *YES
,START = *AT-LOAD / *BY-OPERATOR / *AT(...) / *EARLIEST(...)
      *AT(...)
            |     TIME = <time 1..8>
      *EARLIEST(...)
            |     TIME = <time 1..8>
,STOP = *AT-SHUTDOWN / *BY-OPERATOR / *AT(...) / *AFTER(...)
      *AT(...)
            |     TIME = 00:00 / <time 1..8>
      *AFTER(...)
            |     HOURS = 00 / <integer 0..23>
            |     ,MINUTES = 00 / <integer 0..59>
,STREAM-PARAMETER = *NO / <c-string 1..127>
```

Operands

NAME = <name 1..8>
Name of the stream definition to be written to SJMSFILE. A string of between 1 and 8 alphanumeric characters may be specified, starting with the character A-Z, @ or #.

FILE = <filename 1..54>
Name of the ENTER file containing the job that is initiated during the stream start and that activates the job scheduler.
The job scheduler can only process batch jobs.

FILE = *LIBRARY-ELEMENT(...)

LIBRARY = <filename 1..41>
File name of the library.

ELEMENT = <name 1..8>
The library element containing the ENTER file named above.

RUN-PRIORITY = 65 / <integer 30..255>
Specifies the starting priority to be assigned to the stream task under which the job scheduler runs.

DEFAULT = *NO / *YES
Specifies whether the stream involved is to be the default stream for the system.
YES means the stream is to be the default stream for those job classes that have specified that they wish to use the default stream.

START =
Specifies when the stream is to be started.

START = *AT-LOAD
The stream is to be started automatically when the system is loaded.

START = *BY-OPERATOR
The stream must be started by the operator or system administration using `/START-JOB-STREAM`.

START = *AT(...)
The stream is to be started automatically during each session at the specified time. If a session is started after the specified time, the stream can be started only within the next 30 minutes after the time specified. A start at a later point within the current session is not possible.

> If the job stream is to be entered in the JMS database, the following applies: The stream is started when the START time is not exceeded by more than 5 minutes and the STOP time has not yet been reached.

START = *EARLIEST(...)
The stream is to be started at the earliest at the specified time during each session. If the system is not active at the time specified, the start is retried until a specified STOP time or until 24:00 hours.

> If the job stream is to be entered in the JMS database, the following applies: The stream is started when the START time is exceeded and the STOP time has not yet been reached.

TIME = 00:00 / <time 1..8>
Time of day in the format hh:mm; i.e. hours and minutes only, seconds are ignored.

STOP =
Specifies when the stream is to be stopped.

STOP = *AT-SHUTDOWN
The stream is to be stopped when the system is shut down.

STOP = *BY-OPERATOR
The stream must be stopped by the operator or system administration by means of `/STOP-JOB-STREAM`.

STOP = *AT(...)
The stream is to be stopped automatically at the specified time (hh:mm see START=*AT...).

TIME = 00:00 / <time 1..8>
Time of day in the format hh:mm; i.e. hours and minutes only, seconds are ignored.

STOP = *AFTER(...)
The stream is to be stopped after the specified time has elapsed.

HOURS = 00 / <integer 0..23>
HOURS= between 0 and 23 hours may be specified.

MINUTES = 00 / <integer 0..59>
A value between 0 and 59 minutes may be given.

> If for any reason the system is shut down during the time between system start and the time specified for stopping the stream, and the system is then restarted before the time the stream was to be stopped, the stream is also restarted automatically.

STREAM-PARAMETER =
This operand defines special scheduling parameters for the job scheduler in free syntax. The contents of this operand are not evaluated by the system.
However, the job scheduler involved must understand both the syntax and the meaning of STREAM-PARAMETER in order to be able to accept the scheduling parameters defined there.
The job scheduler gets this information via the job scheduler interface. The job scheduler interface provides, via a TU interface, functions required by the job scheduler to carry out its tasks (for further details, see “Introduction to System Administration” [5 (Related publications)]).

STREAM-PARAMETER = <c-string 1..127>
Sequence of special parameters for the job scheduler.
When the system is started up the information contained in the parameters is transferred to internal tables, where it can be accessed by the job scheduler.

The default job scheduler knows the following parameters:

```
S-PAR =  'JOB-PRIORITY = YES / NO
          ,CPU-TIME = NO / YES
          ,WAIT-TIME = NO / YES
          ,JOB-QUOTA = 1 / <integer 1..255>
          ,LOGGING = YES / NO
          ,CATID-LIST = (catid1,...)
          ,CAT-TIME = min
          ,MSG-OUTPUT = YES / STD'
```

STREAM-PARAMETER = *NO
Means that no special parameters are defined for the job scheduler.

Notes

- This statement is rejected if a stream with the specified name is already contained in the SJMSFILE.
- Only one default stream may exist in the system. An attempt to define more than one default stream will be rejected.
- No more than 16 streams may be defined.

6.3.2.4 DELETE-JOB-CLASS - Delete class definitions

This statement can be used to delete an existing class definition from the SJMSFILE or the JMS database.

If the definition in the file is deleted, any jobs of the class in question that are in the job pool when the next system session starts are lost. If the definition is to be deleted in the system, its status is flagged as “IN-DELETE”. The class definition is not removed from the database until no remaining jobs are assigned to the class.
No new jobs are accepted for job classes flagged as “IN-DELETE”.
Deletion of a default class is not permitted. If a default class is to be deleted, all access rights to it must first be rescinded.

Format

DELETE-JOB-CLASS
NAME = <name 1..8>

Operands

NAME = <name 1..8>
Name, consisting of from 1 to 8 characters, of an existing class definition that is to be deleted.

6.3.2.5 DELETE-JOB-STREAM - Delete stream definitions

This statement deletes a stream definition from the SJMSFILE file and the JMS database.

Format

DELETE-JOB-STREAM
NAME = <name 1..8>

Operands

NAME = <name 1..8>
Name of the job stream to be deleted.

> If job classes are assigned to the stream, the delete request is rejected.
> If the stream is active and is to be deleted from the JMS database, the delete request is rejected.

6.3.2.6 END - Terminate statement input

The END statement terminates input of statements to the JMU routine.

Format

END

This statement has no operands.

6.3.2.7 GRANT-JOB-CLASS-ACCESS - Control access by user IDs to job class

This statement is used to control access by user IDs to a job class.

If a job class is defined as the default class for a user (by means of the SET-JOB-CLASS-DEFAULT statement), it is not necessary to define access once again using the GRANT-JOB-CLASS-ACCESS statement.

If a new user ID is entered in the user catalog during a session, it is immediately granted access to all public classes.

Format

GRANT-JOB-CLASS-ACCESS
NAME = <name 1..8> ,ACTION = *ADD / *REMOVE ,USER = *ALL / list-poss(255): <name 1..8>

Operands

NAME = <name 1..8>
Name of a job class to which a user ID is to receive or be denied access.

ACTION =
Permits or denies access to a job class.

ACTION = *ADD
The user ID specified under USER= may access the job class.

ACTION = *REMOVE
The user ID specified under USER= is denied access to the job class specified for NAME =.

USER =

Controls access to a job class.

USER = *ALL
Access to the job class specified in NAME is to be defined for all user IDs.

USER = list-poss(255)
Access is granted for the user IDs given in this list.

USER = <name 1..8>
Access is granted to this one user ID with a name consisting of 1 to 8 characters. No check is run to ascertain whether the specified user ID is entered in the user catalog unless a modification is made in the current session. All IDs are entered in the SJMSFILE.

6.3.2.8 MODIFY-JOB-CLASS - Modify job class definitions

This statement is used to modify an existing job class definition in the SJMSFILE or the JMS database.

Format

```
MODIFY-JOB-CLASS

 NAME = <name 1..8>

 ,STREAM = *UNCHANGED / *DEFAULT-STREAM / <name 1..8>

 ,CLASS-LIMIT = *UNCHANGED / <integer 0..4095>

 ,CLASS-WEIGHT = *UNCHANGED / <integer 1..9>

 ,CLASS-OPTIMUM = *UNCHANGED / <integer 0..4095>

 ,JOB-PRIORITY = *UNCHANGED / *PARAMETERS(...)

     *PARAMETERS(...)

         |    DEFAULT = <integer 1..9>

         |    ,MAXIMUM = *UNCHANGED / *NO / <integer 1..9>

 ,JOB-TYPE = *UNCHANGED / *BATCH / *DIALOG

 ,TP-ALLOWED = *UNCHANGED / *NO / *YES(...)

     *YES(...)

         |    CATEGORY = TP / <name 1..7>

 ,DIALOG-ALLOWED = *UNCHANGED / *NO / *YES(...)

     *YES(...)

         |    CATEGORY = DIALOG / <name 1..7>

 ,BATCH-ALLOWED = *UNCHANGED / *NO / *YES(...)

     *YES(...)

         |    CATEGORY = BATCH / <name 1..7>

 ,START-ATTRIBUTE = *UNCHANGED / *BATCH / *DIALOG / *TP

 ,RUN-PRIORITY = *UNCHANGED / *PARAMETERS(...)

     *PARAMETERS(...)

         |    DEFAULT = <integer 30..255>

         |    ,MAXIMUM = *UNCHANGED / *NO / <integer 30..255>

 ,NO-CPU-LIMIT = *UNCHANGED / *NO / *YES
```

```
,CPU-LIMIT = *UNCHANGED / *PARAMETERS(...)
    *PARAMETERS(...)
        |   DEFAULT = <integer 1..32767>
        |   ,MAXIMUM = *UNCHANGED / *NO / <integer 1..32767>
,SYSLST-LIMIT = *UNCHANGED / *PARAMETERS(...)
    *PARAMETERS(...)
        |   DEFAULT = *NO-LIMIT / <integer 0..999999>
        |   ,MAXIMUM = *UNCHANGED / *NO / *NO-LIMIT / <integer 0..999999>
,START = *UNCHANGED / *PARAMETERS(...)
    *PARAMETERS(...)
        |   DEFAULT = *SOON / *WITHIN(...)
        |       *WITHIN(...)
        |           |   HOURS = 0 / <integer 0..23>
        |           |   ,MINUTES = 00 / <integer 0..59>
        |   ,ALLOWED = list-poss(7): *IMMEDIATELY / *AT / *EARLIEST / *LATEST /
        |                   *AT-STREAM-STARTUP / *WITHIN / *SOON
,REPEAT-JOB = *UNCHANGED / *PARAMETERS(...)
    *PARAMETERS(...)
        |   DEFAULT = *NO / *AT-STREAM-STARTUP / *WEEKLY / *DAILY / *PERIOD(...)
        |       *PERIOD(...)
        |           |   HOURS = 0 / <integer 0..23>
        |           |   ,MINUTES = 00 / <integer 0..59>
        |   ,ALLOWED = list-poss(5): *NO / *AT-STREAM-STARTUP / *DAILY / *WEEKLY /
        |                   *PERIOD
,JOB-PARAMETER = *UNCHANGED / *NO / <c-string 0..127>
```

For a description of the operands, see the DEFINE-JOB-CLASS statement.

> Database updates are ignored for the operands STREAM, JOB-TYPE and START-ATTRIBUTE. The JOB-TYPE operand must not be modified for default job classes.

6.3.2.9 MODIFY-JOB-STREAM - Modify stream definitions

This statement enables an existing stream definition to be modified.

Format

MODIFY-JOB-STREAM

```
NAME = <name 1..8>
,FILE = *UNCHANGED / <filename 1..54> / *LIBRARY-ELEMENT(...)
        *LIBRARY-ELEMENT(...)
                |       LIBRARY = <filename 1..41>
                |       ,ELEMENT = <name 1..8>
,RUN-PRIORITY = *UNCHANGED / <integer 30..255>
,DEFAULT = *UNCHANGED / *NO / *YES
,START = *UNCHANGED / *AT-LOAD / *BY-OPERATOR / *AT(...) / *EARLIEST(...)
        *AT(...)
                |       TIME = 00:00 / <time 1..8>
        *EARLIEST(...)
                |       TIME = 00:00 / <time 1..8>
,STOP = *UNCHANGED / *AT-SHUTDOWN / *BY-OPERATOR / *AT(...) / *AFTER(...)
        *AT(...)
                |       TIME = 00:00 / <time 1..8>
        *AFTER(...)
                |       HOURS = 00 / <integer 0..23>
                |       ,MINUTES = 00 / <integer 0..59>
,STREAM-PARAMETER = *UNCHANGED / *NO / <c-string 1..127>
```

For a description of the operands, see the DEFINE-JOB-STREAM statement.

When changing the DEFAULT= operand you should bear in mind that there must never be more than one default stream in the system. Any attempt to define more than one will be rejected.

If the START or STOP operand is changed directly in the JMS database, the job stream is started or stopped in accordance with the new operand values.

6.3.2.10 REMOVE-USER - Prohibit access to private job classes

The REMOVE-USER statement is used to deny specified user IDs access to all private job classes. The statement is an extension of the system command of the same name which is used to delete user entries in the user catalog. However, the statement cannot be used to prevent special users accessing public job classes or system default classes.

Format

REMOVE-USER
USER-IDENTIFICATION = *ALL / list-poss(8): <name 1..8>

Operands

USER-IDENTIFICATION =
Specifies the user IDs to be removed.

USER-IDENTIFICATION = *ALL
This causes all access lists to be deleted. All private job classes are thus prohibited. Any ID-specific default job classes are reset to the system default settings.

USER-IDENTIFICATION = list-poss(8): <name 1..8>
The specified user ID(s) is (are) to be prevented from accessing all private job classes. Any default job classes specific to these IDs are reset. Up to 8 user IDs can be specified.

6.3.2.11 SET-JOB-CLASS-DEFAULT - Specify default classes for users

This statement is used to specify default classes for users. At the same time, the users are granted access.

If no public default class has been specified for a job type, $SYSJC is the system default class. Since only privileged users have access to $SYSJC, JMU issues a warning message.

When a new user is entered in the user catalog during a session, he or she receives immediate access to all system default classes.

Format

SET-JOB-CLASS-DEFAULT
NAME = <name 1..8> ,ACTION = *ADD / *REMOVE ,USER = *ALL / list-poss(255): <name 1..8>

The meaning of the operands is the same as for the GRANT-JOB-CLASS-ACCESS statement.

> Changing the default class always consists of the two statements:
>
> - SET-JOB-CLASS-DEFAULT default-class.old,*REMOVE,*ALL
> - SET-JOB-CLASS-DEFAULT default-class.new,*ADD,*ALL.
>
> It is not advisable to change the default class during the current session, because $SYSJC is entered as the default class between the two statements.

6.3.2.12 SET-MODIFICATION-MODE - Set modification mode

This statement can be used to set a modification mode in the JMU utility routine that permits the following modifications to be made to JMS files in the current session:

- modify rights of access to job classes
- assign default job classes to new users
- create, delete and modify job class definitions
- create, delete and modify job stream definitions

If desired, these modifications can be implemented with immediate effect.

> The SET-MODIFICATION-MODE statement is permitted only when the caller possesses the TSOS privilege.

Format

SET-MODIFICATION-MODE
SCOPE = *FILE / *SYSTEM / *ALL

Operands

SCOPE =
Specifies whether the subsequent modifications are to be implemented in the file, the database or both.

SCOPE = *FILE
The modifications are to be implemented in the file only, i.e. they will be effective only as of the next startup.

SCOPE = *SYSTEM
The modifications are to be implemented in the database only, i.e. they are effective only for the current session (for test purposes).

SCOPE = *ALL
The modifications are to be implemented in the file and in the database.

Notes

- If SCOPE=*SYSTEM / *ALL is specified, the STREAM, JOB-TYPE and START-ATTRIBUTE operands of the MODIFY-JOB-CLASS statement are ignored.
- Only the category names already defined can be used for assigning categories.
- If job classes are deleted in the current system, a class containing jobs that are still active is flagged JOB-CLASS-IN-DELETE. Jobs already accepted can be completed, but no new jobs are accepted.
- SHOW-JOB-CLASS as a JMU statement outputs only the contents of the SJMSFILE. If SCOPE=*SYSTEM, the SHOW-JOB-CLASS statement is not executed.
- If SCOPE=*ALL is specified, a modification is made only if it is possible both in the SJMSFILE and in the database.

6.3.2.13 SET-POSIX-JOB-CLASS-DEFAULT - Specify POSIX default classes for users

This statement is used to specify system-wide and/or user-specific POSIX default classes for POSIX-FORK tasks.

If neither a system-wide POSIX default class nor one which applies for the user concerned has been defined, the user's POSIX-FORK tasks are started in the default class for batch or interactive jobs which applies for the user.

Format

SET-POSIX-JOB-CLASS-DEFAULT
NAME = <name 1..8> ,ACTION = *ADD / *REMOVE ,USER = *ALL / list-poss(255): <name 1..8>

The meaning of the operands is the same as for the GRANT-JOB-CLASS-ACCESS statement.

Changing the POSIX default class always consists of the two statements:

- SET-POSIX-JOB-CLASS-DEFAULT default-class.old,*REMOVE,*ALL
- SET-POSIX-JOB-CLASS-DEFAULT default-class.new,*ADD,*ALL.

If no POSIX default class has been assigned yet, the first statement is omitted.

Unlike with SET-JOB-CLASS-DEFAULT, when a job class is assigned as the POSIX default class, the selected users are not automatically guaranteed access to this job class. When required, this must be defined explicitly by the administrator using GRANT-JOB-CLASS-ACCESS. This does not automatically permit normal interactive and batch jobs to be started in this job class. POSIX-FORK tasks can use the job class, however.

A POSIX default class defined using USER=*ALL initially applies throughout the system for all user IDs. If different POSIX default classes are to apply for certain user IDs, this can be defined with further SET-POSIX-JOB-CLASS-DEFAULT statements, e.g. with the operands NAME=other_class,ACTION=*ADD, USER=list_of_users.

Default classes already defined with SET-JOB-CLASS-DEFAULT can also be defined as POSIX default classes.

A POSIX default class can belong to the category interactive or batch. It is relevant for all POSIX-FORK tasks of the interactive or batch job type.

6.3.2.14 SHOW-JOB-CLASS - List contents of class definitions or names of classes

This statement enables the contents of class definitions or the names of classes to be listed. The listing of a class definition includes the names of all users having access to that class.

Format

SHOW-JOB-CLASS
NAME = *ALL / *ALL-NAMES / list-poss(255): <name 1..8> ,OUTPUT = *SYSOUT / *SYSLST

Operands

NAME =
Name of the class to be listed.

NAME = *ALL
All class definitions are to be listed.

NAME = *ALL-NAMES
All names of classes are to be listed (without the contents of the class definitions).

NAME = <list-poss(255): <name 1..8>
All class definitions whose names are specified in this list are to be listed. A maximum of 255 names may be given.

OUTPUT =
Defines the output destination.

OUTPUT = *SYSOUT
Output is to be to SYSOUT.

OUTPUT = *SYSLST
Output is to be to SYSLST.

Example

```
REQUESTED DETAILS OF JOB CLASS: JCBATCHF
NAME.........:JCBATCHF
STREAM........:JSSTD
CLASS LIMIT...:255
CLASS OPTIMUM.:0
WEIGHT........:3
JOB PRIORITY..:DEFAULT=3            MAXIMUM= 1
JOB ATTRIBUTES:JOBTYPE=BATCH        ST-ATTR= BATCH
TP ALLOWED....:TP
DIALOG ALLOWED:NO
BATCH ALLOWED.:BATCH
RUN PRIORITY..:DEFAULT=180          MAXIMUM= 30
NO CPU LIMIT..:YES
CPU LIMIT.....:DEFAULT=32767        MAXIMUM= 32767
SYSLST LIMIT..:DEFAULT=NO-LIMIT     MAXIMUM= NO-LIMIT
SYSOPT LIMIT..:DEFAULT=NO-LIMIT     MAXIMUM= NO-LIMIT
START.........:DEFAULT=SOON         ALLOWED= SOON EARLY AT LATE IN IMM STUP
REPEAT JOB....:DEFAULT=NO           ALLOWED= NO STUP DAILY WEEKLY PERIOD
JOB PARAMETER.:UNDEFINED
JCBATCHF IS AVAILABLE TO:
ALL USERS
JCBATCHF IS A DEFAULT FOR:
NO USERS
REQUESTED DETAILS OF JOB CLASS: JCBSTD
NAME.........:JCBSTD
STREAM........:JSSTD1
CLASS LIMIT...:50
CLASS OPTIMUM.:0
WEIGHT........:8
JOB PRIORITY..:DEFAULT=9            MAXIMUM= 1
JOB ATTRIBUTES:JOBTYPE=BATCH        ST-ATTR= BATCH
TP ALLOWED....:TP
DIALOG ALLOWED:NO
BATCH ALLOWED.:BATCH
RUN PRIORITY..:DEFAULT=220          MAXIMUM= 180
NO CPU LIMIT..:YES
CPU LIMIT.....:DEFAULT=32000        MAXIMUM= 32767
SYSLST LIMIT..:DEFAULT=NO-LIMIT     MAXIMUM= NO-LIMIT
SYSOPT LIMIT..:DEFAULT=NO-LIMIT     MAXIMUM= NO-LIMIT
START.........:DEFAULT=SOON         ALLOWED= SOON EARLY AT LATE IN IMM
REPEAT JOB....:DEFAULT=NO           ALLOWED= NO STUP DAILY WEEKLY PERIOD
JOB PARAMETER.:UNDEFINED
JCBSTD   IS AVAILABLE TO:
ALL USERS
JCBSTD   IS A SYSTEM DEFAULT
```

6.3.2.15 SHOW-JOB-STREAM - List contents of stream definitions or names of streams

This statement is used to list the contents of stream definitions or the stream names themselves.

Format

SHOW-JOB-STREAM
NAME = *ALL / *ALL-NAMES / list-poss(255): <name 1..8> ,OUTPUT = *SYSOUT / *SYSLST

Operands

NAME =
Name of the job stream to be listed.

NAME = *ALL
All stream definitions are to be listed.

NAME = *ALL-NAMES
All stream names are to be listed.

NAME = <list-poss(255): <name 1..8>
All stream definitions whose names are specified in this list are to be listed. A maximum of 255 names may be given.

OUTPUT =
Defines the output destination.

OUTPUT = *SYSOUT
The stream definition(s) or stream names are to be output to SYSOUT.

OUTPUT = *SYSLST
Output is to be to SYSLST.

Example

```
REQUESTED DETAILS OF JOB STREAM: JSSTD
NAME.........:JSSTD
FILE.........:SYSENT.JOBSCHED.150
RUN PRIORITY..:125
DEFAULT.......:NO
START.........:AT-LOAD
STOP..........:AT-SHUTDOWN
STREAMPARAM...:JOB-PRIORITY=Y,CPU-TIME=Y,WAIT-TIME=Y,JOB-QUOTA=50,LOGGING=NO
REQUESTED DETAILS OF JOB STREAM: JSSTD1
NAME.........:JSSTD1
FILE.........:SYSENT.JOBSCHED.150
RUN PRIORITY..:130
DEFAULT.......:YES
START.........:AT-LOAD
STOP..........:AT-SHUTDOWN
STREAMPARAM...:JOB-PRIORITY=Y,CPU-TIME=Y,WAIT-TIME=Y,JOB-QUOTA=30,LOGGING=NO
REQUESTED DETAILS OF JOB STREAM: JSSTD2
NAME.........:JSSTD2
FILE.........:SYSENT.JOBSCHED.150
RUN PRIORITY..:150
DEFAULT.......:NO
START.........:AT-LOAD
STOP..........:AT-SHUTDOWN
STREAMPARAM...:JOB-PRIO=Y,CPU-TIME=Y,WAIT-TIME=N,JOB-QUOTA=20,LOGGING=NO
REQUESTED DETAILS OF JOB STREAM: JSTSOS
NAME.........:JSTSOS
FILE.........:SYSENT.JOBSCHED.150
RUN PRIORITY..:120
DEFAULT.......:NO
START.........:AT-LOAD
STOP..........:AT-SHUTDOWN
STREAMPARAM...:JOB-PRIORITY=Y,CPU-TIME=Y,WAIT-TIME=Y,JOB-QUOTA=50,LOGGING=NO
```

7 LMSCONV Generation and management of libraries

Version: LMSCONV (SDF format) V3.6A

LMSCONV (Library Maintenance System Converter) is a routine for managing libraries and the members they contain.

LMSCONV performs the following functions:

- create libraries
- add files as library members
- Outputs members in files
- copy members to another library
- list members
- delete members
- correct members
- Outputting library directories

LMSCONV supports the processing of files > 32 GB (LARGE OBJECTS).

LMSCONV, a variant of the software product LMS with functional restrictions, is provided with both an ISP interface and an SDF interface. LMSCONV (ISP format) remains functionally unaltered as of BS2000/OSD-BC V2.0. This chapter describes the functionality of LMSCONV (SDF format).

LMSCONV (SDF format) is derived from the LMS (SDF format) software product. The functional scope is a subset of LMS (SDF format). A list of restrictions compared to LMS is given on "Comparison between LMSCONV and LMS"
.

Notes

- LMSCONV indicates all attributes for libraries, types and members, i.e. including attributes that can only be modified using LMS.
- For the processing of libraries under LMSCONV, all attributes, i.e. not only those that can be influenced directly under LMSCONV, are important. Library processing may therefore only be possible in restricted form (e.g. no read right for a member).
- Members of a defined use type can be processed with LMSCONV.
- Library lists and type control can be used.
- When installing LMSCONV, it is possible to specify the name which individual product files should receive and the ID under which they are to be stored. The complete path name of all product files of LMSCONV can be determined using /SHOW-INSTALLATION-PATH.

Overview of LMSCONV statements

Statement	Remarks
ADD-ELEMENT	Adds files as members
CLOSE-LIBRARY	Closes libraries
COPY-ELEMENT	Copies members
COPY-LIBRARY	Copies a library
DELETE-ELEMENT	Deletes members
END	Terminates the LMSCONV run
EXTRACT-ELEMENT	Outputs members in files
MODIFY-DEFAULTS	Modifies the global default values
MODIFY-ELEMENT	Modifies members
MODIFY-ELEMENT-ATTRIBUTES	Modifies member attributes
MODIFY-LOGGING-PARAMETERS	Modifies the logging scope
OPEN-LIBRARY	Opens a global library
SHOW-DEFAULTS	Shows the global default values
SHOW-ELEMENT	Shows the member contents
SHOW-ELEMENT-ATTRIBUTES	Shows the member attributes
SHOW-LIBRARY-ATTRIBUTES	Shows the library attributes
SHOW-LIBRARY-STATUS	Shows the library states
SHOW-LOGGING-PARAMETERS	Shows the values for the logging scope
SHOW-TYPE-ATTRIBUTES	Shows the type attributes
SHOW-USER-EXITS	Shows the LMSCONV version
WRITE-COMMENT	Writes a comment in the log

LMSCONV input and output stream

The following figure shows the input and output options for LMSCONV:

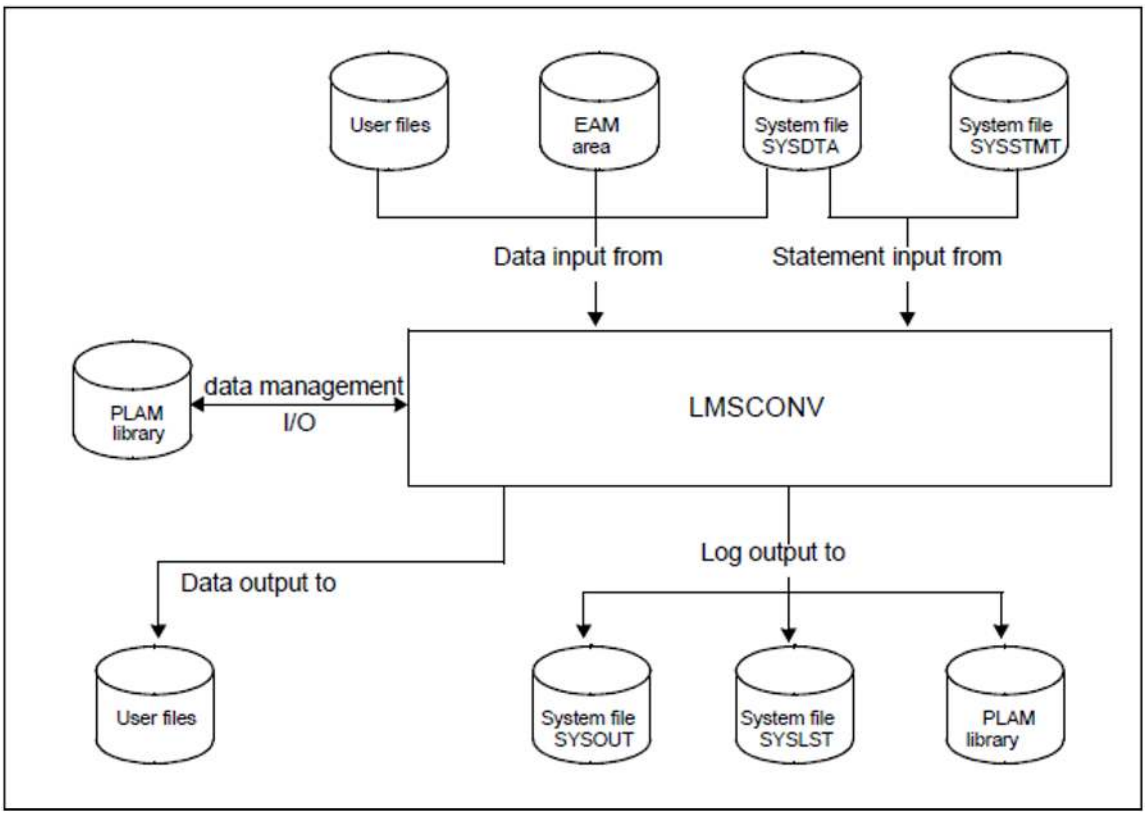


Figure 2: I/O of LMSCONV

LMSCONV reads all statement inputs via the dialog interface SDF. For detailed information, please refer to the “SDF Dialog Interface” manual [20 (Related publications)].

Example of an LMSCONV run

```
(IN)     start-lmsconv
(OUT)    %  LMC0310 LMSCONV VERSION '<version>' STARTED                    1.
(IN)     modify-logging-parameters logging=*maximum                       2.
(IN)     open-library library=ueb.bib,mode=update                         3.
(OUT)    LIBRARY IS CLEARED AND PREPARED                                  4.
(IN)     add-element from-file=test, to-elem=(elem=test.ueb,type=s)       5.
(OUT)    INPUT  FILE
(NL)     OUTPUT LIBRARY= :1OSN:$USER.UEB.BIB
(NL)              ADD :1OSN:$USER.TEST AS (S)TEST.UEB/@(0001)/<date>
(IN)     show-element-attributes                                          6.
(OUT)    INPUT  LIBRARY= :1OSN:$USER.UEB.BIB
(NL)     TYP NAME     VER (VAR#) DATE
(NL)     (S) TEST.UEB @   (0001) <date>
(NL)           1 (S)-ELEMENT(S) IN THIS TABLE OF CONTENTS                 7.
(IN)     show-element (element=test.ueb, type=s)                          8.
(OUT)    INPUT  LIBRARY= :1OSN:$USER.UEB.BIB
(NL)     INPUT  ELEMENT= (S)TEST.UEB/@(0001)/<date>
(NL)     TESTFILE, CAN BE DELETED AT ANY TIME !!                          9.
(NL)     NUMBER OF PROCESSED RECORDS IS       1
(IN)     end                                                             10.
(OUT)    %  LMC0311 LMSCONV VERSION '<version>' TERMINATED NORMALLY
```

1. LMSCONV is called.
2. In addition to error messages, success messages are also logged.
3. The new global library (i.e. assigned by the OPEN-LIBRARY statement) UEB.BIB is created and assigned as an I/O library.
4. The UEB.BIB library has been created.
5. The TEST file is added to the library as member TEST.UEB of type S.
6. The contents of the library UEB.BIB are to be listed.
7. Contents entry of the UEB.BIB library.
8. The TEST.UEB member is to be listed.
9. Contents of the TEST.UEB member.
10. LMSCONV is terminated.

LMSCONV in interactive and batch modes

LMSCONV runs in interactive mode and in batch mode.

Interactive mode

Since members can also be selected using wildcards, it is not possible initially to determine which member is currently being processed. For statements which delete or overwrite member data, it is thus possible to proceed “step-by-step”.

For each member, the user is prompted to process or skip the member, or abort the function.

The query mechanism is activated with the operand DIALOG-CONTROL=*YES.

The default value of DIALOG-CONTROL can be modified by the MODIFY-DEFAULTS statement.

The inquiry mechanism can be restricted to error cases by DIALOG-CONTROL = *ERROR. In the following non-recoverable error cases, the inquiry mechanism is activated in interactive mode even if DIALOG-CONTROL = *NO:

- when a member cannot be accessed, e.g. because it is locked by another user (temporary exclusive use of a member for modification, for example)
- when a library cannot be accessed, e.g. because the current access rights do not permit this (temporary exclusive use of a library).

It is possible to make LMSCONV behave as if it is running in batch mode, which, among others, totally deactivates the inquiry mechanism, by setting RUN-MODE=*BATCH in MODIFY-DEFAULTS statement.

Batch mode

If a library, a member or a type is locked, the user can set the number of new attempts and the time between two attempts using the statement MODIFY-DEFAULTS ..., NEXT-ATTEMPT. By default, no new attempts are made.

7.1 Libraries

LMSCONV processes PLAM libraries. PLAM libraries are PAM files in BS2000 that are processed with the library access method PLAM (Program Library Access Method).

A PLAM library contains members and a table of contents or directory of these stored members. PLAM libraries are used for the storage of source programs, macros, object modules, phases (load modules), lists, procedures, text etc.

PLAM libraries are also referred to simply as “libraries”.

7.1.1 Logical structure of a library

A library is a file with a substructure. It contains members and a directory.

A member (also referred to as “element” in examples and messages) is a logically related set of data, e.g. a procedure, an object module or a source program. Each member of a library can be referenced individually.

Storing a number of files as members in a library decreases the burden on the file name catalog since each library has only one catalog entry. Storage space is saved because the members are always stored in compressed form in the library. Furthermore, elements may also be stored as delta members, but you need LMS to do so.

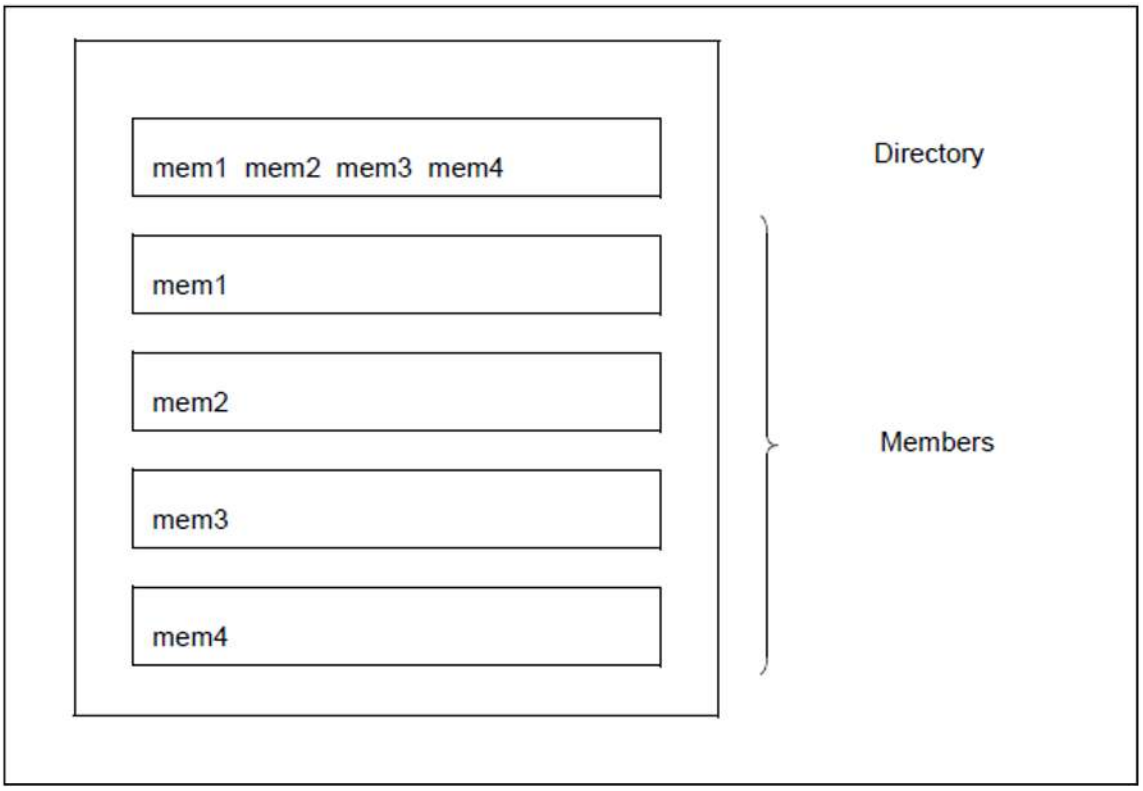


Figure 3: Logical structure of a library

Each library has a single entry in the system catalog. The user can define the name and other file attributes such as the retention period or shareability. Catalog entries and changes to them are made by the user with the aid of system commands.

7.1.2 Input and output libraries

LMSCONV processes a library in the form of an input and/or output library:

An input library is assigned globally by means of the LMSCONV statement OPEN-LIBRARY or locally by means of the operand ELEMENT=*LIBRARY-ELEMENT(LIBRARY=).

An output library is assigned globally by means of the LMSCONV statement OPEN-LIBRARY or locally by means of the operand TO-ELEMENT=*LIBRARY-ELEMENT(LIBRARY=).

7.1.3 Multiple access to libraries

Multiple access to libraries occurs when several users access the same library at the same time, either in read or in write mode. Access can take place from different tasks. The tasks can run under different user IDs.

Unrestricted multiple access to libraries is possible under the following conditions:

- the necessary access rights have been granted
- access is not via Remote File Access (RFA).
- for libraries on shared pubsets:
  the accessing tasks are running on different systems, that form an XCS network.

The user can explicitly restrict the use of multiple access with the following commands:

`/ADD-FILE-LINK ...,SUPPORT=*DISK(SHARED-UPDATE=*NO)`

Libraries are opened by default with SHARED-UPDATE=*YES.

`/ADD-FILE-LINK SHARED-UPDATE=*NO` and the opening of this library with the specified link name prevents further update accesses.

`/SECURE-RESOURCE-ALLOCATION FILE=`

The library is reserved exclusively for this task.

`/MODIFY-FILE-ATTRIBUTES ...,PROTECTION=`

Only users who have the necessary access rights can access the library.

For a description of these commands, see the “Commands” manual [1 (Related publications)].

7.2 Members

A member is a set of logically related data, e.g. a file, procedure, object module or source program. Each member can be addressed individually in the library by way of its member designation.

The member designation identifies a member and consists of three parts: name, version and type.

Name:	Describes the logical contents of the member.
Version:	Describes the current development status of the member.
Type:	Classifies the member.

Libraries may contain any LMSCONV-supported member types.

7.2.1 Multiple access to members

A member can be read simultaneously by several users; it can, however, be written to by one user only. When the member has been opened for writing, no other access - including read access - to this member can be performed, but access to other members of the library is possible.

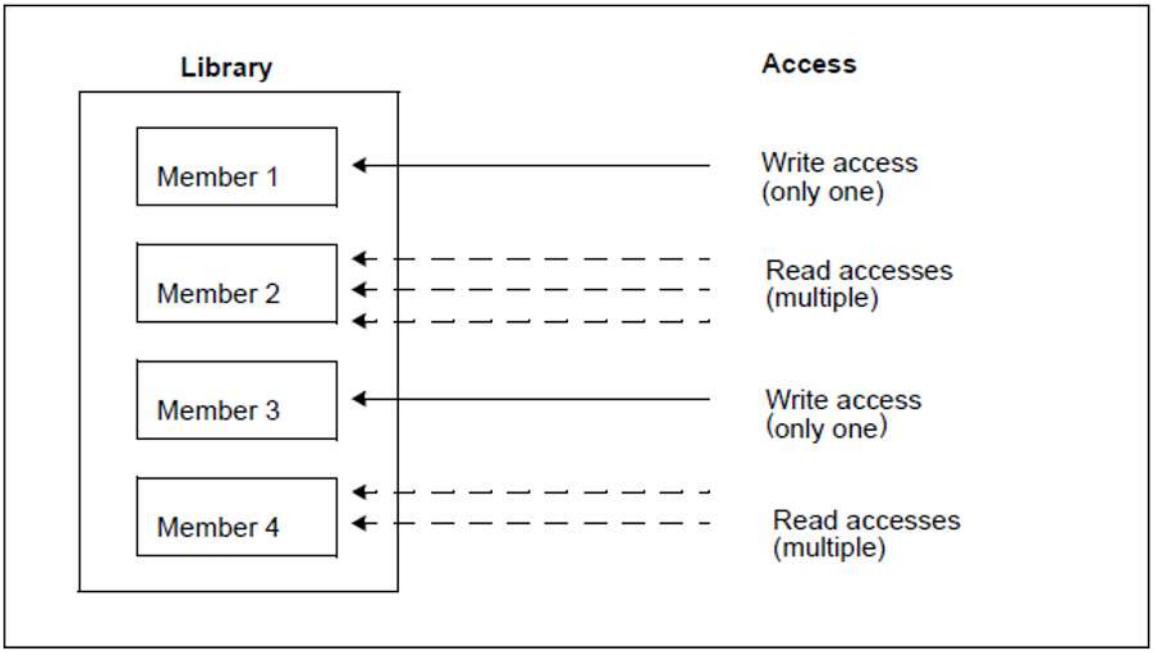


Figure 4: Multiple access to members

As a result of the multiple access options to a library a member may still exist while the directory is being listed, but be no longer in existence when it is subsequently accessed. Another user has deleted it in the meantime. A listing of the library's directory (statement SHOW-ELEMENT-ATTRIBUTES) will therefore only show the current state of the input library.

7.2.2 Member type definition

The member type indicates how the contents of the members are to be interpreted.

Standard types

Standard or predefined types are 1 character long, or start with $ or SYS

C	Load modules
D	Text data
F	IFG format masks
H	Compiler result information
J	Procedures
L	Link and load modules (LLMs)
M	Macros
P	Data edited for printing
R	Object modules
S	Source programs
U	IFG user profiles
X	Data of any format
SYSJ	Compiled procedure
$...	Reserved
SYS...	Reserved

The maximum record length of stored members is 32 Kbytes (including record header).

Member type C - phases

A phase generated by the linkage editor TSOSLNK is normally stored in a file. LMSCONV can be directed to write such a file as a C-type member to a library. Alternatively, the phases generated by the linkage editor can be stored directly in a library.

Member type D - Text data

Any text may be written to D-type members. The same functions are possible as with S-type members.

Member type F - IFG format masks

Members of this type are created by IFG and stored in the libraries.

Member type H - compiler result information

Members of this type are generated by the compilers and the assembler and stored in libraries.

Member type J - procedures

In this member type BS2000 procedures can be stored.

Member type L - link and load modules (LLM)

The BINDER (refer to the “BINDER” manual [12 (Related publications)]), and the compilers store the link and load modules (LLMs) created.

Member type M - Macros

The assembler takes the macro members addressed in the program from the assigned library.

Member type P - List members

Formatted data are referred to a list member. The first character of the record must be a valid line feed character; this is checked in the system file SYSLST.

Member type R - object modules

LMSCONV can store object modules created by compilers and the assembler and stored in a temporary EAM area in the library as members of type R. Optionally the object modules created by compilers and the assembler can also be stored directly in a library. The object modules and the dynamic link and load modules are used as inputs for these members.

Member type S - source programs

Source programs in libraries are used as inputs for the compilers and the assembler in compilation runs.

Member type U - IFG user profiles

Members of this type are created by IFG and stored in the libraries.

Member type X - data of any format

The member type X can accept any data.

Text-type member types - Text members

Types S, M, J, P, D, X are designated text-type member types. Text members are members of these types, provided that they do not contain and block-oriented records.

PAM members

Members with block-oriented records are referred to as PAM members below, since they are generally produced by adding a file as a member with FILE-STRUCTURE=PAM.

7.2.3 Convention for member designations

Members are identified in libraries by means of a member designation. This is stored in the directory of the library and can be output using the LMSCONV statement SHOW-ELEMENT-ATTRIBUTES.

The member designation comprises the following three components:

- member name for the logical contents of the members
- member version for the current status of the members
- member type for classification of the members.

7.2.4 Member designations in statements

The member designation, i.e. member name, version and type, in the LMSCONV statements corresponds to the ELEMENT, VERSION and TYPE operands in the data structure *LIBRARY-ELEMENT (see "Constructors for member designations").

The specification of the version is optional. If no value is specified for the version in a statement, the member having the highest value is selected by default. If a different version is to be selected, note that the version specification must be a substructure of the member name.

```
*LIBRARY-ELEMENT(...)
    |
    |  ELEMENT = <composed-name ... >(...)
    |     <composed-name ...>(...)
    |        |  VERSION = ...
    |  ,TYPE = ...
 ...
```

Syntax of the member designations

ELEMENT	composed-name 1..64 with-under with-wild(132)(...) The member name may begin with a catalog ID not exceeding four characters in length.
VERSION	composed-name 1..24 with-under with-wild(52)(...) For further information on possible version entries, with particular regard to the use and meaning of keywords, see section “Version management”. '@' is not permitted as a version entry since it is used to represent the version specification '*UPPER-LIMIT'.
TYPE	alphanum-name 1..8 with-wild(20)

7.2.5 Logging the member designations

The member designations are logged as follows with each output from LMSCONV:

```
(type)membername/version[(variantnumber)]/date
```

The variant number is set to (0001) by default and is incremented by 1 by each write access.

7.2.6 Selectors for member designations

If certain members are to be selected in the LMSCONV statements for processing, this can be done in two ways:

- through the use of “wildcard” specifications in the ELEMENT, VERSION and TYPE operands.

  The “wildcard” syntax is described in the “Commands” manual [1 (Related publications)].

  It is also possible to use the keyword *ALL for an individual asterisk (*) in the operands ELEMENT, VERSION and TYPE.

  Negative selection, i.e. exclusion of members, can also be effected through the use of the minus sign.
- through the qualification of attributes, e.g. date and time.

Examples of selectors

- Selectors

  ELEMENT=AB/C*

  All members whose names begin with AB, have any character in the 3rd, and a C in the 4th position are selected. The contents as of the 5th position are freely selectable.

  ELEMENT=<:999>(VERSION=B*)

  All members having a name length of up to 3 characters and with a B in the first position of the version are selected.

  ELEMENT=*(VERSION=*), CREATION-DATE=*INTERVAL(FROM=2016-01-01)

  All members entered since 1.1.2016 are selected.

  ELEMENT=AB*(VERSION=*HIGHEST-EXISTING)

  The highest version of all members whose name begins with AB is selected each time.
- Selectors with limiting values

  ELEMENT=A*(VERSION=*HIGHEST-EXISTING),USER-DATE=*INTERVAL(TO=2016-12-31)

  All members of the highest version whose names begin with A and which have a date earlier than 1.1.2017 are selected.

  ELEMENT=AB<:9>(V=107)

  All members whose names begin with AB, are up to 3 characters long and have the version 107 are selected.
- Multiple selection with members for exclusion

  ELEMENT=-ABC

  All members except member ABC are selected.

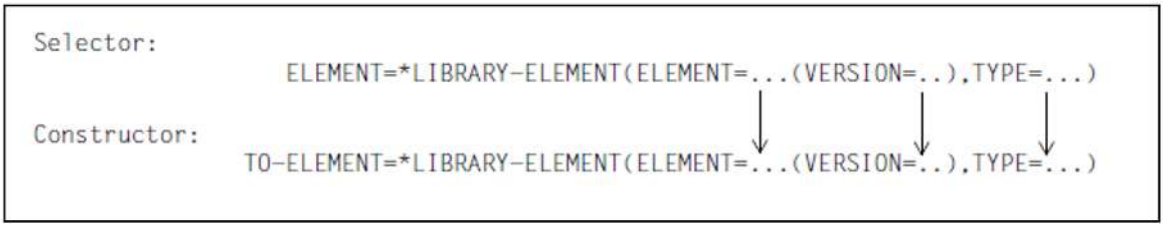
  TYPE=-S

  All members whose names begin with A and whose versions do not begin with B are selected.

7.2.7 Constructors for member designations

For those LMSCONV statements which permit a second member designation in addition to the selector, the designation of the second member can be constructed from the designation of the selector.

The constructor is restricted to the member designation, i.e. to member name, version and type. This corresponds in the statements to the ELEMENT, VERSION and TYPE operands in the data structure *LIBRARY-ELEMENT. In each case here like-named operands, which are identified by means of certain wildcard characters, are mapped onto one another.



The constructor syntax is described in the “SDF Dialog Interface” manual [20 (Related publications)].

Examples of constructors

The examples below illustrate how LMSCONV forms member names. The member type S is preset using the MODIFY-DEFAULTS statement. The individual examples are mutually independent, i.e. the result of one example is not used in another.

The following members are defined:

```
TYP  NAME    VER (VAR#) DATE          NAME    VER (VAR#) DATE
(S)  ABC     001 (0001) <date>        ABCD    001 (0001) <date>
(S)  ABCDE   001 (0001) <date>        X.1     001 (0001) <date>
```

Statement	Effect
`//COPY-ELEMENT (,ELEM= ABC),` `TO-ELEM=(,ELEM= ABX)`	`ABC` is copied to `ABX` `ABCD` is not copied `ABCDE` is not copied

`//COPY-ELEMENT (,ELEM= AB*),`
`TO-ELEM=(,ELEM= XY*(A02))`
	`ABC` is copied to `XYC(A02)`
	`ABCD` is copied to `XYCD(A02)`
	`ABCDE` is copied to `XYCDE(A02)`

`//COPY-ELEMENT (,EL= ABC<,D>),`
`TO-ELEM=(,ELEM= AXC<1>)`
	`ABC` is copied to `AXC`
	`ABCD` is copied to `AXCD`
	`ABCDE` is not copied

`//COPY-ELEMENT (,ELEM= AB*),`
`TO-ELEM=(,ELEM= S.AB*)`
	`ABC` is copied to `S.ABC`
	`ABCD` is copied to `S.ABCD`
	`ABCDE` is copied to `S.ABCDE`

`//COPY-ELEMENT (,ELEM= *B*),`
`TO-ELEM=(,ELEM= *.*)`
	`ABC` is copied to `A.C`
	`ABCD` is copied to `A.CD`
	`ABCDE` is copied to `A.CDE`

`//COPY-ELEMENT (,ELEM= /B/),` `TO-ELEM=(,ELEM= //)`	`ABC` is copied to `AC` `ABCD` is not copied `ABCDE` is not copied

`//COPY-ELEMENT (,ELEM= *CD*),`
`TO-ELEM=(,ELEM= <1>.<1>)`
	`ABC` is not copied
	`ABCD` is copied to `AB.AB`
	`ABCDE` is copied to `AB.AB`

`//COPY-ELEMENT (,ELEM= /B/),` `TO-ELEM=(,ELEM= <2> <1>)`	`ABC` is copied to `CA` `ABCD` is not copied `ABCDE` is not copied
`//COPY-ELEMENT (,ELEM= /B/),` `TO-ELEM=(,ELEM= XYZ<2>)`	`ABC` is copied to `XYZC` `ABCD` is not copied `ABCDE` is not copied
`//COPY-ELEMENT (,ELEM= X.),` `TO-ELEM=(,ELEM= Y.)`	`X.1` is copied to `Y.1`

Table 3: Effect of the COPY statement

When using selector and constructor structures, please observe the following:

- At least one placeholder of the source member must be present in the constructor specification.
- Several different input names may be mapped to the same output name. Depending on what has been set for the WRITE-MODE processing operand, the various data may be overwritten. Example: /A/ -> BA/. Both members XA1 and XA2 are mapped to BAX.
- *ALL in the selector specification can be referenced with a * in the constructor specification just like the single asterisk (*) in the selector specification. *ALL in the constructor specification is not possible. Example: *ALL -> *B* means the same as * -> *B*

7.2.8 Member attributes

All members have certain attributes, regardless of their type:

- CREATION-DATE and CREATION-TIME
- MODIFICATION-DATE and MODIFICATION-TIME
- USER-DATE and USER-TIME
- SECONDARY-NAME and SECONDARY-ATTRIBUTE (reference entry)(*1)
- CODED-CHARACTER-SET

The values of this attribute are displayed when outputting the contents of a library.

(*1) Reference entries are entries in the secondary directory of the library. They are created when the user creates reference records (record type 163) of the form
<secondary-name> <secondary-attribute> when writing a member.

The reference entries document the relation “one particular <secondary-name> and <secondary-attribute> occurs in the member”. This type-based sorting of the reference entries permits the query: “In which member of type TYPE does the reference entry occur?”.

This is implemented by the following LMSCONV statement:

```
//SHOW-ELEMENT-ATTR (*STD,*,TYPE= ...,SECONDARY-NAME=.., -
      SECONDARY-ATTRIBUTE=...,)
```

This designation is used for modules (type R or L) as the basis for the autolink function (see the “Binder” manual [12 (Related publications)]);
reference entries are, for example, `<name> <CSECT>` or `<name> <ENTRY>`.

Input format of the date

The input format of the date is as follows:

`[YY]YY-MM-DD`	`[YY]YY`	: Year; optionally 2- or 4-digit
	`MM`	: Month
	`DD`	: Day

If the year is entered in 2-digit form, LMSCONV adds the century data using the reference year:

```
YY   <    60      -->      20YY

YY   >=   60      -->      19YY
```

7.2.9 Type dependencies

The following table shows which member types can be used in the various LMSCONV statements and what type checks LMSCONV performs. Only if the conditions specified in the type check apply, will the relevant statement be executed.

Statement	Member type		
	Source	Target	Check
SHOW-ELEMENT SHOW-ELEM-ATTR DELETE-ELEMENT	alphanum 1...8 alphanum 1...8 alphanum 1...8		
COPY-ELEMENT	alphanum 1...8	alphanum 1...8	Source = Target
MODIFY-ELEMENT	R,C,L S,M,P,D,J,X	R,C,L S,M,P,D,J,X	Source = Target Source = Target
MOD-LOGGING-PAR		P	
ADD-ELEMENT	"text "(I)SAM-file "blocks " PAM-file "module " file,*OMF "phase " PAM-file	S,M,P,D,J,X X R C	
EXTRACT-ELEMENT	S,M,P,D,J,X X R C	(I)SAM-file PAM-File (I)SAM-file PAM-File	

Table 4: Possible member types for LMSCONV statements

7.2.10 Version management

The version of a member is defined in the member designation and identifies the current state of the member.

The following section describes the possible version designations and provides information on version maintenance and storage.

Version maintenance and storage

In libraries, a member is uniquely defined by its type, name and version designation. Furthermore, it is possible to store several versions under one member type and member name.

If the user does not specify the version to be processed, LMSCONV automatically performs the following actions:

- In read mode that member is sought which has the specified name and the highest version designation. The date is ignored.
- In write mode the actions depend on the statement:
  - `//ADD-ELEMENT` and `MODIFY-LOGGING-PARAMETERS TEXT-OUTPUT=`

    The member is generated or overwritten with the highest version X'FF'. LMSCONV identifies this version by @.
  - Other statements
    The output member is given the version designation of the input member.

    If an identically named member is overwritten, the internal variant number is incremented by 1. This serves as a write counter.

Different states of a development object are stored in different members. Initially, only the user is aware of the relationships between individual members; they are not anchored in the library. Each member is an independent unit in the context of a library.

Version designations

The member versions which are to be processed by LMSCONV are identified in the LMSCONV statements by means of the VERSION operand. A distinction is made between the source version and the target version.

- **Source version**
  If a member is used as the input for a function, e.g. if it is to be copied or changed, the member version is then called the source version.

  The source version can be specified as follows:
  - composed-name
    The version specified by composed-name is the source version.
  - *UPPER-LIMIT
    This entry selects the highest possible version (represented internally by X FF).
  - *HIGHEST-EXISTING
    This entry selects the highest existing version of the specified member name.

- **Target version**
  If the member is used as the result of a function, e.g. if it is to be written back, the member version is then called the target version.

  The target version can be specified as follows:

  - *BY-SOURCE
    The source version is also the target version. If the source is not a library member, *UPPER-LIMIT is the target version.
  - *UPPER-LIMIT
    The absolute highest version, represented internally by X'FF', is the target version.
  - composed-name
    The version specified by composed-name is the target version. If “@” is entered, it will be rejected.

7.2.11 Data protection by overwriting

Data protection by overwriting means that the user specifically deletes files that are no longer required by overwriting them. The data is physically deleted by this operation, i.e. overwritten with X'00'. Overwriting of data is controlled locally by the LMSCONV statement DELETE-ELEMENT or globally in MODIFY-DEFAULTS, by the DESTROY-DATA operand in each case.

The DESTROY-DATA operand is on the one hand a member attribute, i.e. overwriting automatically acts on this member, and on the other hand a processing parameter of the LMSCONV statement DELETE-ELEMENT. As a processing parameter, DESTROY-DATA causes all members covered by the statement to be overwritten on deletion.

The data is overwritten with X'00' if one of the following specifications calls for overwriting:

- system parameter DESTLEV
- specification for member: value of DESTROY-DATA on last creation or on last write access to the member
- specification via the DESTROY-DATA operand in the MODIFY-DEFAULTS statement

7.2.12 Auditing

The access method PLAM used by LMSCONV has an interface with the subsystem SAT (security audit trail) in the security package SECOS (see the “SECOS” manual [21 (Related publications)]). When SAT is active, the following events can be selected by the security administrator for logging:

- CREATE ELEMENT
- MODIFY ELEMENT
- READ ELEMENT
- EXECUTE ELEMENT
- CLOSE ELEMENT
- DELETE ELEMENT
- RENAME ELEMENT
- CREATE SECURITY ATTRIBUTES
- MODIFY SECURITY ATTRIBUTES
- DELETE SECURITY ATTRIBUTES

7.2.13 Extended Host Code Support (XHCS)

LMSCONV supports the use of special (national) character sets. This coded character set name (CCSN) assigned to the member is - as far as possible - forwarded to the interfaces and taken into account during output.

If XHCS is not offered at the relevant interface, the default “no code” is always used.

LMSCONV itself does not require a particular character set, and does not even interpret the default setting in the user ID; internal sorting procedures, e.g. for the member names, are carried out independently of the CCS.

1. Implicit setting of the CCSN for a member
   Each member of a library can be assigned a character set. LMSCONV then always transfers the CCSN of the source to the target member.
   This can happen by adding a file with the ADD-ELEMENT statement by adopting the catalog attribute CCS. However, the CCSN is not stored extra in the attribute record (record type 164), so as to avoid inconsistencies.

   If data is added from the logical system file SYSDTA, the particular character set applicable is determined and the name is assigned to the member as an attribute.

   When adding modules from the EAM file, the modules are assigned the attribute “no code”.

   When copying members, the CCSN of the source member is always assigned to the target member.
2. CCSN logging
   The SHOW-ELEMENT-ATTRIBUTES statement and the operand INFORMATION= *MAXIMUM provide information on the assignment of coded character sets to members. Members that have the CCSN “no code” are not shown in the display.
3. Interpreting and forwarding the CCSN of a member
   When members are output to files, the corresponding CCSN of the member is assigned to the file.

   The following applies to the output information created by LMSCONV:
   When outputting member records to SYSOUT (also formatted) with the SHOW-ELEMENT statement, the CCS of the corresponding member is used. If SYSOUT is assigned to a file, the user must explicitly assign the desired character set to this file with /MODIFY-FILE-ATTRIBUTES.

   When outputting member records to SYSLST, no CSSN is interpreted. If SYSLST is assigned to a file, the user can explicitly assign the desired character set to this file with /MODIFY-FILE-ATTRIBUTES .

   Extracting an element into a SAM node file on Net-Storage can fail because the CCSN of the element cannot be converted into the NETCCSN of the node file. This can have several cause:

   a. An inappropriate CCSN is assigned to the element. In this case a suitable CCSN can be assigned with // MODIFY-ELEMENT-ATTRIBUTES.
   b. An unsuitable NETCCSN is stored in the file attributes for the element (can be seen in the output of // SHOW-ELEMENT TEXT-INFORMATION = * FILE-ATTRIBUTES). In this case, the node file into which the element is to be extracted must be assigned the appropriate NETCCSN with / MODIFY-FILE-ATTRIBUTES. The element can then be extracted by specifying FILE-ATTRIBUTES = * BY-CATALOG.
   c. No NETCCSN is saved for the element (e.g. because it was not originally a node file or because the SOURCE-ATTRIBUTES = * KEEP operand was not specified when the node file was added to the library), and the user's preset NETCCSN does not match CCSN. In general, the user attribute NET-CODED-CHARACTER-SET should be changed accordingly in this case. In individual cases, however, the same procedure can be used as in case b.

   If the output stream is reassigned to a library member by MODIFY-LOGGING-PARAMETERS, this member receives the CCSN “no code”.

When outputting directories or other member information created by LMSCONV, the CCSN “no code” is always assumed.

4. Extending members and files with WRITE-MODE=*EXTEND
   If WRITE-MODE=*EXTEND is used, LMSCONV checks the CCS names of the source and target. If they do not match, processing is terminated with an error.

7.3 LMSCONV functions

This chapter gives an overview of the LMSCONV functions.

7.3.1 Starting LMSCONV

LMSCONV is called with the following command:

```
/START-LMSCONV
```

START	Alias: **LMSCONV**
VERSION = ***STD** / <product-version>	
,MONJV = ***NONE** / <filename 1..54 without-gen-vers>	
,CPU-LIMIT = ***JOB-RE**ST / <integer 1..32767 *seconds*>	

Preset options following LMSCONV startup

Default values come into effect following LMSCONV startup. These apply to certain operands in statements in which the keyword *DEFAULT may be specified. The default values can be modified using the MODIFY-DEFAULTS statement.

The following table indicates which statements are affected by the default values, or whether the default values influence the entire LMSCONV run.

LMSCONV default	Default value	Effect on	Meaning of the defaults
ELEMENT-ATTRIBUTES		Statements with type specification	
TYPE	*UNDEFINED *ALL	SHOW-ELEMENT-ATTR	Specifies member type
ELEMENT-VERSION	*BY-SOURCE	Target version	Specifies source version
TO-ELEM-VERSION	*IGNORE	ADD-ELEMENT	Specifies target version
SOURCE-ATTRIBUTES			Handles file attributes
FILE-ATTRIBUTES	*ISAM	EXTRACT-ELEMENT	Attributes for target file
DESTROY-DATA	*NO	DELETE-ELEMENT[1)]	Overwrites data
WRITE-MODE	*CREATE	ADD-ELEMENT COPY-ELEMENT EXTRACT-ELEMENT MODIFY-ELEMENT	Overwrites members

LMSCONV default	Default value	Effect on	Meaning of the defaults
DIALOG-CONTROL	*NO	ADD-ELEMENT COPY-ELEMENT DELETE-ELEMENT EXTRACT-ELEMENT MODIFY-ELEMENT MODIFY-ELEMENT-ATTRIBUTES	Interactive query-driven interface that offers each member individually.
INFORMATION LAYOUT SORT	*MEDIUM *VARIABLE *BY-NAME	SHOW-ELEM-ATTR " "	Information set Display Sort order
OUTPUT-FORM	*STD	SHOW-ELEMENT	Output form
DELETE-SOURCE	*NO	ADD-ELEMENT	Deletes source file
MAX-ERROR-WEIGHT	*SERIOUS	alle Anweisungen	Spin-off control
NEXT-ATTEMPT	*NO	Anweisungen mit *LIB-ELEM-Angabe	Controls attempts to open for file/type member lock
TEXT-INFORMATION MODULE-INFORMATION PHASE-INFORMATION LLM-INFORMATION	*ALL *ALL *ALL *ALL	SHOW-ELEMENT " " "	Info. for text type members Module information (type R) Phase information (type C) LLM information (type L)

Table 5: Effect of the default values

[1)] If the DESTROY-DATA operand is modified with MODIFY-DEFAULTS, it indirectly affects the ADD-ELEMENT, COPY-ELEMENT and EXTRACT-ELEMENT statements as well.

7.3.2 Assigning libraries

In LMSCONV statements, libraries are specified by means of the LIBRARY operand. There are several ways of assigning a library, all of which ultimately lead directly or indirectly to libraries:

1. Globally defined library (*STD)
2. Direct name of a library
3. Indirectly via a link name

Numbers 2 and 3 of the above options can be used to define the global library.

If a statement uses libraries, LMSCONV opens the libraries that are used in this statement. LMSCONV closes all other open libraries, regardless how many libraries have been used before in the current LMSCONV run. The statements, which don't use libraries (e.g. SHOW-DEFAULTS), don't influence the states of libraries.

The LMSCONV statement SHOW-LIBRARY-STATUS provides information on the status of the libraries to be processed.

Globally defined library (*STD)

The globally defined library is assigned by means of OPEN-LIBRARY and reset to undefined by means of CLOSE-LIBRARY.

Direct name of a library

The specified file name designates a library. This specification is often used during work with LMSCONV.

Indirectly via a link name

The name <link> specified under *LINK designates a link name. As a rule, the link name is a file link name defined via /ADD-FILE-LINK, to which a library is assigned in the FILE-NAME operand.

7.3.3 Processing members

The following sections provide an overview of the possible ways in which members can be processed with LMSCONV. LMSCONV can

- add elements to libraries
- output elements to files
- output elements to other libraries (copy)
- list elements
- delete elements
- correct elements
- output the directory of a library

Adding members to a library

The following statements add members to the assigned library:
ADD-ELEMENT, COPY-ELEMENT and MODIFY-LOGGING-PARAMETERS.

The WRITE-MODE operand determines whether or not an identically named member in the output library is overwritten.

ADD-ELEMENT

The ADD-ELEMENT statement (see "ADD-ELEMENT - Add member to library") adds files, modules from the EAM area and records from the LMSCONV statement stream to the assigned library as members. If no library is specified, the library opened by OPEN-LIBRARY is used.

The FIXED and UNDEFINED record formats are converted into the VARIABLE record format, i.e. given a 4-byte record header. Libraries permit files with a RECORD-SIZE of up to 32 Kbytes (including the record header) to be stored.

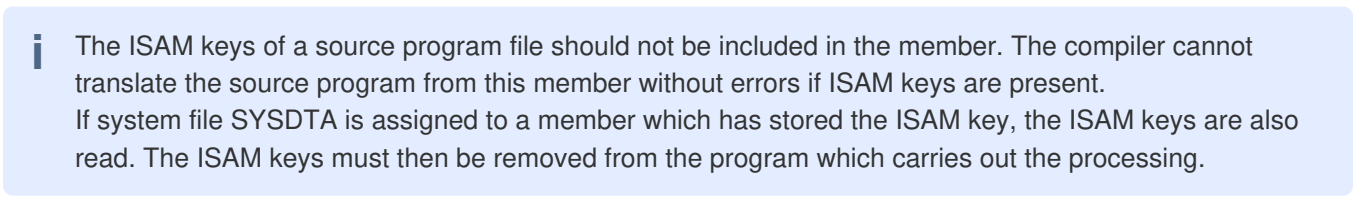
If an ISAM file is added, the SOURCE-ATTRIBUTES operand determines whether the file attributes, the ISAM key and information on ISAM secondary keys are included.

ISAM keys having a length of up to 255 bytes may then be stored. Members having ISAM keys are suitable only for archiving (see notes below).

If the operand SOURCE-ATTRIBUTES=*KEEP is set, it is also possible to include files with RECORD-FORMAT=*FIXED; if not, only RECORD-FORMAT=*VARIABLE is allowed.

The last byte pointer of a PAM file is always retained when the file is added to a library as a member of the type X. If it is added as a member of the type C, its LBP is only retained if SOURCE-ATTRIBUTES= *KEEP (KEEP-TYPES=*ALL).

For SAM node files, the coded character set on Net-Storage (NETCCSN) can be preserved if they are added to the library using ADD-ELEMENT SOURCE-ATTRIBUTES=*KEEP.



The ISAM keys of a source program file should not be included in the member. The compiler cannot translate the source program from this member without errors if ISAM keys are present.
If system file SYSDTA is assigned to a member which has stored the ISAM key, the ISAM keys are also read. The ISAM keys must then be removed from the program which carries out the processing.

Files can be stored under the following member types:

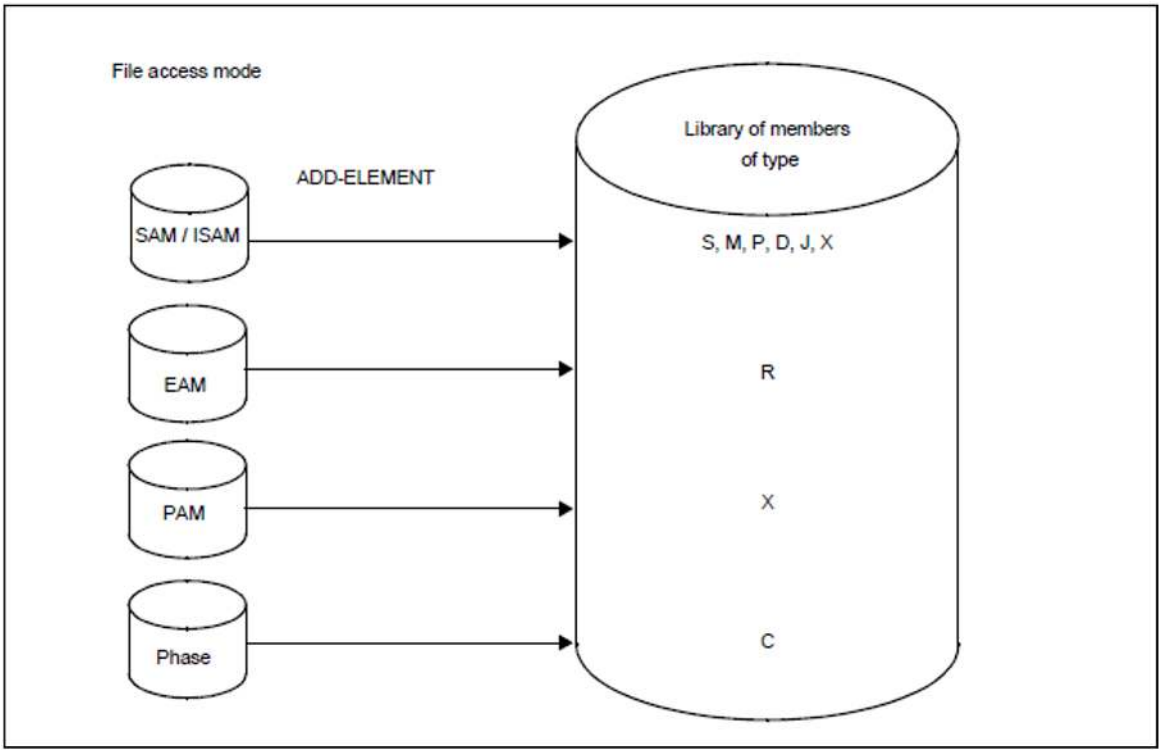


Figure 5: Adding members with ADD-ELEMENT

COPY-ELEMENT

The COPY-ELEMENT statement (see "COPY-ELEMENT - Copy member") copies members from the input library to the output library, storing them there with different member designations, if desired:

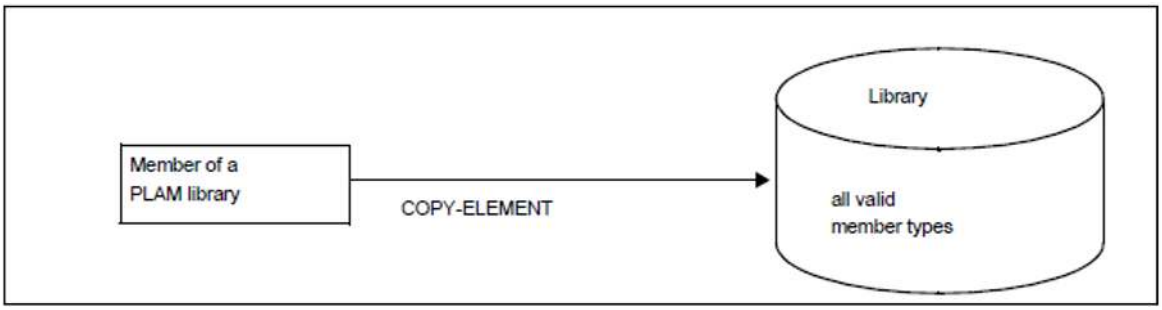


Figure 6: Adding members with COPY-ELEMENT

MODIFY-LOGGING-PARAMETERS

The statement MODIFY-LOGGING-PARAMETERS TEXT-OUTPUT=*LIBRARY-ELEMENT (see "MODIFY-LOGGING-PARAMETERS - Modify logging settings") writes the LMSCONV log in the member specified by *LIBRARY-ELEMENT:

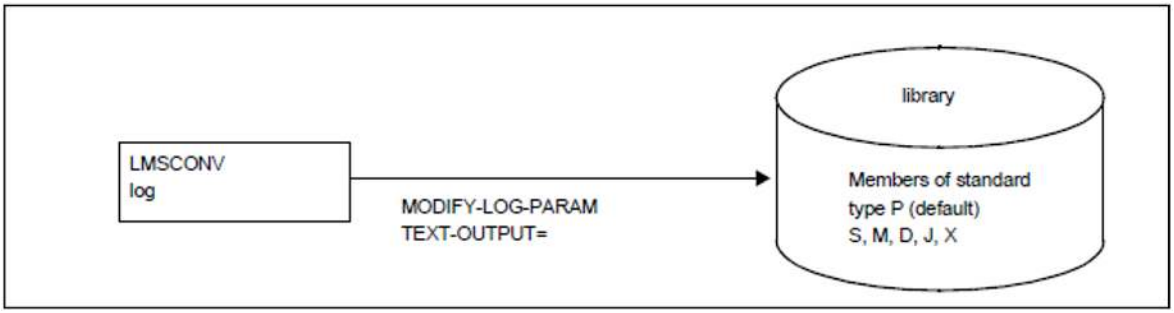


Figure 7: Storing the LMSCONV log in a member

Outputting members to a file

The members of a library are output to a file by means of the EXTRACT-ELEMENT statement (see "EXTRACT-ELEMENT - Output member to file") :

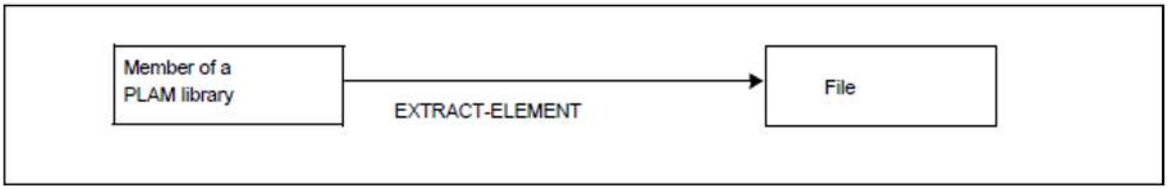


Figure 8: Outputting members

File attributes that were stored when adding a member will be restored when outputting the member to a file. This includes also the last byte pointer (LBP) for PAM files and the coded character set on Net-Storage (NETCCSN) for SAM node files.

> If the LBP is stored when adding a member, the member cannot be extracted to a tape or private disk. LMS with a version smallller than V3.4C cannot extract C-members with stored file attributes and may create an empty PAM file and/or output an error message.

Listing members

The SHOW-ELEMENT statement (see "SHOW-ELEMENT - Display contents of member") displays the member content. It is possible to define the format in which the members are to be output and how much information is to be displayed.

Deleting members

The DELETE-ELEMENT statement (see "DELETE-ELEMENT - Logically delete member") deletes members in the assigned library. A distinction is made between logical and physical deletion:

- Logical deletion
  The entries in the directory are deleted and storage space for the member concerned is released (this only affects space within the library, i.e. the size of the library itself is not reduced).
- Physical deletion
  In addition to logical deletion, the storage space of the corresponding member is overwritten with binary zeros. A member of a library is deleted physically if the DESTROY=*YES parameter has been set or if the member contains a code for physical deletion or if the system parameter DESTLEV requires it.

Correcting members

LMSCONV provides the following correction statement:

MODIFY-ELEMENT (see "MODIFY-ELEMENT - Modify member") corrects object modules, link and load modules, phases and text members via substatements (Member types R, L, C and member types S, M, J, P, D, X.)

Correction of these members is controlled by various substatements. These are read from the statement stream immediately following MODIFY-ELEMENT up to the END-MODIFY substatement.

The corrected member is then written back to the assigned library. It may receive a new member designation.

For types R, L and C, you may use the following functions:

- correct text records
- cancel corrections
- delete record types from the input member

For type R only, you may also use the following functions:

- generate REP records
- modify control section attributes
- rename symbols

For text members, you may use the following functions:

- insert records
- delete records

Outputting library directories

The SHOW-ELEMENT-ATTRIBUTES statement (see "SHOW-ELEMENT-ATTRIBUTES - Display member attributes") logs the directory entries of the specified members or of the entire library.

The directory is always output sorted by member type. The remainder of the sort sequence is determined by the SORT operand. Unless otherwise specified, member designations are output sorted by type, name and version.

To obtain the complete directory of a library, all you have to do is enter the SHOW-ELEMENT-ATTRIBUTES statement without further operands, provided no individual member type was specified using the MODIFY-DEFAULTS statement.

Storing procedures

LMSCONV allows the user to store procedures and ENTER jobs as members in libraries (member type J).

Existing procedure files can be added to libraries as members by means of ADD-ELEMENT.

Storing procedures in this way, especially where small command files are concerned, saves storage space. The number of catalog entries is decreased.

Note, however, that any ISAM keys that have been stored (see "Processing members") must be removed from the members before the procedure is called.

A library member can also be assigned as the system input file (SYSDTA) by means of /ASSIGN-SYSDTA (see the “Commands” manual [1 (Related publications)]).

7.3.4 Controlling the LMSCONV run

The following section describes the facilities provided by LMSCONV for controlling the LMSCONV run.

Controlling log output

The LMSCONV log contains everything output by LMSCONV, such as the result of the statements, their execution or abnormal termination, the assigned I/O libraries as well as lists generated, for example, when members are listed or compared.

The log may be written to the system file SYSOUT, SYSLST or to a library member. The output medium is specified by means of the TEXT-OUTPUT= operand of the MODIFY-LOGGING-PARAMETERS statement.

If the log is written to a member, LMSCONV normally creates a P-type member.

Error messages are always output.

The following table shows which operands in which statements control output of the log:

Statement	Operand	Function
MODIFY-LOGGING-PARAMETERS	LOGGING	Specifies whether or not positive acknowledgments are logged
	TEXT-OUTPUT	Specifies the output medium for the log
	OUTPUT-LAYOUT	Specifies the output format for the log
	LINES-PER-PAGE	Specifies the number of lines on a log page
	LINE-SIZE	Specifies the length of a line
	EXTRA-FORM-FEED	Controls the form feed of the log for member change
	HEADER-LINES	Specifies whether headers are output
SHOW-ELEMENT	OUTPUT-FORM	Specifies the record layout when listing member contents.
	TEXT-/ MODULE-/ PHASE-/ LLM-INFORMATION	Specifies the output scope when listing the member contents.
SHOW-ELEMENT-ATTRIBUTES	SORT	Specifies the sort mode for the contents
	LAYOUT	Specifies the log format of the contents
	INFORMATION	Specifies the log scope of the contents.
	TEXT-OUTPUT	Controls the log output

Table 6: Log output control

Positive and negative acknowledgments

If the operand LOGGING=*MAXIMUM is set in the LMSCONV statement MODIFY-LOGGING-PARAMETERS, the execution of each LMSCONV statement affecting a member will be logged. If the statement is executed successfully, LMSCONV will issue a positive acknowledgment.

If the statement cannot be executed, LMSCONV will log a negative acknowledgment and, if applicable, a corresponding LMSCONV error message.

All success and failure messages have the following format:

```
[NO] statement mem[word mem][cause]
```

Function

NO	The statement has not been executed
statement	Statement name
mem	Member designation or file name (in ADD-ELEMENT and EXTRACT-ELEMENT)
word	Keyword: AS, INTO, WITH
cause	Result: EXISTING, REPLACED, etc.

Controlling screen overflow

LMSCONV does not itself perform any screen overflow control. BS2000 handles this function. LMSCONV outputs can therefore only be aborted if the program interrupt key (K2) key is pressed and /INFORM-PROGRAM is subsequently entered and sent, see "MODIFY-DEFAULTS - Modify defaults".

Error handling in interactive and procedure modes

LMSCONV differentiates between interactive mode and procedure mode.

- Interactive mode

  In interactive mode, following output of the error message, the prompt // appears, requesting entry of the next statement.

- Procedure mode

  Users can themselves specify what errors are to trigger LMSCONV’s spin-off mechanism. They control this by setting the MAX-ERROR-WEIGHT operand (see the MODIFY-DEFAULTS statement). According to the setting of this operand, serious errors cause LMSCONV to branch to the next STEP or END statement. If a serious error occurs during the processing of substatements, the associated main statement is aborted and the spin-off mechanism is activated. This means that a main statement is always expected to follow STEP. If the spin-off mechanism is activated in LMSCONV without a STEP having been read, LMSCONV then terminates with TERM UNIT=STEP,MODE=ABNORMAL.

Interrupting the LMSCONV run

The user can interrupt the LMSCONV run by pressing one of the program interrupt keys (e.g. K2).

Continuation of the LMSCONV run can be controlled by /INFORM-PROGRAM, which may optionally be supplied with an input text. This input text is subsequently interpreted by LMSCONV during interrupt handling. The current function is informed of the type of termination and terminates in the desired way. The possible inputs are described under the DIALOG-CONTROL operand of the MODIFY-DEFAULTS statement (see "MODIFY-DEFAULTS - Modify defaults").

Termination of LMSCONV due to errors

Error handling is also controlled by means of the STXIT routine.

In the case of program termination, connection failure, or specification of `/START-EXECUTABLE-PROGRAM`, `/LOAD-EXECUTABLE-PROGRAM`, `/CANCEL-JOB`, `/LOGOFF`, `/CANCEL-PROGRAM`, `/ABEND`, `/EXIT-JOB`, a check is performed to ensure that the libraries remain consistent.

The following applies to all program termination conditions:

- All STXIT routines in LMSCONV are deactivated in order to prevent any incorrect continuation of processing by /INFORM-PROGRAM.
- LMSCONV simulates an END statement. This causes all open libraries to be closed.

7.3.5 Disks without PAM key

New disk formats, particularly the non-key (NK) format, have been introduced to increase the storage capacity and the data transfer rate of disk storage. A non-key (NK) disk is formatted without PAM key. In order to be able to use the NK disk, K files must be converted to NK files (PAM key elimination).

BS2000 only supports disks with fixed block size (2 Kbytes, 4 Kbytes, etc.). These fixed block sizes are not readily compatible with PAM keys. For this reason the PAM keys have been done away with.

There are two different file formats on disk for SAM, ISAM and UPAM files; the previous format tied to the PAM key (K format) and the non-key format (NK2 and NK4 formats).

The file format is defined by the BLKCTRL value. BLKCTRL can assume the value PAMKEY, DATA, DATA2K, DATA4K or NO. For details on the file formats, please refer to the “Introductory Guide to DMS” [4 (Related publications)].

Library files

The distinction between K and NK format derives primarily from DMS. This distinction is reflected in the following ways in the internal file organization of the library:

The PAM key is not required. With regard to files, however, there is difference which is represented by the BLKCTRL file attribute.

Libraries need not be converted with PAMCONV when migrating between the K and the NK environments.

Member processing

The following diagram provides an overview of the situations which may arise when transferring data between the file and library members. For members, logical information units are listed; for files, the BLKCTR value is given. The arrows indicate the transfer direction.

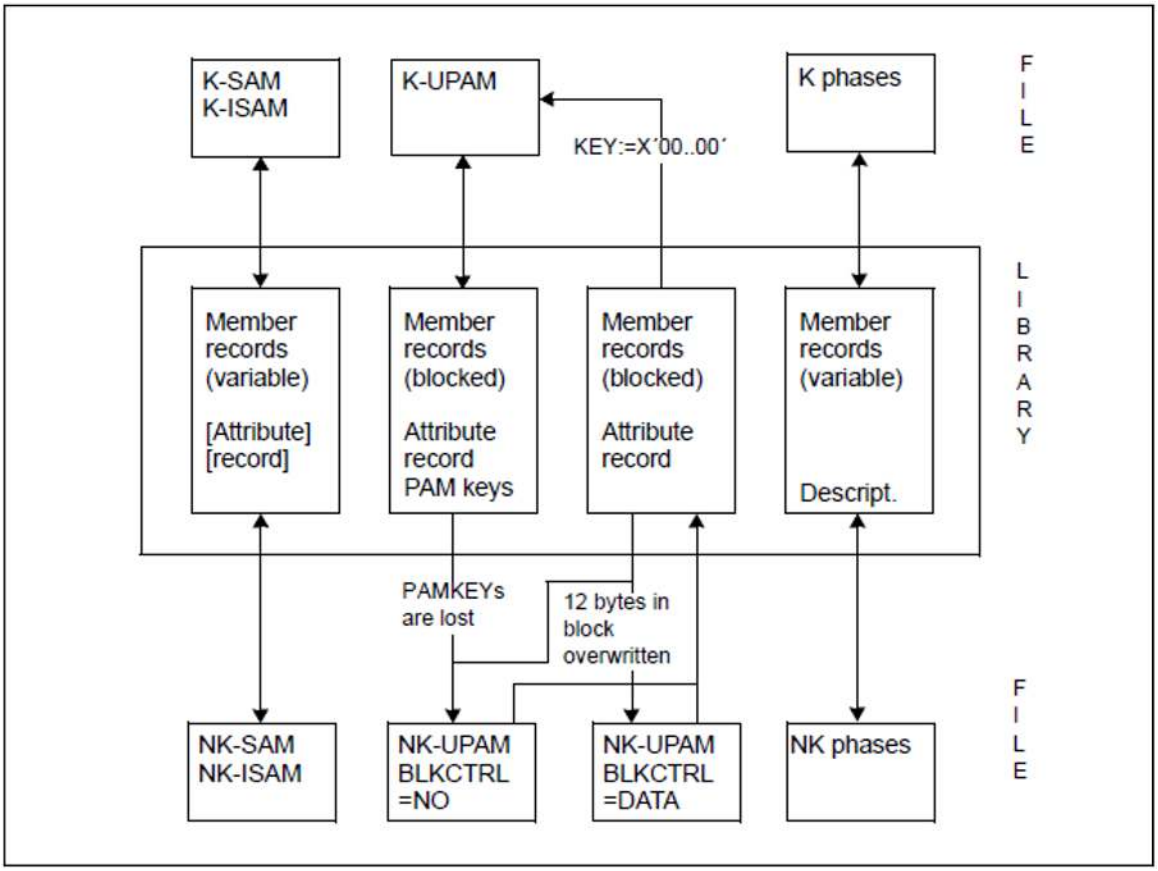


Figure 9: Transfer of information between the file and the library members

Use of the ADD-ELEMENT statement

The ADD-ELEMENT statement is used to store file contents in members. Note the following details:

- SAM/ISAM files

  When SAM and ISAM files are added, the BLKCTRL value is also stored if SOURCE-ATTRIBUTES=*KEEP has been set, i.e. the original file block structure determined by the BLKCTRL value is documented in the attribute record.

  The individual records are read using the SAM/ISAM logical access method and written unchanged to the member as variable-format records.

  The member structure generated is independent of the original BLKCTRL attribute.

- PAM files

  When PAM files are added, the BLKCTRL value, too, is always stored. The blocks of the file are read using the UPAM access method and stored unchanged as a block in the member. If PAM keys are specified, i.e. BLKCTRL=PAMKEY, these PAM keys are stored in the member.

  The generated member thus retains the block structure determined by the BLKCTRL value.

- phases

  When phases are added, the BLKCTRL value is not stored. The corresponding format specification is stored on file in the phase information. In the library, K phases and NK phases have the same format. The PAM key information is stored in descriptors.

ADD file > member	File type	BLKCTRL entry in attribute record	PAMKEY storage
File resides on NK disk	SAM/ISAM	--[1)]	
	SAM/ISAM	from the catalog	no
	UPAM	from the catalog	no
File resides on K disk	SAM/ISAM	--[1)]	no
	SAM/ISAM	from the catalog	no
	UPAM	from the catalog	for BLKCTRL=PAMKEY

Table 7: BLKCTRL and PAMKEY for ADD-Element

[1)] Storage can be controlled via the SOURCE-ATTRIBUTES operand

Use of the EXTRACT-ELEMENT statement

The EXTRACT-ELEMENT statement is used to output the contents of members to files. The BLKCTRL value is determined in accordance with the following hierarchy:

1. The specification in the catalog entry or /ADD-FILE-LINK.
2. BLKCTRL value stored for the member. This is relevant only for files which were original PAM files.
3. Settings of the system parameter BLKCTRL= PAMKEY or NONKEY. This can be displayed by means of /SHOW-SYSTEM-PARAMETERS.
4. Disk attribute PAMKEY or NONKEY.

If no catalog entry exists and the BLKCTRL value has not been stored, the system parameter and the disk attribute determine the BLKCTRL value:

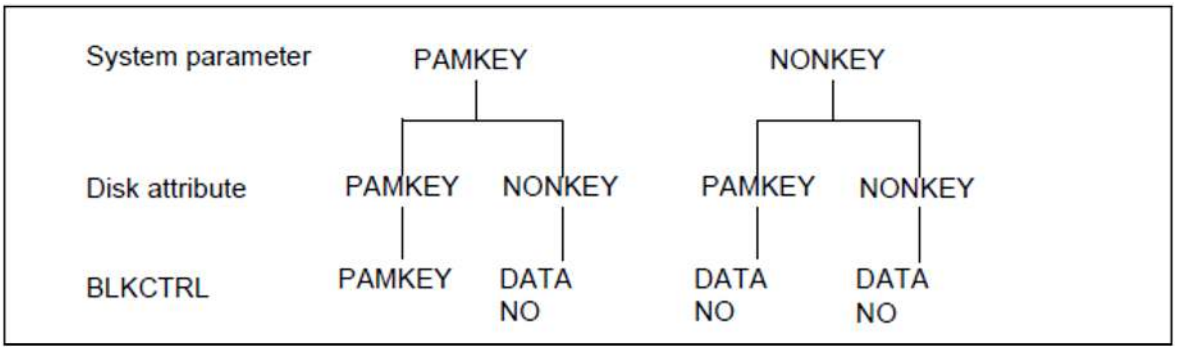


Figure 10: Relationship between system parameter and BLKCTRL value

If the system parameter has been set to PAMKEY, LMSCONV lets the system define the BLKCTRL value, i.e. BLKCTRL is not specified.

If the system parameter has been set to NONKEY, LMSCONV sets BLKCTRL=DATA for SAM and ISAM files, and BLKCTRL=NO for PAM files.

Note the following details:

- ISAM files

  Variable-length member records are written using the ISAM logical access method. The BLKCTRL value of the file is determined according to the algorithm described above; in this case, however, point 2 of the hierarchy above does not apply, as the BLKCTRL value stored for the member is used for documentation purposes only and is ignored.

- SAM files

  If BLKCTRL=DATA is specified, a DMS error occurs if records in the member are longer than 32 Kbytes - 16 bytes. In the K environment, these records may have a length of up to 32 Kbytes - 4 bytes. When selecting records, LMSCONV passes those which are too long to DMS without checking them. The BLKCTRL value is determined in the same way as for ISAM.

- PAM files

  In the NK environment, the PAM keys are lost. In addition, when BLKCTRL=DATA is specified, the first 12 bytes of each logical block are overwritten by the system. In both cases LMSCONV issues a warning.

- Phases (C-type members)

  Phases (C-type members) are handled in a special way. In addition to the old phase format (K phase), there is a new “PAM-key-free” phase format (NK phase) for files.

Summary

- SAM/ISAM files

  It is always possible to add and select files. Any BLKCTRL values stored are used for documentation purposes only.

  The internal file format is always determined by the SAM/ISAM access method. This method also converts records to the internal block format of the file.

- UPAM files

  Neither the UPAM access method nor LMSCONV can be used for the automatic conversion of data, since this would result in data being lost.

  Control rests ultimately with the user.

File type UPAM	**Created / stored BLKCTRL entry in the attribute record**			
	PAMKEY	**DATA**	**NO**	**--**
File resides on NK disk	[1] [2]	ADD EXTRACT	ADD EXTRACT	--- EXTRACT
File resides on K disk	ADD EXTRACT	ADD EXTRACT	ADD EXTRACT	--- EXTRACT

Table 8: BLCTRL entry depending on the disk format for UPAM files

[1] The value BLKCTRL=PAMKEY is not possible

[2] The selection process must be controlled by the user, e.g. by specifying a link name in the statement.

7.3.6 NK4 disks

In BS2000 there are two formats for libraries. A 2K-oriented format (NK2 PLAM file) and a 4K-oriented format (NK4 PLAM file). The COPY-LIBRARY statement converts from one format to the other. The user determines the appropriate format with the aid of /ADD-FILE-LINK ...,BUFFER-LENGTH=*STD(1 or 2). LMSCONV supports both library formats. It also supports NK4 disks with the ADD-ELEMENT and EXTRACT-ELEMENT statements.

Adding files with ADD-ELEMENT

Using the ADD-ELEMENT statement, files of any BUFFER-LENGTH can be added to a library.

Outputting files with EXTRACT-ELEMENT

For the EXTRACT-ELEMENT statement, a distinction is made between the following cases:

1. The member has an attribute record with the original BUFFER-LENGTH specification (e.g. following ADD-ELEMENT with SOURCE-ATTR=*KEEP or for original UPAM files, e.g. also libraries).
   a. A BUFFER-LENGTH value is explicitly preset for the target file, either through an entry in the TASK-FILE-TABLE (TFT) via /ADD-FILE-LINK or directly in the catalog. In this case, the preset value is always used. When the value is transferred, the following problems may occur:
      - SAM/ISAM file

        The member records are too long for the preset BUFFER-LENGTH. A DMS error is then reported.
      - UPAM file

        When creating UPAM files, LMSCONV fills up a logical block (except for the last one) with 2K units and only then outputs it with UPAM.

        With BLKCTRL=DATA, every logical block (BUFFER-LENGTH) begins with a 12-byte block control field (CF). If the preset BUFFER-LENGTH does not correspond to the stored value, data can be overwritten with the CF by the DMS. The file is then unusable.

        With BLKCTRL=NO, however, unusable files may likewise be generated if the BUFFER-LENGTH is changed (e.g. PLAM files).

        This is why LMSCONV generally outputs a warning for different BUFFER-LENGTH values (user specification versus stored value). However, it always attempts to create the file.

   b. **No** BUFFER-LENGTH value is specified or known for the target file. In this case, the value is taken from the attribute record.

      If n in STD(n) is odd, LMSCONV increments to n+1.
2. The member does not contain **any** attribute record, e.g. for phase members.
   a. The BUFFER-LENGTH value is specified explicitly for the target disk. Procedure as for 1a above.

      When creating phases, specifying BUFFER-LENGTH values not equal to STD(1) or STD(2) produces an error.
   b. **No** BUFFER-LENGTH value is specified or known for the target file.
      - For phases, the BUFFER-LENGTH is given by the current environment, i.e. BUFFER-LENGTH=STD(1) on NK2 disks and BUFFER-LENGTH=STD(2) on NK4 disks. The contents of these phases do not differ.
      - Otherwise, BUFFER-LENGTH is calculated on the basis of the maximum record lengths.

All things considered, the following procedure is recommended if files are to be brought onto an NK4 disk via a library:

1. Actions on the NK2 disk:
   Extract all “critical” members of the library as files. Critical members are “PAM” members under type X, which, as a file, have one of the following characteristics:
   - BUFFER-LENGTH=STD(n), where n is odd
   - PAM key phases or
   - 2K-oriented PLAM files.
2. Using PAMCONV, convert all files with an odd BUFFER-LENGTH (except PLAM files) into NK4 files.
3. Using PAMCONV, convert all PAM key phases into NK phases.
4. Using the LMSCONV statement COPY-LIBRARY, convert NK2 PLAM files to NK4 PLAM files.
5. Then use the ADD-ELEMENT statement to add the NK4 files to an NK4 PLAM file and transfer that file to the NK4 disk.

7.3.7 Handling alias names (ACS)

The ACS (Alias Catalog System) subsystem permits the use of alias names for managing files. How LMSCONV treats ACS alias names is described below.

Formulation of member names

The file name converted by ACS is always used for building member names.

Example

Alias X becomes file name FILE.X

The LMSCONV statement ADD-ELEMENT X,(,*BY-SOURCE(001),S) creates the member S/FILE.X/001

Formulation of file names

The current file name is always used for building file names. It is then possibly converted by ACS.

Example

Alias X becomes file name FILE.X

The LMSCONV statement EXTRACT-ELEMENT (,X,S),*BY-SOURCE creates the file FILE.X

Logging file names

LMSCONV always logs the fully converted file name.

7.4 Statements

- Overview of the LMSCONV statements
- LMSCONV statements ADD-ELEMENT to MODIFY-ELEMENT
  - ADD-ELEMENT - Add member to library
  - CLOSE-LIBRARY - Close library
  - COPY-ELEMENT - Copy member
  - COPY-LIBRARY - Copy library
  - DELETE-ELEMENT - Logically delete member
  - END - Terminate LMSCONV
  - EXTRACT-ELEMENT - Output member to file
  - MODIFY-DEFAULTS - Modify defaults
  - MODIFY-ELEMENT - Modify member
- Substatements of MODIFY-ELEMENT for member types R, C and L
  - ADD-REP-RECORD - Add REP records to object module
  - ADD-TEXT-MODIFICATION - Correct text records of an object module
  - DELETE-RECORD-TYPE - Exclude record types from input member
  - END-MODIFY - Terminate input of substatements
  - MODIFY-CSECT-ATTRIBUTES - Modify CSECT attributes
  - MODIFY-MODIFICATION-DEFAULTS - Specify global defaults
  - REMOVE-MODIFICATION - Cancel corrections
  - RENAME-SYMBOLS - Rename symbols
- Substatements of MODIFY-ELEMENT for text members
  - ADD-RECORD - Add records
  - END-MODIFY - Conclude substatements
  - REMOVE-RECORD - Delete record or record area in member
- LMSCONV statements MODIFY-ELEMENT-ATTRIBUTES to WRITE- COMMENT
  - MODIFY-ELEMENT-ATTRIBUTES - Modify member attributes
  - MODIFY-LOGGING-PARAMETERS - Modify logging settings
  - OPEN-LIBRARY - Open global library
  - SHOW-DEFAULTS - Output current default values
  - SHOW-ELEMENT - Display contents of member
  - SHOW-ELEMENT-ATTRIBUTES - Display member attributes
  - SHOW-LIBRARY-ATTRIBUTES - Display library attributes
  - SHOW-LIBRARY-STATUS - Display library status
  - SHOW-LOGGING-PARAMETERS - Display global LMSCONV parameters
  - SHOW-TYPE-ATTRIBUTES - Display attributes of a member type
  - SHOW-USER-EXITS - Display LMSCONV version
  - WRITE-COMMENT - Write comments to output medium

7.4.1 Overview of the LMSCONV statements

The syntax of the SDF command/statement language is explained in the “Commands” manual [1 (Related publications)]. The following short forms are used in this manual:

cat-id	cat
completion	compl
generation	gen
lower-case	low
underscore	under
userid	user
version	vers
wildcards	wild

The keyword *DEFAULT is no longer described in the individual statements. It always means the value set with the MODIFY-DEFAULTS statement.

The following also applies to the member type:

The member type is preset to *UNDEFINED. Therefore, MODIFY-DEFAULTS must first be used to define a member type, since the specification *DEFAULT would otherwise produce an error.

Input rules

The LMSCONV statements are read via the SDF user interface and processed by the command processor SDF (System Dialog Facility). Different forms of guided or unguided dialog exist, enabling you to request help menus for the statements. See the “SDF Dialog Interface” manual [20 (Related publications)].

Continuation lines

> It is also possible for statements to extend over more than one record. Splitting is governed by the BS2000 command language conventions. A hyphen (-) is used as the separator character. Statement lines may be up to 16364 characters long.

Abbreviation options

> When entering LMSCONV statements it is permissible to abbreviate statement names, operand names and keywords.
>
> The following rules then apply:
> It is possible in each case to abbreviate from right to left as long as uniqueness is maintained. This applies both to the name as a whole and to subnames (beginning with a hyphen) and allows for the possibility of the subname being omitted entirely.
>
> The guaranteed abbreviation options for all statements, operands and operand values are indicated in the syntax descriptions of the statements by boldface print. It is, however, possible to abbreviate beyond these (so long as uniqueness is maintained within a structure).
>
> Either no abbreviations or only guaranteed abbreviations should be used in procedures.

Positional operands

SDF allows the optional specification of operands as keyword operands or positional operands. However, the possibility of an operand position changing in a subsequent version cannot be completely ruled out. It is therefore advisable to avoid using positional operands in procedures.

7.4.2 LMSCONV statements ADD-ELEMENT to MODIFY-ELEMENT

- ADD-ELEMENT - Add member to library
- CLOSE-LIBRARY - Close library
- COPY-ELEMENT - Copy member
- COPY-LIBRARY - Copy library
- DELETE-ELEMENT - Logically delete member
- END - Terminate LMSCONV
- EXTRACT-ELEMENT - Output member to file
- MODIFY-DEFAULTS - Modify defaults
- MODIFY-ELEMENT - Modify member

7.4.2.1 ADD-ELEMENT - Add member to library

ADD-ELEMENT adds files as members to a library. The member data is read from SYSDTA as standard. It can, however, also be read from an explicitly specified file or *OMF. The files are always added as a member to a library without a prefix, i.e. without catalog ID or user ID, unless the user has explicitly specified a prefix in the construction specification.

Files cataloged with RECORD-FORMAT=*UNDEFINED can also be incorporated in libraries. Files having RECORD-FORMAT=*FIXED can only be stored using SOURCE-ATTRIBUTES=*KEEP. The record formats FIXED and UNDEFINED are converted into the VARIABLE record format, i.e. are given a 4-byte record header. The record length, including record header, must not exceed 32 Kbytes.

File generation groups can only be incorporated using link names and a valid LMSCONV member designation.

In the case of the ADD-ELEMENT statement, LMSCONV adopts the catalog attribute CCS of the file as a member attribute. If the data is read from SYSDTA, the member generated is given the CCS name set for SYSDTA as an attribute. If the data is read from *OMF, the member is assigned “no code”.

Format

ADD-ELEMENT

FROM-FILE = ***STD** / ***SYSDTA**(...) / ***ALL** / <filename 1..80 without-vers with-wild> / ***LINK**(...) / ***OMF**

***SYSDTA**(...)

| **END** = '***END**' / <c-string 1..8>

***LINK**(...)

| **LINK**-NAME = <structured-name 1..8>

,**TO-E**LEMENT = ***LIB** RARY **-ELEM** ENT (...)

***LIB**RARY**-ELEM**ENT(...)

| **LIB**RARY = ***STD** / ***LINK**(...) / <filename 1..54 without-vers>

| ***LINK**(...)

| | **LINK**-NAME = <structured-name 1..8>

| ,**ELEM**ENT = ***BY-SOUR** CE (...) / <composed-name 1..132 with-under with-wildcard-constr>(...)

| ***BY-SOURCE** (...)

| | **VERSION** = ***DEFAULT** / ***UP**PER**-LIM**IT / <composed-name 1..24 with-under>

| <composed-name 1..132 with-under with-wildcard-constr>(...)

| | **VERSION** = ***DEFAULT** / ***UP**PER**-LIM**IT / <composed-name 1..24 with-under>

| ,**TYPE** = ***DEF**AULT / <alphanum-name 1..8>

| ,**USER-DATE** = ***TODAY** / ***BY_SOURCE** / <date 8..10 with-compl>

,**ELEM**ENT**-ATTR**IBUTES = ***DEFAULT** / ***PAR**AMETERS(...)

***PAR**AMETERS(...)

| **SOUR**CE**-ATTR**IBUTES = ***DEFAULT** / ***STD** / ***IGNORE** / ***KEEP**(...)

| ***KEEP**(...)

| | **KEEP-TYPES** = ***DEFAULT** / ***STD** / ***ALL**

,**DEL**ETE**-SOUR**CE = ***DEFAULT** / ***NO** / ***Y**ES

,**WR**ITE**-MODE** = ***DEFAULT** / ***CREATE** / ***REPLACE** / ***EXTEND** / ***ANY**

,**DIALOG-CONTR**OL = ***DEFAULT** / ***NO** / ***Y**ES / ***ERROR**

Operands

FROM-FILE = *STD / * SYSDTA(...) / *ALL / <filename 1..80 without-vers with-wild> / *LINK (...) / *OMF
Specifies the file to be added to the library as a member.

FROM-FILE = *STD
Data records are read from the default file, i.e. the system file SYSDTA.
Permissible member types: S, M, P, J, D, X

FROM-FILE = *SYSDTA(...)
The records are read with RDATA from system file SYSDTA. The records must directly follow the ADD-ELEMENT statement.
Permissible member types:S, M, P, J, D, X, R

END = '*END' / <c-string 1..8>
End criterion for the input. The sequence of records must be concluded with “*END” or a user-defined end criterion. If the input data contains no end criterion, reading continues to EOF.

> If records are read from the system file SYSDTA (=SYSCMD), they must not begin with “/”. The reason for this is that the RDATA macro interprets such records as commands and thus passes the return code for EOF. Therefore it is not possible to pass system commands as records.

FROM-FILE = <filename 1..80 without-vers with-wild >
The data is read from the specified file.
Permissible member types:S, M, P, J, D, X, R, C
Files of the PAM type can be stored only under the member type X.

FROM-FILE = *LINK(...)
The data is read from the file specified via the link name.

LINK-NAME = <structured-name 1..8>
Link name referencing the file.

FROM-FILE = *OMF
Applies only to R-type members.
The data is read from the OMF file. All modules from the OMF file are incorporated. If the EAM area contains more than one module of the same name, LMSCONV adds the last module processed to the library.

TO-ELEMENT = *LIBRARY-ELEMENT(...)
Specifies the destination and name under which the member is to be added.

LIBRARY = *STD / *LINK(...) / <filename 1..54 without-vers>
Specifies the library to which the member is to be added.

LIBRARY = *STD
The library opened globally by OPEN-LIBRARY.

LIBRARY = *LINK(...)
The library assigned via a link name.

LINK-NAME = <structured-name 1..8>
Link name of the library.

LIBRARY = <filename 1..54 without-vers>
Name of the library to which the file is to be added as a member.

ELEMENT = *BY-SOURCE(...) /
<composed-name 1..132 with-under with-wildcard-constr>(...)
Name that the new member to be added is to receive. A constructor specification refers to the file name.

ELEMENT = *BY-SOURCE(...)
The member name corresponds to the file name or, in the case of *OMF, to the module name.

VERSION = *DEFAULT / *UPPER-LIMIT / <composed-name 1..24 with-under>Version that the new member to be added is to receive.

VERSION = *DEFAULT
The default value *BY-SOURCE or the current value set MODIFY-DEFAULTS.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' is generated.

VERSION = <composed-name 1..24 with-under>
The text specified here is interpreted as the version designation.

ELEMENT = <composed-name 1..132 with-under with-wildcard-constr>(...) Specifies the name under which the member is stored.

VERSION = *DEFAULT / *UPPER-LIMIT / <composed-name 1..24 with-under>Version that the new member to be added is to receive.
For a description of the operands, see above.

TYPE = *DEFAULT / <alphanum-name 1..8>
Type that the new member to be added is to receive. If the value is *DEFAULT and the current value set with MODIFY-DEFAULTS is *NONE, LMSCONV requires a type specification.

USER-DATE = *TODAY / *BY-SOURCE / <date 8..10 with-compl>
Date given by the user.

USER-DATE = *TODAY
The current date is given.

USER-DATE = *BY-SOURCE
The CHANGE-DATE of the file is taken over.

USER-DATE = <date 8..10 with-compl>
The date must be specified in the form [YY]YY-MM-DD.

ELEMENT-ATTRIBUTES = *DEFAULT / *PARAMETERS(...)
Determines whether the file characteristics and also the ISAM key are incorporated into the output member..

ELEMENT-ATTRIBUTES = *DEFAULT
The member attributes TYPE=*UNDEFINED, VERSION=*BY-SOURCE and SOURCE-ATTRIBUTES=*IGNORE or the current value set with MODIFY-DEFAULTS.

ELEMENT-ATTRIBUTES = *PARAMETERS(...)

SOURCE-ATTRIBUTES = *DEFAULT / *STD / *IGNORE / *KEEP(...)
Stores file attributes. This operand has no effect if the data is read from SYSDTA or *OMF. Original attributes are not stored. If the data is read from a file of the type UPAM, this entry has no effect; it is always as though *KEEP had been specified.

SOURCE-ATTRIBUTES = *DEFAULT
The default value is *IGNORE (see below) or the current value set via MODIFY-DEFAULTS.

SOURCE-ATTRIBUTES = *STD
No file attributes are stored, and neither is an ISAM key. In this case, it is only possible to add ISAM files to the member with KEY-POSITION=5, KEY-LENGTH<= 16 and RECORD-FORMAT=*VARIABLE.
For ISAM files a warning is issued that the ISAM key has not been added.

SOURCE-ATTRIBUTES = *IGNORE
As for SOURCE-ATTRIBUTES=*STD, except that no warning is issued.

SOURCE-ATTRIBUTES = *KEEP(...)

The following file attributes are stored unchanged in the new member being added: ACCESS-METHOD, RECORD-FORMAT, RECORD-SIZE, BUFFER-LENGTH, PERFORMANCE, USAGE, ACCESS and USER-ACCESS. If ACCESS-
METHOD=ISAM is specified, the PADDING-FACTOR, LOGICAL-FLAG-LENGTH, VALUE-FLAG-LENGTH, PROPAGATE-VALUE-FLAG and the ISAM key and information relating to the ISAM secondary key are also stored. For SAM-Node-Files the character code is also stored on Net-Storage (NETCCSN).

KEEP-TYPES = *DEFAULT / *STD / *ALL
Specifies types of members for which file attributes are to be stored.

KEEP-TYPES = *DEFAULT
The default value is *STD (see below) or the current value set with MODIFY-DEFAULTS.

KEEP-TYPES = *STD
Attributes are stored for members of base types S, M, P, D, J, X.

KEEP-TYPES = *ALL
Attributes are stored for members of all base types that are permissible for ADD-ELEMENT.

i LMSCONV stores the last byte pointer (LBP) in addition to the previously stored file attributes

- for PAM members of type X
- for members of type C if KEEP-TYPES = *ALL has been specified

DELETE-SOURCE = *DEFAULT / *NO / *YES
Here, the user can specify whether the original file is to be retained or deleted. This operand has no effect if the data is read from SYSDTA or *OMF.

DELETE-SOURCE = *DEFAULT
The default value is *NO (see below) or the current value set with MODIFY-DEFAULTS.

DELETE-SOURCE = *NO
The original file will not be deleted.

DELETE-SOURCE = *YES
The original file will be deleted.

WRITE-MODE = *DEFAULT / *CREATE / *REPALCE / *EXTEND / *ANY
Overwriting of a member having the same name. If the member does not exist under this name, it will be created as a new member.

WRITE-MODE = *DEFAULT
The default value is *CREATE (see below) or the current value set with MODIFY-DEFAULTS.

WRITE-MODE = *CREATE

The target member must not yet exist and is created as a new member.

WRITE-MODE = *REPLACE
A member will only be overwritten if a member having the same name is already present. Otherwise ADD-ELEMENT will be rejected with an error message.

WRITE-MODE = *EXTEND
A member will, however, only be extended if no ISAM keys are stored in the member and the file attributes ACCESS-METHOD, RECORD-FORMAT and RECORD-SIZE stored in the member match the attributes of the file. Otherwise ADD-ELEMENT will be rejected with an error message. *EXTEND is not permitted when input is from SYSDTA.

WRITE-MODE = *ANY
The target member is replaced if it already exists. Otherwise it will be created as a new member.

DIALOG-CONTROL = *DEFAULT / *NO / *YES / *ERROR
This operand specifies whether or not the execution of the statement is carried out interactively with the user.

For more detailed information on dialog control, see the MODIFY-DEFAULTS statement.

DIALOG-CONTROL = *DEFAULT
The default value is *NO or the current value set with MODIFY-DEFAULTS..

DIALOG-CONTROL = *NO
All members are processed without dialog queries, i.e. without the user being able to take control.
Exception: If a member or a library is locked, LMSCONV inquires whether the attempt to access it shall be repeated.

DIALOG-CONTROL = *YES / *ERROR
See the description in the MODIFY-DEFAULTS statement .

Notes

- If SOURCE-ATTRIBUTES=*KEEP is specified, the following should be noted: Should any ISAM keys be present, this can impair subsequent processing such as language processing and /CALL-PROCEDURE. This parameter value is particularly suited to archiving.
- When temporary files with wildcards are being added, no constructor specification of the target member name is permitted, i.e. only ELEM=*BY-SOURCE is permitted.
- When temporary files are being added with ELEM=*BY-SOURCE, the member receives the internal file name. This member cannot be output to a file again under another task without explicit specification of a file name.

Example

Adding a member

The file “testmem” is added to library LIB1 under the same name as the member. The type specification must be specified explicitly here in the ADD-ELEMENT statement since the type is preset to *UNDEFINED as standard.

```
/start-lmsconv
//open-library lib1,*update
//add-element from-file=testelem,to-elem=(type=d)
. . .
```

7.4.2.2 CLOSE-LIBRARY - Close library

This statement closes the specified library/libraries. If this statement is specified for a global library, this library is closed and its name is reset.

If the LMSCONV output is held in a library member, then the associated library will not be closed and an error message is issued.

If this statement is specified without parameters, all open libraries are closed and the name of the global library is reset.

Format

CLOSE-LIBRARY

LIBRARY = ***ALL** / ***STD** / <filename 1..54 without-vers> / ***LINK**(...)

***LINK**(...)

| **LINK**-NAME = <structured-name 1..8>

Operands

LIBRARY = *ALL / *STD / <filename 1..54 without-vers> / *LINK (...)

Specifies the library or libraries to be closed.

LIBRARY = *ALL
All open libraries are closed.

LIBRARY = *STD
The library opened by OPEN-LIBRARY is closed.

LIBRARY = <filename 1..54 without-vers>
Name of the library to be closed.

LIBRARY = *LINK(...)
The library assigned via a link name.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of /ADD-FILE-LINK prior to the LMSCONV run.

Examples

All open libraries are closed and the name of the global library is reset.

```
//close-library
```

The library lib1 is closed.

```
//close-library library=lib1
```

The library that was assigned via the link name lib2 is closed.

```
//close-library library=*link(link-name=lib2)
```

7.4.2.3 COPY-ELEMENT - Copy member

COPY-ELEMENT copies members and libraries one to one. The copied members may receive new member designations.

The following copy options are available:

- copying one or more members in the same library
- copying one or more members to a different library
- Copy an entire library

Copying with WRITE-MODE=*SUBSTITUTE

Specifying WRITE-MODE=*SUBSTITUTE makes the copied member the only member in the target library with its type and name. Before copying the member into the target library, LMSCONV deletes all members having the same type and name as the target member.

Restrictions

- The input library must not be the same as the output library.
- If an error occurs during deletion, the statement is aborted.

Format

```
COPY-ELEMENT

ELEMENT = *LIBRARY-ELEMENT (...)

    *LIBRARY-ELEMENT(...)

        | LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)

        |     *LINK(...)

        |         | LINK-NAME = <structured-name 1..8>

        | ,ELEMENT = *ALL(...) / <composed-name 1..64 with-under with-wild(132)>(...)

        |     *ALL(...)

        |         | VERSION = *HIGHEST-EXISTING / *ALL / *UPPER-LIMIT /

        |         |       <composed-name 1..24 with-under with-wild(52)>

        |     <composed-name 1..64 with-under with-wild(132)>(...)

        |         | VERSION = *HIGHEST-EXISTING / *ALL / *UPPER-LIMIT /

        |         |       <composed-name 1..24 with-under with-wild(52)>

        | ,TYPE = *DEFAULT / *ALL / <alphanum-name 1..8 with-wild(20)>

        | ,USER-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)

        |     *INTERVAL(...)

        |         | FROM = 1900-01-01 / <date 8..10 with-compl>

        |         | ,TO = *TODAY / <date 8..10 with-compl>

        | ,CREATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)

        |     *INTERVAL(...)

        |         | FROM = 1900-01-01 / <date 8..10 with-compl>

        |         | ,TO = *TODAY / <date 8..10 with-compl>

        | ,MODIFICATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)

        |     *INTERVAL(...)

        |         | FROM = 1900-01-01 / <date 8..10 with-compl>

        |         | ,TO = *TODAY / <date 8..10 with-compl>
```

```
,TO-ELEMENT = *LIBRARY-ELEMENT (...)

    *LIBRARY-ELEMENT(...)

        | LIBRARY = *STD / *BY-SOURCE / <filename 1..54 without-vers> / *LINK(...)

        |     *LINK(...)

        |         | LINK-NAME = <structured-name 1..8>

        | ,ELEMENT = *BY-SOURCE (...) / <composed-name 1..132 with-under with-wildcard-constr>(...)

        |     *BY-SOURCE(...)

        |         | VERSION = *DEFAULT / *BY-SOURCE / *UPPER-LIMIT /

        |         |     <composed-name 1..52 with-under with-wildcard-constr>

        |     <composed-name 1..132 with-under with-wildcard-constr>(...)

        |         | VERSION = *DEFAULT / *BY-SOURCE / *UPPER-LIMIT /

        |         |     <composed-name 1..52 with-under with-wildcard-constr>

        | ,TYPE = *BY-SOURCE / *DEFAULT / <alphanum-name 1..20 with-wildcard-constr>

        | ,USER-DATE = *BY-SOURCE / *TODAY / <date 8..10 with-compl>

,WRITE-MODE = *DEFAULT / *CREATE / *REPLACE / *EXTEND / *SUBSTITUTE / *ANY

,DIALOG-CONTROL = *DEFAULT / *NO / *YES / *ERROR
```

Operands

ELEMENT = *LIBRARY-ELEMENT(...)

Specifications for the desired member designation.

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library containing the members to be copied.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = <filename 1..54 without-vers>
Name of the library from which the members are to be copied.

LIBRARY = *LINK(...)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library.

ELEMENT = *ALL (...) / <composed-name 1..64 with-under with-wild(132)>(...)
Name of the member to be copied.

VERSION = *HIGHEST-EXISTING / *ALL / *UPPER-LIMIT /
<composed-name 1..24 with-under with-wild(52)>
Version of the member to be copied.

VERSION = *HIGHEST-EXISTING
The member with the highest existing version is copied.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' in the library under the specified TYPE and name is copied.

VERSION = <composed-name 1..24 with-under with-wild(52)>
The text specified here is interpreted as the version designation.

TYPE = *DEFAULT / *ALL / <alphanum-name 1..8 with-wild(20)>
Type of the member to be copied. If the value is *DEFAULT and the current value set with MODIFY-DEFAULTS is *NONE, LMSCONV requires a type specification.

USER-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date given by the user.

USER-DATE = *ANY
The member to be copied has any date.

USER-DATE = *TODAY
The member with the current date is copied.

USER-DATE = <date 8..10 with-compl>
The member whose date is entered explicitly in the form [YY]YY-MM-DD is copied.

USER-DATE = *INTERVAL(...)
All members lying in the specified interval are copied.

FROM = 1900-01-01 / <date 8..10 with-compl>
Beginning of interval.

TO = *TODAY / <date 8..10 with-compl>
End of interval.

CREATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Creation date of the member. For a description of the operands, see the USER-DATE operand of this statement.

MODIFICATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date of the last modification to the member. For a description of the operands, see the USER-DATE operand of this statement.

TO-ELEMENT = *LIBRARY-ELEMENT(...)
Specifies the destination and name under which the member is to be copied.

LIBRARY = *STD / *BY-SOURCE / <filename 1..54 without-vers> / *LINK(...)
Specifies the new library name or library to which the member is to be added.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = *BY-SOURCE
The member is copied to the library which contains the member being copied.

LIBRARY = <filename 1..54 without-vers>
Name of the library to which the file is to be copied as a member. If the library does not yet exist, it will be created.

LIBRARY = *LINK(...)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library.

ELEMENT = *BY-SOURCE(...) /
<composed-name 1..132 with-under with-wildcard-constr>(...)
Name that the new member is to receive.

ELEMENT = *BY-SOURCE(...)
The new name is the same as the old name.

VERSION = *DEFAULT / *BY-SOURCE / *UPPER-LIMIT /
<composed-name 1..52 with-under with-wildcard-constr>
Version that the new member is to receive.

VERSION = *DEFAULT
The default value *BY-SOURCE or the current value set MODIFY-DEFAULTS.

VERSION = *BY-SOURCE
The new member receives the same version as the original member.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' is generated.

VERSION = <composed-name 1..52 with-under with-wildcard-constr>
The new member receives the version specified here.

ELEMENT = <composed-name 1..132 with-under with-wildcard-constr>(...)
Name of the new member to be added. It can also be entered using wildcards.

VERSION = *DEFAULT / *BY-SOURCE / *UPPER-LIMIT /
<composed-name 1..52 with-under with-wildcard-constr>
Version that the new member is to receive.
For a description of the operands, see above.

TYPE = *BY-SOURCE / *DEFAULT /
<alphanum-name 1..20 with-wildcard-constr>
Type that the new member to be added is to receive.

TYPE = *BY-SOURCE
The new member receives the same type designation as the original member.

TYPE = *DEFAULT
If the value is *DEFAULT and the current value set with MODIFY-DEFAULTS is *NONE, LMSCONV requires a type specification.

USER-DATE = *BY-SOURCE / *TODAY / <date 8..10 with-compl>
Date given by the user.

USER-DATE = *BY-SOURCE
The new member is given the same date as the source member.

USER-DATE = *TODAY
The current date is given.

USER-DATE = <date 8..10 with-compl>
The date must be specified in the form [YY]YY-MM-DD.

WRITE-MODE = *DEFAULT / *CREATE / *REPLACE / *EXTEND / *SUBSTITUTE / *ANY
Overwriting of a member having the same name. If the member does not exist under this name, it will be created as a new member.

WRITE-MODE = *DEFAULT
The default value is *CREATE (see below) or the current value set with MODIFY-DEFAULTS.

WRITE-MODE = *CREATE
The target member must not yet exist and is created as a new member.

WRITE-MODE = *REPLACE
The target member must already exist and is replaced.

WRITE-MODE = *EXTEND
The target member is extended if it already exists. Otherwise it will be created as a new member.

WRITE-MODE = *SUBSTITUTE
All members having the same type and name as the source member are deleted from the target library. The source member is then copied into the library.

WRITE-MODE = *ANY
The target member is replaced if it already exists. Otherwise it will be created as a new member.

DIALOG-CONTROL = *DEFAULT / *NO / *YES / *ERROR
This operand specifies whether or not the execution of the statement is carried out interactively with the user.
For more detailed information on dialog control, see the MODIFY-DEFAULTS statement.

DIALOG-CONTROL = *DEFAULT
The default value is *NO or the current value set with MODIFY-DEFAULTS..

DIALOG-CONTROL = *NO
All members are processed without dialog queries, i.e. without the user being able to take control.
Exception: If a member or a library is locked, LMSCONV inquires whether the attempt to access it shall be repeated.

DIALOG-CONTROL = *YES / *ERROR
See the description in the MODIFY-DEFAULTS statement .

Examples

Input library X contains the S-type member A/1. Output library Y contains the S-type member A/2. Following

```
//COPY-ELEM  ELEMENT= *LIB(LIB=X,ELEM=A,TYPE=S),-
             TO-ELEMENT= *LIB(LIB=Y), WRITE-MODE=*SUBSTITUTE
```

member A/1 is then the only member of type S and name A existing in the output library. Member A/2 has been deleted.

All the members of a product version are located in input library X. These members are to be copied into an existing output library Y in such a way that, after the copying process is concluded, Y contains only the copied product version and no other version. This can be done with

```
//COPY-ELEM  ELEMENT= *LIB(LIB= X,ELEM= *,TYPE= *),-
             TO-ELEMENT= *LIB(LIB= Y), WRITE-MODE=*SUBSTITUTE
```

Copy an entire library

Library lib1 is copied in its entirety and is given the name lib2. By specifying “*” for member and type, no knowledge is required of the members contained, i.e. all members are copied one to one to the newly created library lib2.

```
//COPY-ELEM  ELEMENT= *LIB(LIB=lib1,ELEM= *,VERSION= *,TYPE= *),-
             TO-ELEMENT= *LIB(LIB=lib2)
```

7.4.2.4 COPY-LIBRARY - Copy library

The COPY-LIBRARY statement copies a library in its entirety, i.e. together with all its library, type and member attributes. The target library must either have FILE-STRUCTURE=NONE or not yet exist. The target library is given the library format corresponding to its value for BUFFER-LENGTH. The statement is thus suitable for converting a library format.

The file protection attributes of the source library can be taken from the target library. This means that the statement is suited for reorganizing libraries. The target library is logically identical with the original library and now occupies the smallest possible amount of disk space.

If an error occurs during processing of the COPY-LIBRARY statement (e.g. insufficient disk space), the target library is not complete.

Format

COPY-LIBRARY

LIBRARY = <filename 1..54 without-vers> / ***LINK**(...)

***LINK**(...)

| **LINK**-NAME = <structured-name 1..8>

,TO-LIBRARY = <filename 1..54 without-vers> / ***LINK**(...)

***LINK**(...)

| **LINK**-NAME = <structured-name 1..8>

,FILE-**ATTR**IBUTES = ***STD** / ***BY-SOUR**CE

Operands

LIBRARY = <filename 1..54 without-vers> / *LINK(...)
Copies the library with the specified name.

LIBRARY = <filename 1..54 without-vers>
Copies the library assigned by means of the name.

LIBRARY = *LINK(...)
Copies the library assigned by means of a link name.

LINK-NAME = <structured-name 1..8>
Link name of the library, which was declared with /ADD-FILE-LINK.

TO-LIBRARY = <filename 1..54 without-vers> / *LINK(...)
Specifies the target library.

TO-LIBRARY = <filename 1..54 without-vers>
Generates a library with the specified name.

TO-LIBRARY = *LINK(...)
Generates the library assigned by means of a link name.

LINK-NAME = <structured-name 1..8>
Link name of the library, which was declared with /ADD-FILE-LINK.

FILE-ATTRIBUTES = *STD / *BY-SOURCE
Attributes of the target library file.

FILE-ATTRIBUTES = *STD
The file attributes of the target library are not changed. New files will be generated with the default values defined by the file management system.

FILE-ATTRIBUTES = *BY-SOURCE
The file protection attributes of the source library are applied to the target library (analogous to /COPY-FILE .., PROTECTION=*SAME).

Examples

Copy a library to an NK4 pubset

```
/start-lmsconv
//copy-library library=lib,to-library=:nk4:lib
//end
```

Set the initial NK4 library format

```
/add-file-link file-name=nk4lib,link-name=nk4,buffer-length=*std(2)
/start-lmsconv
//copy-library library=nk2lib,to-library=*link(nk4)
//end
```

Reorganize a library with intermediate storage

```
/delete-file file-name=tolib
/start-lmsconv
//copy-library library=lib,to-library=tolib,file-attributes=*by-source
//end
/copy-file from-file=tolib,to-file=lib
/delete-file file-name=tolib
```

7.4.2.5 DELETE-ELEMENT - Logically delete member

The DELETE-ELEMENT statement deletes the specified members in the assigned library (logical deletion). The directory entries are thereby deleted and storage space is released (release is performed only within the library, i.e. the library as a whole does not become smaller).

A member of a library is deleted physically if the DESTROY=*YES parameter has been set or if the member contains a code for physical deletion or if the system parameter DESTLEV requires it.

The statement is executed only if a library has been specified explicitly in the statement or the library specified under OPEN-LIBRARY has been opened with MODE=*UPDAT.

The DELETE-ELEMENT statement is permitted for all member types.

Format

DELETE**-ELEM**ENT

ELEMENT = ***LIB**RARY**-ELEM**ENT (...)

***LIB**RARY**-ELEM**ENT(...)

| **LIB**RARY = ***STD** / <filename 1..54 without-vers> / ***LINK**(...)

| ***LINK**(...)

| | **LINK**-NAME = <structured-name 1..8>

| ,**ELEM**ENT = ***ALL**(...) / <composed-name 1..64 with-under with-wild(132)>(...)

| ***ALL**(...)

| | **VERSION** = ***HIGHEST-EXIST**ING / ***ALL** / ***UP**PER**-LIM**IT /

| | <composed-name 1..24 with-under with-wild(52)>

| <composed-name 1..64 with-under with-wild(132)>(...)

| | **VERSION** = ***HIGHEST-EXIST**ING / ***ALL** / ***UP**PER**-LIM**IT /

| | <composed-name 1..24 with-under with-wild(52)>

| ,**TYPE** = ***DEFAULT** / ***ALL** / <alphanum-name 1..8 with-wild(20)>

| ,**USER-DATE** =***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

| ***INTERVAL**(...)

| | **FROM** = **1900-01-01** / <date 8..10 with-compl>

| | ,**TO** = ***TODAY** / <date 8..10 with-compl>

| ,**CRE**ATION**-DATE** =***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

| ***INTERVAL**(...)

| | **FROM** = **1900-01-01** / <date 8..10 with-compl>

| | ,**TO** = ***TODAY** / <date 8..10 with-compl>

| ,**MOD**IFICATION**-DATE** =***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

| ***INTERVAL**(...)

| | **FROM** = **1900-01-01** / <date 8..10 with-compl>

| | ,**TO** = ***TODAY** / <date 8..10 with-compl>

,**DESTROY-DATA** = ***DEFAULT** / ***NO** / ***Y**ES

,**DIALOG-CONTR**OL = ***DEFAULT** / ***NO** / ***Y**ES

Operands

ELEMENT = *LIBRARY-ELEMENT(...)
Specifies the member to be deleted. Specifications for the desired member designation.

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library from which the member is to be deleted.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = <filename 1..54 without-vers>
Name of the library from which the member is to be deleted.

LIBRARY = *LINK(...)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library.

ELEMENT = *ALL(...) /
<composed-name 1..64 with-under with-wild(132)>(...)
Name of the member to be deleted

ELEMENT = *ALL(...)
All members are deleted.

ELEMENT = <composed-name 1..64 with-under with-wild(132)>(...)
Name of the member to be deleted.

VERSION = *HIGHEST-EXISTING / *ALL / *UPPER-LIMIT /
<composed-name 1..24 with-under with-wild(52)>
Version of the member to be deleted.

VERSION = *HIGHEST-EXISTING
The member with the highest existing version is deleted.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' in the library under the specified TYPE and name is deleted.

VERSION = <composed-name 1..24 with-under with-wild(52)>
Explicitly specifies the version of the member to be deleted.

TYPE = *DEFAULT / *ALL / <alphanum-name 1..8 with-wild(20)>
Type of the member to be deleted.

TYPE = *DEFAULT
If the value is *DEFAULT and the current value set with MODIFY-DEFAULTS is *NONE, LMSCONV requires a type specification.

TYPE = *ALL
All types are deleted

TYPE = <alphanum-name 1..8 with-wild(20)>
Only the specified type is deleted.

USER-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date given by the user.

USER-DATE = *ANY
The member to be deleted has any date.

USER-DATE = *TODAY
The member with the current date is deleted.

USER-DATE = <date 8..10 with-compl>
The member whose date is entered explicitly in the form [YY]YY-MM-DD is deleted.

USER-DATE = *INTERVAL(...)
All members lying in the specified interval are deleted.

FROM = 1900-01-01 / <date 8..10 with-compl>
Beginning of interval.

TO = *TODAY / <date 8..10 with-compl>
End of interval.

CREATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Creation date of the member. For a description of the operands, see the USER-DATE operand of this statement.

MODIFICATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date of the last modification to the member. For a description of the operands, see the USER-DATE operand of this statement.

DESTROY-DATA = *DEFAULT / *NO / *YES
Deletes the data for all members defined by *LIBRARY-ELEMENT.

DESTROY-DATA = *DEFAULT
The default value is *NO (see below) or the current value set with MODIFY-DEFAULTS.

DESTROY-DATA = *NO
A member of a library is physically deleted only if the member is marked for physical deletion or the system parameter DESTLEV requires it.

DESTROY-DATA = *YES
Following logical deletion, the data, if present, is deleted physically, i.e. overwritten with X'00'.

DIALOG-CONTROL = *DEFAULT / *NO / *YES
This operand specifies whether or not the execution of the statement is carried out interactively with the user.

DIALOG-CONTROL = *DEFAULT
The default value is *NO or the current value set with MODIFY-DEFAULTS..

DIALOG-CONTROL = *NO
All members are processed without dialog queries, i.e. without the user being able to take control.
Exception: If a member or a library is locked, LMSCONV inquires whether the attempt to access it shall be repeated.

DIALOG-CONTROL = *YES
LMSCONV prompts for confirmation on each member, e.g. process or skip member, or abort function.

For more detailed information on dialog control, see the MODIFY-DEFAULTS statement, where the value *ERROR which might have been set there has the same effect as *NO. Likewise, the value *ERROR which may have been set for DIALOG-CONTROL= in /INFORM-PROGRAM has the same effect as *NO with DELETE-ELEMENT.

Example

Member TEST3 is deleted from library LIB1.

```
/start-lmsconv
//open-library lib1
//show-element-attributes
INPUT  LIBRARY= :N:$USER.LIB1
TYP  NAME    VER (VAR#) DATE         NAME    VER (VAR#) DATE
(S)  TEST1   @   (0001) <date>       TEST2   @   (0001) <date>
(S)  TEST3   @   (0001) <date>       TEST4   @   (0001) <date>
//delete-element (elem=test3,type=s)
//show-element-attributes
INPUT  LIBRARY= :N:$USER.LIB1
TYP  NAME    VER (VAR#) DATE         NAME    VER (VAR#) DATE
(S)  TEST1   @   (0001) <date>       TEST2   @   (0001) <date>
(S)  TEST4   @   (0001) <date>
. . .
```

7.4.2.6 END - Terminate LMSCONV

END terminates the LMSCONV program. All libraries that are still open are closed.

Format

END

If a monitoring job variable was specified when calling LMSCONV with `/START-LMSCONV MONJV= <name>` this variable is given a value on termination of LMSCONV. The MONJV value is divided into a 3-byte status display and a 4-byte return code display. The return code display is split into a 1-byte termination code (TC) and a 3-byte program information part (PI).

LMSCONV sets the status display and the termination code as follows:

Status display

Status display (byte 1 - 3)	Termination code TC (byte 4)	Program information PI (byte 5 - 7)	Remarks
$T_	0 or 1	see below	Normal termination
$A_	2 or 3	see below	Abnormal termination

Termination code

BC	Explanation
0	Normal termination LMSCONV ran without error.
1	Normal termination Warnings or error occurred that were weaker than the value set with MAX-ERROR-WEIGHT (see MODIFY-DEFAULTS).
2	Abnormal termination An abort criterion set with MAX-ERROR-WEIGHT has been reached(see MODIFY-DEFAULTS).
3	Abnormal termination The error encountered is so serious that it is no longer possible or expedient to continue the LMSCONV session. The LMSCONV session has been terminated internally.

Program information

PI	Explanation
000	BC: 0 LMSCONV ran without error.
001	BC: 1 At most warnings occurred.
002	BC: 1 or 2 Only errors of class RECOVERABLE occurred, i.e. a member could not be found or overwritten.
003	BC: 1 or 2 At most errors of class SIGNIFICANT occurred, i.e. no serious errors.
004	BC: 2 At most errors of class SERIOUS occurred, i.e. serious errors.
005	BC: 3 The error encountered is so serious that it is no longer possible or expedient to continue the LMSCONV session. The LMSCONV session has been terminated internally.

7.4.2.7 EXTRACT-ELEMENT - Output member to file

The EXTRACT-ELEMENT statement outputs members to files. LMSCONV creates the files in accordance with

- the entry in the task file table (TFT), if the file has been assigned via the link name
- the stored file attributes and the FILE-ATTRIBUTES operand
- the catalog entry.

The files can have RECORD-FORMAT=UNDEFINED and arbitrary BUFFER-LENGTH and RECORD-SIZE values. However, the maximum record length of 32 Kbytes (including the record header) must not be exceeded.

If the ISAM keys of an ISAM file have been included in the member, the ISAM keys are also output when EXTRACT-ELEMENT is issued.

If a text member is extracted to an existing PAM file, the (possibly implicit) setting of the operand ACCESS-METHOD determines the new access method (SAM or ISAM) of the output file.

If information on ISAM secondary keys was stored when the file was added, the secondary keys are recreated. If some or all of the secondary keys cannot be recreated, the file is generated without those keys.

The EXTRACT-ELEMENT statement is permissible for member types S, M, R, J, P, D, X, C. Members of type C, and PAM members under type X, are created as PAM files.

The file created contains the CCS name of the source member as its CCS catalog attribute.

> ¡ Valid member names are not always permitted as file names.

Generating ISAM files

When members are output to ISAM files, LMSCONV generates the ISAM keys as follows:

- If the ISAM keys are also added when an ISAM file is included as a library member, LMSCONV generates the ISAM file with those ISAM keys which have been stored.
- If no ISAM keys have been stored in the input member, an ISAM file with KEY-POSITION=5 and KEY-LENGTH=8 is created. LMSCONV then normally generates ISAM keys with an initial value of 1000 and an increment of 1000. If the member is too large for this increment (more than 100,000 records), the increment will be calculated from the number of records.

> ¡
> - R-type members are output up to the END record. Any records which come afterwards are ignored.
> - Correction journal records (TXTP) are not included in the output in the case of C-type members.
> - RECORD-SIZE is supplied with values only with RECORD-FORMAT=FIXED; with RECORD-FORMAT=*VARIABLE, the value is 0.

Format

EXTRACT-ELEMENT

ELEMENT = ***LIB**RARY**-ELEM**ENT (...)

***LIB**RARY**-ELEM**ENT(...)

| **LIB**RARY = ***STD** / <filename 1..54 without-vers> / ***LINK**(...)

| ***LINK**(...)

| | **LINK**-NAME = <structured-name 1..8>

| ,**ELEM**ENT = ***ALL**(...) / <composed-name 1..64 with-under with-wild(132)>(...)

| ***ALL**(...)

| | **VERSION** = ***HIGHEST-EXIST**ING / ***ALL** / ***UP**PER**-LIM**IT /

| | <composed-name 1..24 with-under with-wild(52)>

| <composed-name 1..64 with-under with-wild(132)>(...)

| | **VERSION** = ***HIGHEST-EXIST**ING / ***ALL** / ***UP**PER**-LIM**IT /

| | <composed-name 1..24 with-under with-wild(52)>

| ,**TYPE** = ***DEF**AULT / ***ALL** / <alphanum-name 1..8 with-wild(20)>

| ,**USER-DATE** =***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

| ***INTERVAL**(...)

| | **FROM** = **1900-01-01** / <date 8..10 with-compl>

| | ,**TO** = ***TODAY** / <date 8..10 with-compl>

| ,**CRE**ATION**-DATE** =***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

| ***INTERVAL**(...)

| | **FROM** = **1900-01-01** / <date 8..10 with-compl>

| | ,**TO** = ***TODAY** / <date 8..10 with-compl>

| ,**MOD**IFICATION**-DATE** =***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

| ***INTERVAL**(...)

| | **FROM** = **1900-01-01** / <date 8..10 with-compl>

| | ,**TO** = ***TODAY** / <date 8..10 with-compl>

```
,TO-FILE = *STD / *BY-SOURCE / <filename 1..54 without-gen-vers with-wild-constr> /
                                        *LINK(...)
      *LINK(...)
           |     LINK-NAME = <structured-name 1..8>
,FILE-ATTRIBUTES = *BY-ELEMENT / *BY-CATALOG / *DEFAULT / *PARAMETERS(...)
      *PARAMETERS(...)
           |     ACCESS-METHOD = *DEFAULT / *ISAM / *SAM
,WRITE-MODE = *DEFAULT / *CREATE / *REPLACE / *EXTEND / *ANY
,DIALOG-CONTROL = *DEFAULT / *NO / *YES / *ERROR
```

Operands

ELEMENT = *LIBRARY-ELEMENT(...)
Specifications for the desired member designation.

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library containing the member.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = <filename 1..54 without-vers>
Name of the library containing the member.

LIBRARY = *LINK(...)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library.

ELEMENT = *ALL(...) / <composed-name 1..64 with-under with-wild(132)>(...)
Name of the members to be extracted from the library.

ELEMENT = *ALL(...)
All members are taken from the library.

ELEMENT = <composed-name 1..64 with-under with-wild(132)>(...)
Name of the member to be extracted from the library and included in a file.

**VERSION = *HIGHEST-EXISTING / *ALL / *UPPER-LIMIT /
<composed-name 1..24 with-under with-wild(52)>**
Version of the member to be output.

VERSION = *HIGHEST-EXISTING
The member with the highest existing version is used.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' in the library under the specified TYPE and name is output.

VERSION = <composed-name 1..24 with-under with-wild(52)>
Explicitly specifies the version of the member to be output.

TYPE = *DEFAULT / *ALL / <alphanum-name 1..8 with-wild(20)>
Type of the member to be adopted.

TYPE = *DEFAULT
If the value is *DEFAULT and the current value set with MODIFY-DEFAULTS is *NONE, LMSCONV requires a type specification.

TYPE = *ALL
All types are adopted.

TYPE = <alphanum-name 1..8 with-wild(20)>
Only the specified type is adopted.

USER-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date given by the user.

USER-DATE = *ANY
The member to be output has any date.

USER-DATE = *TODAY
The member with the current date is selected.

USER-DATE = <date 8..10 with-compl>
The member whose date is entered explicitly in the form [YY]YY-MM-DD is chosen.

USER-DATE = *INTERVAL(...)
All members lying in the specified interval are chosen.

FROM = 1900-01-01 / <date 8..10 with-compl>
Beginning of interval.

TO = *TODAY / <date 8..10 with-compl>
End of interval.

CREATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Creation date of the member. For a description of the operands, see the USER-DATE operand of this statement.

MODIFICATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date of the last modification to the member. For a description of the operands, see the USER-DATE operand of this statement.

TO-FILE = *STD / *BY-SOURCE / <filename 1..54 without-gen-vers with-wild-constr> / *LINK(...)
Name of the target file.

TO-FILE = *STD
Unless otherwise specified, the member data is output to a file which is given the same name as the member.

TO-FILE = *BY-SOURCE
The file name is the same as the member name.

TO-FILE = <filename 1..54 without-gen-vers with-wild-constr>
Name of the target file. A design specification refers to the member name.

TO-FILE = *LINK(...)
The member is output to the file that was assigned via the link name.

LINK-NAME = <structured-name 1..8>
File link name.

FILE-ATTRIBUTES = *BY-ELEMENT / *BY-CATALOG / *DEFAULT / *PARAMETERS(...)
File attributes specified when the file is created. LMSCONV defines the file attributes in accordance with the following hierarchy:

- LINK entry
- file attributes stored in the member
- catalog entry
- LMSCONV default values.

The following specifications take effect only when TO-FILE=*LINK has not been specified.

FILE-ATTRIBUTES = *BY-ELEMENT
The file attributes stored in the member take priority.

FILE-ATTRIBUTES = *BY-CATALOG
The attributes stored in the catalog entry take priority. If there is no catalog entry, specifying *BY-CATALOG has the same effect as *BY-ELEMENT.

> The file can also be created as a SAM node file on Net-Storage with the NETCCSN that has been stored as a file attribute.

FILE-ATTRIBUTES = *PARAMETERS(...)

ACCESS-METHOD = *DEFAULT / *ISAM / *SAM
Specifies the access method ISAM or SAM for the target file. The default value is *ISAM or the current value set with MODIFY-DEFAULTS.

WRITE-MODE = *DEFAULT / *CREATE / *REPLACE / *EXTEND / *ANY

Overwriting of a file having the same name. If the file does not exist under this name, it will be created as a new file.

WRITE-MODE = *DEFAULT
The default value is *CREATE (see below) or the current value set with MODIFY-DEFAULTS.

WRITE-MODE = *CREATE
The new file must not yet exist and is created as a new file.

WRITE-MODE = *REPLACE
The file must already exist and is replaced.

WRITE-MODE = *EXTEND
The file is extended if it already exists. Otherwise it will be created as a new file.

WRITE-MODE = *ANY
The file is replaced if it already exists. Otherwise it will be created as a new file.

DIALOG-CONTROL = *DEFAULT / *NO / *YES / *ERROR
This operand specifies whether or not the execution of the statement is carried out interactively with the user.
For more detailed information on dialog control, see the MODIFY-DEFAULTS statement.

DIALOG-CONTROL = *DEFAULT
The default value is *NO or the current value set with MODIFY-DEFAULTS..

DIALOG-CONTROL = *NO
All members are processed without dialog queries, i.e. without the user being able to take control.
Exception: If a member or a library is locked, LMSCONV inquires whether the attempt to access it shall be repeated.

DIALOG-CONTROL = *YES / *ERROR
See the description in the MODIFY-DEFAULTS statement.

Note

Extracting an element into a SAM node file on Net-Storage can fail because the CCSN of the element cannot be converted into the NETCCSN of the node file. This can have several cause:

1. An inappropriate CCSN is assigned to the element. In this case a suitable CCSN can be assigned with // MODIFY-ELEMENT-ATTRIBUTES.
2. An unsuitable NETCCSN is stored in the file attributes for the element (can be seen in the output of // SHOW-ELEMENT TEXT-INFORMATION = * FILE-ATTRIBUTES). In this case, the node file into which the element is to be extracted must be assigned the appropriate NETCCSN with / MODIFY-FILE-ATTRIBUTES. The element can then be extracted by specifying FILE-ATTRIBUTES = * BY-CATALOG.
3. No NETCCSN is saved for the element (e.g. because it was not originally a node file or because the SOURCE-ATTRIBUTES = * KEEP operand was not specified when the node file was added to the library), and the user's preset NETCCSN does not match CCSN. In general, the user attribute NET-CODED-CHARACTER-SET should be changed accordingly in this case. In individual cases, however, the same procedure can be used as in case 2.

Examples

Member ELEM1 is output by EXTRACT-ELEMENT to file TEST having the specified file attributes.

```
/add-file-link file-name=test,link-name=out,access-method=*sam, -
/              record-format=*variable
/start-lmsconv
//open-library library=libin
//extract-element (,elem1,s),*link(link-name=out)
//end
```

If all the members in a library are to be output by name, the following statement must be specified:

```
//extract-element (elem=*all,type =*all)
```

Note

It is not possible to modify the element's NETCCSN. That means that after changing the CCSN and extracting a member the resulting SAM node file may eventually not be edited if the required code conversion cannot be executed. In this case, the correct code table should be set using the MODIFY-FILE-ATTRIBUTES command.

7.4.2.8 MODIFY-DEFAULTS - Modify defaults

The MODIFY-DEFAULTS statement can be used to modify the default values. If an explicit value is used locally in an LMSCONV statement, this value has priority over the default value.

The reference to the values set here in the LMSCONV statements is the *DEFAULT specification.

At the beginning of the LMSCONV run, the values immediately following *UNCHANGED apply. If one of these values is changed using the MODIFY-DEFAULTS statement, the new setting becomes the current setting. This setting remains valid for the LMSCONV session (*UNCHANGED) until a new MODIFY-DEFAULTS statement is issued for this value.

Format

MODIFY-**DEFAULT**S

ELEMENT-**ATTR**IBUTES = ***UNCHANGED** / ***PAR**AMETERS(...)

***PAR**AMETERS(...)

| **TYPE** = ***UNCHANGED** / ***NONE** / <alphanum-name 1..8>

| ,**ELEM**ENT-**VERSION** = ***UNCHANGED** / ***ALL** / ***HIGH**EST-**EXIST**ING

| ,**TO-ELEM**ENT-**VERSION** = ***UNCHANGED** / ***BY-SOUR**CE

| ,**SOURC**E-**ATTR**IBUTES = ***UNCHANGED** / ***STD** / ***IGNORE** / ***KEEP(...)**

| ***KEEP**(...)

| | **KEEP-TYPES** = ***UNCHAN**GED / ***STD** / ***ALL**

,**F**ILE-**ATTR**IBUTES = ***UNCHANGED** / ***PAR**AMETERS(...)

***PAR**AMETERS(...)

| **ACCESS-METHOD** = ***UNCHANGED** / ***ISAM** / ***SAM**

,**DESTROY-DATA** = ***UNCHANGED** / ***NO** / ***Y**ES / ***BY-SOUR**CE

,**WR**ITE-**MODE** = ***UNCHANGED** / ***CREATE** / ***REPLACE** / ***EXTEND** / ***ANY**

,**DIALOG-CONTR**OL = ***UNCHANGED** / ***NO** / ***Y**ES / ***ERROR**

,**INF**ORMATION = ***UNCHANGED** / ***MED**IUM / ***MIN**IMUM / ***MAX**IMUM / ***SUMM**ARY

,**LAYOUT** = ***UNCHANGED** / ***VAR**IABLE / ***FIXED**

,**SORT** = ***UNCHANGED** / ***BY-NAME** / ***BY-VERS**ION / ***BY-USER-DATE** / ***BY-SEC**ONDARY-**NAME**

,**OUT**PUT**-FORM** = ***UNCHANGED** / ***STD** / ***CHAR**ACTER / ***HEX**ADECIMAL / ***DUMP**

,**DEL**ETE**-SOUR**CE = ***UNCHANGED** / ***NO** / ***Y**ES

,**MAX-ERROR-WEIGHT** = ***UNCHANGED** / ***SERIOUS** / ***SIGNIFICANT** / ***RECOV**ERABLE

,**RUN-MODE** = ***UNCHANGED** / ***STD** / ***BATCH**

,**NEXT-ATTEM**PT = ***UNCHANGED** / ***NO** / ***Y**ES(...)

***Y**ES(...)

| **NUM**BER**-OF-ATTEM**PTS = ***UNCHANGED** / <integer 1..2147483647>

| ,**PER**IOD = ***UNCHANGED** / <integer 1..21599>

,**TEXT-INF**ORMATION = ***UNCHANGED** / ***ALL** / ***STATIS**TICS / ***F**ILE**-ATTR**IBUTES / ***PAR**AMETERS(...)

***PAR**AMETERS(...)

| **INF**ORMATION = ***UNCHANGED** / ***ALL** / list-poss(2): ***TEXT** / ***COMMENT**

| ,**REC**ORD**-RANGE** = ***UNCHANGED** / ***ALL** / ***RANGE**(...)

| ***RANGE**(...)

| | **FROM** = ***UNCHANGED** / <integer 1..2147483647>

| | ,**TO** = ***UNCHANGED** / ***LAST** / <integer 1..2147483647>

| ,**REC**ORD**-PART** = ***UNCHANGED** / ***ALL** / ***PART**(...)

| ***PART**(...)

| | **START** = ***UNCHANGED** / <integer 1..32764>

| | ,**LEN**GTH = ***UNCHANGED** / ***REST** / <integer 1..32764>

| ,**REC**ORD**-NUM**BER = ***UNCHANGED** / ***BY-OUT**PUT / ***Y**ES / ***NO**

,**MOD**ULE**-INF**ORMATION = ***UNCHANGED** / ***ALL** / ***STATIS**TICS / ***PAR**AMETERS(...)

***PAR**AMETERS(...)

| **INF**ORMATION = ***UNCHANGED** / ***ALL** / ***TXT**(...) / ***TXTP**(...) / list-poss(9): ***ESD** / ***ISD** /

| ***LSD** / ***RLD** / ***REP** / ***INCL**UDE / ***DSDD** / ***REF** / ***END**

| ***TXT**(...)

| | **CSECT**-NAME = ***UNCHANGED** / ***ALL** / <c-string 1..32 with-low> / <text 1..32>

| | ,**ADDR**ESS = ***UNCHANGED** (...) / <x-string 1..8>(...)

| | ***UNCHANGED**(...)

| | | **BASE-ADDR**ESS = ***UNCHANGED** / <x-string 1..8>

| | <x-string>(...)

```
|          |          |     BASE-ADDRESS = *UNCHANGED / <x-string 1..8>
|          |    ,LENGTH = *UNCHANGED / *REST / <integer 1..2147483647> / <x-string 1..8>
```

```
        |       *TXTP(...)
        |           | MODIFICATION-ID = *UNCHANGED / *ALL / <c-string 1..8 with-low> / *RANGE(...)
        |           |       *RANGE(...)
        |           |           | FROM = *UNCHANGED / *LOWEST / <c-string 1..8 with-low>
        |           |           | ,TO = *UNCHANGED / *HIGHEST / <c-string 1..8 with-low>
,PHASE-INFORMATION = *UNCHANGED / *ALL / *STATISTICS / *PARAMETERS(...)
    *PARAMETERS(...)
        | SEGMENT = *UNCHANGED / *ALL / *ROOT / <name 1..8>
        | ,INFORMATION = *UNCHANGED / *ALL / *TXT(...) / *TXTP(...) / list-poss(4): *ESD / *ISD /
        |                                 *LSD / *RLD
        |       *TXT(...)
        |           | ADDRESS = *UNCHANGED (...) / <x-string 1..8>(...)
        |           |       *UNCHANGED(...)
        |           |           | BASE-ADDRESS = *UNCHANGED / <x-string 1..8>
        |           |       <x-string>(...)
        |           |           | BASE-ADDRESS = *UNCHANGED / <x-string 1..8>
        |           | ,LENGTH = *UNCHANGED / *REST / <integer 1..2147483647> / <x-string 1..8>
        |       *TXTP(...)
        |           | MODIFICATION-ID = *UNCHANGED / *ALL / <c-string 1..8 with-low> / *RANGE(...)
        |           |       *RANGE(...)
        |           |           | FROM = *UNCHANGED / *LOWEST <c-string 1..8 with-low>
        |           |           | ,TO = *UNCHANGED / *HIGHEST / <c-string 1..8 with-low>
,LLM-INFORMATION = *UNCHANGED / *ALL / *STATISTICS / *PARAMETERS(...)
    *PARAMETERS(...)
        | LLM-PART = *UNCHANGED / *ALL / *SLICE(...) / *SUB-LLM(...)
        |       *SLICE(...)
        |           | NAME = *UNCHANGED / <structured-name 1..32>
        |       *SUB-LLM(...)
        |           | PATH-NAME = *UNCHANGED / <c-string 1..255 with-low> / <text 1..255>
        | ,INFORMATION = *UNCHANGED / *ALL / *TXT(...) / *TXTP(...) / *LOGICAL(...) / *PHYSICAL / *REF /
        |                                 list-poss(4): *RELOCATION / *ESVD / *ESVR / *LRLD
```

```
|  *TXT(...)
|     |  CSECT-NAME = *UNCHANGED / *ALL / <c-string 1..32 with-low> / <text 1..32>
|     |  ,ADDRESS = *UNCHANGED (...) / <x-string 1..8>(...)
|     |     *UNCHANGED(...)
|     |        |  BASE-ADDRESS = *UNCHANGED / <x-string 1..8>
|     |     <x-string>(...)
|     |        |  BASE-ADDRESS = *UNCHANGED / <x-string 1..8>
|     |  ,LENGTH = *UNCHANGED / *REST / <integer 1..2147483647> / <x-string 1..8>
|  *TXTP(...)
|     |  CSECT-NAME = *UNCHANGED / *ALL / <c-string 1..32 with-low> / <text 1..32>
|     |  ,MODIFICATION-ID = *UNCHANGED / *ALL / <c-string 1..12 with-low> / *RANGE(...)
|     |     *RANGE(...)
|     |        |  FROM = *UNCHANGED / *LOWEST / <c-string 1..12 with-low>
|     |        |  ,TO = *UNCHANGED / *HIGHEST / <c-string 1..12 with-low>
|  *LOGICAL(...)
|     |  LEVEL = *UNCHANGED / *ALL / *NEXT
```

Operands

ELEMENT-ATTRIBUTES = *UNCHANGED / *PARAMETERS(...)
Specifies the member type, the member version, the storage form and the file attributes.

TYPE = *UNCHANGED / *NONE / <alphanum-name 1..8>
Specifies the member type.

TYPE = *NONE
No global member type is defined, i.e. the types must be specified locally in the statements.

TYPE = <alphanum-name 1..8>
The name specified here is used as the type in the statement.

ELEMENT-VERSION = *UNCHANGED / *ALL / *HIGHEST-EXISTING
Specifies the member version for SHOW-ELEMENT-ATTRIBUTES.

ELEMENT-VERSION = *ALL
All versions for a member are output.

ELEMENT-VERSION = *HIGHEST-EXISTING
Only the highest version of a member is output.

TO-ELEMENT-VERSION = *UNCHANGED / *BY-SOURCE
Specifies the version of a target members.

TO-ELEMENT-VERSION = *BY-SOURCE
The target member contains the same version as the source member.

SOURCE-ATTRIBUTES = *UNCHANGED / *STD / *IGNORE / *KEEP(...)
Only the ADD-ELEMENT statement is interpreted.

SOURCE-ATTRIBUTES = *STD
No file attributes are stored, and neither is an ISAM key. In this case, it is only possible to add ISAM files with KEY-POSITION=5, KEY-LENGTH 16 and RECORD-FORMAT=*VARIABLE to the member.
For ISAM files a warning is issued that the ISAM key has not been added.

SOURCE-ATTRIBUTES = *IGNORE
As for SOURCE-ATTRIBUTES=*STD, except that no warning is issued.

SOURCE-ATTRIBUTES = *KEEP(...)
The following file attributes are stored unchanged in the new member to be added: ACCESS-METHOD, RECORD-FORMAT, RECORD-SIZE, BUFFER-LENGTH, PERFORMANCE, USAGE, ACCESS and USER-ACCESS. If ACCESS-METHOD=ISAM is specified, the PADDING-FACTOR, LOGICAL-FLAG-LENGTH, VALUE-FLAG-LENGTH, PROPAGATE-VALUE-FLAG and the ISAM key and information relating to the ISAM secondary key are also stored. For SAM-Node-Files the character code is also stored on Net-Storage (NETCCSN).

KEEP-TYPES = *UNCHANGED / *STD / *ALL
Specifies types of members for which file attributes are to be stored.

KEEP-TYPES = *STD
Attributes are stored for members of base types S, M, P, D, J, X.

KEEP-TYPES = *ALL
Attributes are stored for members of all base types that are permissible for ADD-ELEMENT.

i For SAM node files LMSCONV stores the coded character set on Net-Storage (NETCCSN) as an element attribute. LMSCONV stores the last byte pointer (LBP) in addition to the previously stored file attributes

- for PAM members of type X
- for members of type C if KEEP-TYPES = *ALL has been specified

FILE-ATTRIBUTES = *UNCHANGED / *PARAMETERS(...)
File attributes specified when the file is created.

ACCESS-METHOD = *UNCHANGED / *ISAM / *SAM
Specifies the file access method.

ACCESS-METHOD = *ISAM
Create ISAM file.

ACCESS-METHOD = *SAM
Create SAM file.

DESTROY-DATA = *UNCHANGED / *NO / *YES / *BY-SOURCE
Defines whether or not the data is physically deleted, i.e. overwritten with X'00'.

DESTROY-DATA = *NO
A member of a library is physically deleted only if the member is marked for physical deletion or the system parameter DESTLEV requires it.

DESTROY-DATA = *YES
Any data present is physically deleted.

DESTROY-DATA = *BY-SOURCE
The flag for overwriting the data is taken from the source member or source file and assigned to the target member or target file. If the source is missing, *BY-SOURCE has the same effect as DESTROY-DATA=*NO.

WRITE-MODE = *UNCHANGED / *CREATE / *REPLACE / *EXTEND / *ANY
Overwriting of a member having the same name. If the member does not exist under this name, it will be created as a new member.

If the selected value is not possible for WRITE-MODE in the local statement, the setting WRITE-MODE= *CREATE applies.

WRITE-MODE = *CREATE
The target member must not yet exist and is created as a new member.

WRITE-MODE = *REPLACE
The target member must already exist and is replaced.

WRITE-MODE = *EXTEND
The target member is extended if it already exists. Otherwise it will be created as a new member.

WRITE-MODE = *ANY
The target member is replaced if it already exists. Otherwise it will be created as a new member.

DIALOG-CONTROL = *UNCHANGED / *NO / *YES / *ERROR
This operand specifies whether or not the execution of the statement is carried out interactively with the user. (This operand is not applicable in procedure or batch mode.)

DIALOG-CONTROL = *NO
All members are processed without query.

DIALOG-CONTROL = *YES
LMSCONV prompts for confirmation on each member, e.g. process or skip member, or abort function.

DIALOG-CONTROL = *ERROR
If a fatal error occurs during member processing, e.g. overwriting a member, the user is prompted to specify how LMSCONV should react.

The user now has the following options:

YES	Process the member.
NO	Do not process the member.
ALL	Abort the statements without prompting.
TERMINATE	Terminate the statement.

By pressing the K2 key and entering “/INFORM-PROGRAM”, the user can modify the value of the DIALOG-CONTROL operand. If LMSCONV is in the middle of member processing, the user can influence the further processing with:

`/INFORM-PROGRAM '[N-I / N-E / C][,DIALOG-CONTROL=*NO / *YES / *ERROR]'`

`/INFORM-PROGRAM 'NEXT-INPUT (N-I)'`: the current statement is aborted;
LMSCONV reads a further statement as soon as it becomes active again. An unrecognized or missing text has the same effect as NEXT-INPUT.

`/INFORM-PROGRAM 'NEXT-ELEMENT (N-E)'`: the processing of the current member in the current statement is aborted; LMSCONV continues processing with the next member (if available). If no next member is present, NEXT-ELEMENT has the same effect as NEXT-INPUT.

`/SEND-MSG 'CONTINUE (C)'`:LMSCONV processing is continued as normal

C,NO	The member is to be processed, but interactive mode not activated.
N-E/N-I,NO	Member or statement processing is to be aborted, but interactive mode not activated.
C,YES/ERROR	The member is to be processed and interactive mode then activated.
N-E/N-I,YES/ERROR	Member or statement processing is to be aborted and interactive mode then activated.

ℹ The value set for `DIALOG-CONTROL` with `/`INFORM-PROGRAM applies only to the current statement.

INFORMATION = *UNCHANGED / *MEDIUM / *MINIMUM / *MAXIMUM / *SUMMARY
This parameter defines the scope of directory information to be output.

INFORMATION = *MEDIUM
Outputs the type, name, version, variant number and the date or the member size.

INFORMATION = *MINIMUM
Outputs only the type, name and version.

INFORMATION = *MAXIMUM
In addition to the type, name, version, variant number and the user date, the existing member protection, the storage form, the assigned character set, as well as the user date, creation date and modification date and time are logged for all members.

INFORMATION = *SUMMARY
Outputs only the number of members per type.

LAYOUT = *UNCHANGED / *VARIABLE / *FIXED
This parameter defines the format of the directory to be output.

LAYOUT = *VARIABLE
The number of print columns depends on the longest member designation within a member type.

LAYOUT = *FIXED
The directory is printed in a single column in fixed format. Single column means that the entries in the directory appear one beneath the other.

SORT = *UNCHANGED / *BY-NAME / *BY-VERSION / *BY-USER-DATE / *BY-SECONDARY-NAME
Sort criterion for the directory entries of the selected members. The type is always used as the first sort criterion.

SORT = *BY-NAME
The directory entries of the selected members are sorted on the basis of the following sequence: type, name and version.

SORT = *BY-VERSION
The directory entries of the selected members are sorted on the basis of the following sequence: type, version and name.

SORT = *BY-USER-DATE
The directory entries of the selected members are sorted on the basis of the following sequence: type, user date, name and version.

SORT = *BY-SECONDARY-NAME
The directory entries of the selected members are sorted on the basis of the following sequence: type, secondary name, secondary attribute, name and version.

For more detailed information, refer to the SHOW-ELEMENT-ATTRIBUTES statement.

OUTPUT-FORM = *UNCHANGED / *STD / *CHARACTER / *HEXADEZIMAL / *DUMP
Specifies the display format for the output.

OUTPUT-FORM = *STD
The form of representation is selected dependent on the member type.

OUTPUT-FORM = *CHARACTER
The output is in alphanumeric form.

OUTPUT-FORM = *HEXADECIMAL
The output is in alphanumeric and hexadecimal form, one above the other.

OUTPUT-FORM = *DUMP
The output is in alphanumeric and hexadecimal form, side by side. For member types S, P, D, J and M, this operand has the same effect as OUTPUT-FORM=*HEXADECIMAL.

DELETE-SOURCE = *UNCHANGED / *NO / *YES
Here, the user can specify whether the original file is to be retained (default value *NO) or deleted (parameter *YES). This operand has no effect when read by *OMF.

MAX-ERROR-WEIGHT = *UNCHANGED / *SERIOUS / *SIGNIFICANT / *RECOVERABLE
This operand defines the cases in which LMSCONV is to trigger the spin-off mechanism.

MAX-ERROR-WEIGHT = *SERIOUS
The spin-off mechanism is triggered for serious errors, i.e. errors for which processing of the statement is no longer possible or useful.

MAX-ERROR-WEIGHT = *SIGNIFICANT
The spin-off mechanism is triggered for *SERIOUS and also for other errors (except when the member could not be found or overwritten).

MAX-ERROR-WEIGHT = *RECOVERABLE
The spin-off mechanism is triggered for all errors.

RUN-MODE = *UNCHANGED / *STD / *BATCH
When LMSCONV is running in interactive mode, this operand determines whether LMSCONV should continue to run normally or behave as if it is running in batch mode. This operand has no effect in procedures or in batch mode.

RUN-MODE = *STD
LMSCONV shall continue to run normally.

RUN-MODE = *BATCH
LMSCONV shall behave like running in batch mode.

NEXT-ATTEMPT = *UNCHANGED / *NO / *YES(...)
Controls the open attempts for file, type or member lock in procedure or batch mode.

NEXT-ATTEMPT = *NO
No further attempts to open are made.

NEXT-ATTEMPT = *YES(...)
Further attempts to open are made.

NUMBER-OF-ATTEMPTS = *UNCHANGED / <integer 1..2147483647>
Number of further attempts (default: 9 attempts).

PERIOD = *UNCHANGED / <integer 1..21599>
Wait period between attempts in seconds (default: 6 seconds).

TEXT-INFORMATION = *UNCHANGED / *ALL / *STATISTICS / *FILE-ATTRIBUTES / *PARAMATERS(...)
Defines the information scope for all members except member types R, C and L. For PAM members, all values have the same effect as *ALL, except for *FILEATTRIBUTES.

TEXT-INFORMATION = *ALL
Everything is output.

TEXT-INFORMATION = *STATISTICS
The number of records per record type and the total number of records are output.

TEXT-INFORMATION = *FILE-ATTRIBUTES
Only the stored file attributes are output.

TEXT-INFORMATION = *PARAMETERS(...)
Defines a member extract to be output.

INFORMATION = *UNCHANGED / *ALL / list-poss(2): *TEXT / *COMMENT
The member section to be output.

INFORMATION = *ALL
Outputs all user record types.

INFORMATION = *TEXT
Outputs the text itself, i.e. record type 1.

INFORMATION = *COMMENT
Outputs the separately stored comment, i.e. record type 2.

RECORD-RANGE = *UNCHANGED / *ALL / *RANGE(...)
The section of the member to be processed.

RECORD-RANGE = *ALL
Processes all user record types.

RECORD-RANGE = *RANGE(...)
Specifies the range of record numbers which is to be processed. The record numbers refer not to a record type, but to the section of the member designated by means of INFORMATION=. Within this section, the records are numbered consecutively from
1 through n.

FROM = *UNCHANGED / <integer 1..2147483647>
Beginning of the range, specified by the first record number. Record number 1 is the default value.

TO = *UNCHANGED / *LAST / <integer 1..2147483647>
End of the range, specified by the last record number. The last record number is used as the default value.

RECORD-PART = *UNCHANGED / *ALL / *PART(...)
Specifies the part of the record to be processed.

RECORD-PART = *ALL
Processes the entire record.

RECORD-PART = *PART(...)
Specifies the part of the record to be processed. If the default values are not changed, the entire record is processed.

START = *UNCHANGED / <integer 1..32764>
Beginning of the record part, specified by the first character in the record. The first character is used as the default value.

LENGTH = *UNCHANGED / *REST / <integer 1..32764>
Length of the record part. The remainder of the record is used as the default value.

RECORD-NUMBER = *UNCHANGED / *BY-OUTPUT / *YES / *NO
Specifies output of the record numbers.

RECORD-NUMBER = *BY-OUTPUT
Only if the output is directed to SYSOUT will no record numbers be output. With any other output medium, the record numbers are included in the output.

RECORD-NUMBER = *YES
The record numbers are also output to SYSOUT.

RECORD-NUMBER = *NO
No record numbers are included in the output..

MODULE-INFORMATION = *UNCHANGED / *ALL / *STATISTICS / *PARAMETERS(...)
Defines the information scope for object modules (R-type members).

MODULE-INFORMATION = *ALL
Everything is output.

MODULE-INFORMATION = *STATISTICS
The name, length and address of the CSECTS and also the overall length of the module are output.

MODULE-INFORMATION = *PARAMETERS(...)
This parameter determines whether all record types or only selected record types are output.

INFORMATION = *UNCHANGED / *ALL / *TXT(...) / *TXTP(...) / list-poss(9): *ESD / *ISD / *LSD / *RLD / *REP / *INCLUDE / *DSDD / *REF / *END
The record types listed here can be selected.

INFORMATION = *TXT(...)
Text records are selected.

CSECT-NAME = *UNCHANGED / *ALL / <c-string 1..32 with-low> / <text 1..32>
The text records can be restricted to one CSECT.

ADDRESS = *UNCHANGED(...) / <x-string 1..8>(...)
Start address of the text.

BASE-ADDRESS = *UNCHANGED / <x-string 1..8>
The base address specified here is added to the start address.

LENGTH = *UNCHANGED / *REST / <integer 1..2147483647> / <x-string 1..8>
Length of the text.

INFORMATION = *TXTP(...)
TXTP records are output.

MODIFICATION-ID = *UNCHANGED / *ALL / <c-string 1..8 with-low> /*RANGE(...)
Those TXTP records with the specified identification are selected.

MODIFICATION-ID = *RANGE(...)
A group of TXTP records lying in a range can be selected.

FROM = *UNCHANGED / *LOWEST / <c-string 1..8 with-low>
The beginning of the range is by default the lowest identification for the TXTP records, otherwise the value entered here.

TO = *UNCHANGED / *HIGHEST / <c-string 1..8 with-low>
The end of the range is by default the highest identification for the TXTP records, otherwise the value entered here.

PHASE-INFORMATION = *UNCHANGED / *ALL / *STATISTICS / *PARAMETERS(...)
Defines the information scope for phases (C-type members).

PHASE-INFORMATION = *ALL
Everything is output.

PHASE-INFORMATION = *STATISTICS
The name, length and address of the segment and also the overall length of the segment are output.

PHASE-INFORMATION = *PARAMETERS(...)
This parameter determines whether all record types or only selected record types are output.

SEGMENT = *UNCHANGED / *ALL / *ROOT / <name 1..8>
Phase segment that is selected.

INFORMATION = *UNCHANGED / *ALL / *TXT(...) / *TXTP(...) / list-poss(4): *ESD / *ISD / *LSD / *RLD
The record types listed here can be selected.

INFORMATION = *TXT(...)
Text records are selected.

ADDRESS = *UNCHANGED (...) / <x-string 1..8>(...)
Start address of the text.

BASE-ADDRESS = *UNCHANGED / <x-string 1..8>
The base address specified here is added to the start address.

LENGTH = *UNCHANGED / *REST / <integer 1..2147483647> / <x-string 1..8>
Length of the text.

INFORMATION = *TXTP(...)
TXTP records are output.

MODIFICATION-ID = *UNCHANGED / *ALL / <c-string 1..8 with-low> /*RANGE(...)
Those TXTP records with the specified identification are selected.

MODIFICATION-ID = *RANGE(...)
A group of TXTP records lying in a range can be selected.

FROM = *UNCHANGED / *LOWEST / <c-string 1..8 with-low>
The beginning of the range is by default the lowest identification for the TXTP records, otherwise the value entered here.

TO = *UNCHANGED / *HIGHEST / <c-string 1..8 with-low>
The end of the range is by default the highest identification for the TXTP records, otherwise the value entered here.

LLM-INFORMATION = *UNCHANGED / *ALL / *STATISTICS / PARAMETERS(...)
Defines the information scope for link and load modules (L-type members).

LLM-INFORMATION = *ALL
Everything is output.

LLM-INFORMATION = *STATISTICS
General information on the link and load module (name, copyright, ...) is output.

LLM-INFORMATION = *PARAMETERS(...)
This parameter determines whether all record types or only selected record types are output.

LLM-PART = *UNCHANGED / *ALL / *SLICE(...) / *SUB-LLM(...)
Specifies the LLM part to be selected. By default the entire LLM is selected.

LLM-PART = *SLICE(...)
Specifies the slice to be output.

NAME = *UNCHANGED / <structured-name 1..32>
Name of the slice to be output.

LLM-PART = *SUB-LLM(...)
Specifies the sub-LLM to be output.

PATH-NAME = *UNCHANGED / <c-string 1..255 with-low> / <text 1..255>
The sub-LLM to be output is determined by way of its path name.

INFORMATION = *UNCHANGED / *ALL / *TXT(...) / *TXTP(...) / *LOGICAL(...) / *PHYSICAL / *REF / list-poss(3): *ESVD / *ESVR / *LRLD / *RELOCATION
The record types listed here can be selected.

INFORMATION = *TXT(...)
Text records are selected.

CSECT-NAME = *UNCHANGED / *ALL / <c-string 1..32 with-low> / <text 1..32>The text records can be restricted to one CSECT.

ADDRESS = *UNCHANGED(...) / <x-string 1..8>(...)
Start address of the text.

BASE-ADDRESS = *UNCHANGED / <x-string 1..8>
The base address specified here is added to the start address.

LENGTH = *UNCHANGED / *REST / <integer 1..2147483647> / <x-string 1..8>Length of the text.

INFORMATION = *TXTP(...)
TXTP records are output.

CSECT-NAME = *UNCHANGED / *ALL / <c-string 1..32 with-low> / <text 1..32>
The TXTP records can be restricted to one CSECT.

MODIFICATION-ID = *UNCHANGED / *ALL / <c-string 1..12 with-low> / *RANGE(...)
Those TXTP records with the specified identification are selected.

MODIFICATION-ID = *RANGE(...)
A group of TXTP records lying in a range can be selected.

FROM = *UNCHANGED / *LOWEST / <c-string 1..12 with-low>
The beginning of the range is by default the lowest identification for the TXTP records, otherwise the value entered here.

TO = *UNCHANGED / *HIGHEST / <c-string 1..12 with-low>
The end of the range is by default the highest identification for the TXTP records, otherwise the value entered here.

INFORMATION = *LOGICAL(...)
Outputs the logical structure of the LLM.

LEVEL= *UNCHANGED / *ALL / *NEXT
Outputs all substructures by default; otherwise, only the next substructure.

INFORMATION = *PHYSICAL
Outputs the physical structure of the LLM.

Example

The file TEST1 is to be added as member of type D in the library LIB3.

The default setting for the member type is changed to the desired value using the MODIFY-DEFAULTS statement. Thus, it is no longer necessary to specify the member type in the subsequent ADD-ELEMENT statement. To ensure that LMSCONV can confirm successful addition of the files, the MODIFY-LOGGING-PARAMETERS statement sets the scope of the log information to the complete LMSCONV log.

```
/start-lmsconv
//open-library lib3,*update
//modify-defaults type=d
//modify-log-param logging=*maximum
//add-element test1
INPUT  FILE
OUTPUT LIBRARY= :N:$USER.LIB3
    ADD TEST1   AS (D)TEST1/ (0001)/<date>
. . .
```

7.4.2.9 MODIFY-ELEMENT - Modify member

The MODIFY-ELEMENT statement initiates the modification of members. The modifications themselves are controlled by way of substatements.

MODIFY-ELEMENT selects the members to be modified.

Once the MODIFY-ELEMENT statement has been sent, LMSCONV expects a substatement as the next statement. If another statement is entered instead of a substatement, an error message is issued.

Format

MODIFY**-ELEM**ENT

ELEMENT = ***LIB**RARY**-ELEM**ENT (...)

***LIB**RARY**-ELEM**ENT(...)

| **LIB**RARY = ***STD** / <filename 1..54 without-vers> / ***LINK**(...)

| ***LINK**(...)

| | **LINK**-NAME = <structured-name 1..8>

| ,**ELEM**ENT = ***ALL**(...) / <composed-name 1..64 with-under with-wild(132)>(...)

| ***ALL**(...)

| | **VERSION** = ***HIGHEST-EXIST**ING / ***ALL** / ***UP**PER**-LIM**IT /

| | <composed-name 1..24 with-under with-wild(52)>

| <composed-name>(...)

| | **VERSION** = ***HIGHEST-EXIST**ING / ***ALL** / ***UP**PER**-LIM**IT /

| | <composed-name 1..24 with-under with-wild(52)>

| ,**TYPE** = ***DEF**AULT / <alphanum-name 1..8>

| ,**USER-DATE** =***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

| ***INTERVAL**(...)

| | **FROM** = **1900-01-01** / <date 8..10 with-compl>

| | ,**TO** = ***TODAY** / <date 8..10 with-compl>

```
        | CREATION-DATE =*ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
        |     *INTERVAL(...)
        |         | FROM = 1900-01-01 / <date 8..10 with-compl>
        |         | ,TO = *TODAY / <date 8..10 with-compl>
        | ,MODIFICATION-DATE =*ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
        |     *INTERVAL(...)
        |         | FROM = 1900-01-01 / <date 8..10 with-compl>
        |         | ,TO = *TODAY / <date 8..10 with-compl>
,TO-ELEMENT = *LIBRARY-ELEMENT (...)
    *LIBRARY-ELEMENT(...)
        | LIBRARY = *STD / *BY-SOURCE / <filename 1..54 without-vers> / *LINK(...)
        |     *LINK(...)
        |         | LINK-NAME = <structured-name 1..8>
        | ,ELEMENT = *BY-SOURCE (...) / <composed-name 1..132 with-under with-wildcard-constr>(...)
        |     *BY-SOURCE(...)
        |         | VERSION = *DEFAULT / *BY-SOURCE / *UPPER-LIMIT /
        |         |     <composed-name 1..52 with-under with-wildcard-constr>
        |     <composed-name>(...)
        |         | VERSION = *DEFAULT / *BY-SOURCE / *UPPER-LIMIT /
        |         |     <composed-name 1..52 with-under with-wildcard-constr>
        | ,TYPE = *BY-SOURCE / *DEFAULT / <alphanum-name 1..8>
        | ,USER-DATE = *TODAY / *BY-SOURCE / <date 8..10 with-compl>
,TEXT-PARAMETERS = *NONE / *PARAMETERS(...)
    *PARAMETERS(...)
        | INPUT-RECORD-ID = *NONE / *RECORD-PART(...)
        |     *RECORD-PART(...)
        |         | START = <integer 1..251>
        |         | ,LENGTH = <integer 1..16>
,WRITE-MODE = *DEFAULT / *CREATE / *REPLACE / *ANY
,DIALOG-CONTROL = *DEFAULT / *NO / *YES / *ERROR
```

Operands

ELEMENT = *LIBRARY-ELEMENT(...)Specifications for the desired member designation.

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library containing the member to be modified.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = <filename 1..54 without-vers>
Name of the library containing the member to be modified.

LIBRARY = *LINK(...)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library.

ELEMENT = *ALL(...) / <composed-name 1..64 with-under with-wild(132)>(...)

ELEMENT = *ALL(...)
All members are specified.

ELEMENT = <composed-name 1..64 with-under with-wild(132)>(...)
Name of the member to be modified.

VERSION = *HIGHEST-EXISTING / *ALL/ *UPPER-LIMIT /
<composed-name 1..24 with-under with-wild(52)>
Version of the member to be modified.

VERSION = *HIGHEST-EXISTING
The member with the highest existing version is modified.

VERSION = *ALL
All versions of the member are modified.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' in the library under the specified TYPE and name is modified.

VERSION = <composed-name 1..24 with-under with-wild(52)>
Explicitly specifies the version of the member to be modified.

TYPE = *DEFAULT / <alphanum-name 1..8>
Type of the member to be modified. If the value is *DEFAULT and the current value set with MODIFY-DEFAULTS is *NONE, LMSCONV requires a type specification.

USER-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date given by the user.

USER-DATE = *ANY
The member to be modified has any date.

USER-DATE = *TODAY
The member with the current date is modified.

USER-DATE = <date 8..10 with-compl>
The member whose date is entered explicitly in the form [YY]YY-MM-DD is modified.

USER-DATE = *INTERVAL(...)
All members lying in the specified interval are modified.

FROM = 1900-01-01 / <date 8..10 with-compl>
Beginning of interval.

TO = *TODAY / <date 8..10 with-compl>
End of interval.

CREATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Creation date of the member. For a description of the operands, see the USER-DATE operand of this statement.

MODIFICATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date of the last modification to the member. For a description of the operands, see the USER-DATE operand of this statement.

TO-ELEMENT = *LIBRARY-ELEMENT(...)
Specifies the destination to which and the name under which the corrected member is to be written back.

LIBRARY = *STD / *BY-SOURCE / <filename 1..54 without-vers> / *LINK(...)
Specifies the library to which the corrected member is to be written back.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = *BY-SOURCE
The corrected member is written back to the original library.

LIBRARY = <filename 1..54 without-vers>
Name of the library to which the corrected member is to be added.

LIBRARY = *LINK(...)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library.

ELEMENT = *BY-SOURCE(...) / <composed-name 1..132 with-under with-wildcard-constr>(...)
Name that the corrected member is to receive.

ELEMENT = *BY-SOURCE(...)
The new name is the same as the old name.

VERSION = *DEFAULT / *BY-SOURCE / *UPPER-LIMIT / <composed-name 1..52 with-under with-wildcard-constr>
Version that the corrected member is to receive.

VERSION = *DEFAULT
The default value is *BY-SOURCE (see below) or the current value set with MODIFY-DEFAULTS.

VERSION = *BY-SOURCE
The corrected member receives the same version as the original member. If the original member has no version specification, the corrected member receives X'FF' as the version specification.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' is generated.

VERSION = <composed-name 1..52 with-under with-wildcard-constr>
The text specified here is interpreted as the version designation.

ELEMENT = <composed-name 1..132 with-under with-wildcard-constr>(...)
Name of the corrected member. It can also be entered using wildcards.

VERSION = *DEFAULT / *BY-SOURCE / *UPPER-LIMIT /
<composed-name 1..52 with-under with-wildcard-constr>
Version that the corrected member is to receive. For a description of the operands, see above.

TYPE = *BY-SOURCE / *DEFAULT / <alphanum-name 1..8>
Type that the corrected member is to receive.

TYPE = *BY-SOURCE
The corrected member receives the same type designation as the original member.

TYPE = *DEFAULT
If the value is *DEFAULT and the current value set with MODIFY-DEFAULTS is *NONE, LMSCONV requires a type specification.

USER-DATE = *TODAY / *BY-SOURCE / <date 8..10 with-compl>
Date given by the user.

USER-DATE = *TODAY
The current date is given.

USER-DATE = *BY-SOURCE
The new member is given the same date as the source member.

USER-DATE = <date 8..10 with-compl>
The date must be specified in the form [YY]YY-MM-DD.

TEXT-PARAMETERS = *NONE / *PARAMETERS(...)
Specifies parameters for textual members.

TEXT-PARAMETERS = *NONE
No parameters are specified for textual members.

TEXT-PARAMETERS = *PARAMETERS(...)
Specifies parameters for textual members.

INPUT-RECORD-ID = *NONE / *RECORD-PART(...)
Specifies the location of the record ID (see section “Substatements of MODIFY-ELEMENT for text members”) in the input record.

INPUT-RECORD-ID = *NONE
No location is specified for the record ID of the input record.

INPUT-RECORD-ID = *RECORD-PART(...)
Specifies the beginning and length of the record ID area, where beginning + length 252.

START = <integer 1..251>
Specifies the first character in the record ID to indicate the beginning of the record ID area.

LENGTH = <integer 1..16>
Specifies the length of the record ID.

WRITE-MODE = *DEFAULT / *CREATE / *REPLACE / *ANY
Overwriting of a member having the same name. If the member does not exist under this name, it will be created as a new member. If the source member is the same as the target member, this operand is ignored.

WRITE-MODE = *DEFAULT
The default value is *CREATE (see below) or the current value set with MODIFY-DEFAULTS.

WRITE-MODE = *CREATE
The name of the corrected member must not yet exist and is created as a new member.

WRITE-MODE = *REPLACE
The corrected member must already exist and is replaced.

WRITE-MODE = *ANY
The corrected member is replaced if it already exists. Otherwise it will be created as a new member.

DIALOG-CONTROL = *DEFAULT / *NO / *YES / *ERROR
This operand specifies whether or not the execution of the statement is carried out interactively with the user.

For more detailed information on dialog control, see the MODIFY-DEFAULTS statement.

DIALOG-CONTROL = *DEFAULT
The default value is *NO or the current value set with MODIFY-DEFAULTS.

DIALOG-CONTROL = *NO
All members are processed without dialog queries, i.e. without the user being able to take control.
Exception: If a member or a library is locked, LMSCONV inquires whether the attempt to access it shall be repeated.

DIALOG-CONTROL = *YES / *ERROR
See the description in the MODIFY-DEFAULTS statement .

7.4.3 Substatements of MODIFY-ELEMENT for member types R, C and L

They are read from the statement stream until the END-MODIFY substatement is encountered.

The substatements make modifications to object modules, phases and link and load modules.

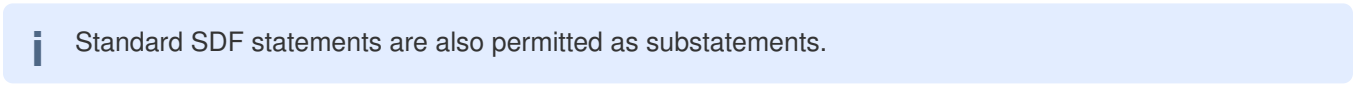
LMSCONV initially collects these substatements and executes them only after the END-MODIFY substatement has been entered.

Overview of LMSCONV substatements

These substatements are dependent on the selected member type and are permitted only for members of types R, C or L.

Statement	Member type	Function
ADD-REP-RECORD	R	Generate REP records
ADD-TEXT-MODIFICATION	R,C,L	Modify text records
DELETE-RECORD-TYPE	R,C,L	Delete record types
END-MODIFY	R,C,L	End modification
MODIFY-CSECT-ATTRIBUTES	R	Modify CSET attributes
MODIFY-MODIFICATION-DEFAULTS	R,C,L	Define global parameters in the MODIFY-ELEMENT statement
REMOVE-MODIFICATION	R,C,L	Cancel corrections
RENAME-SYMBOLS	R	Rename CSECTs, ENTRYs, EXTRNs and COMMONs

Table 9: MODIFY-ELEMENT substatements



7.4.3.1 ADD-REP-RECORD - Add REP records to object module

The ADD-REP-RECORD substatement adds REP records to the object module. These REP records are evaluated by the dynamic binder loader (DBL).

ADD-REP-RECORD substatement is permitted only for object modules (R-type members).

Format

ADD-REP-RECORD

ADDRESS = <x-string 1..8>(...)

<x-string 1..8>(...)

| **BASE-ADDR**ESS = ***MOD**IFICATION-**DEF**AULT / <x-string 1..8>

,**NEW-CONTEN**TS = <x-string 1..100> / <c-string 1..50 with-low>

Operands

ADDRESS = <x-string 1..8>(...)
Specifies the address at which the member selected by MODIFY-ELEMENT is to be modified.

BASE-ADDRESS = *MODIFICATION-DEFAULT / <x-string 1..8>
Base address.
BASE-ADDRESS is added to ADDRESS. The resulting correction address must, in the case of prelinked modules, relate to the prelinked module (and not to the CSECT). The default BASE-ADDRESS is 0.

NEW-CONTENTS = <x-string 1..100> / <c-string 1..50 with-low>
Replacement text specified in character or hexadecimal form.
If the text is specified in character form, it must not exceed 50 characters in length. An apostrophe in the text must be duplicated.
If the text is specified in hexadecimal form, it must not exceed 100 characters in length.

7.4.3.2 ADD-TEXT-MODIFICATION - Correct text records of an object module

The MODIFY-ELEMENT substatement ADD-TEXT-MODIFICATION corrects text records of an object module and phases. This substatement generates a correction journal record (TXTP record) that contains the original contents of the text area.

With the MODIFY-MODIFICATION-DEFAULTS statement, you can specify that no correction journal record is to be created. Corrections without a correction journal record cannot be reversed by the REMOVE-MODIFICATION substatement.

This substatement may be used only for members of types R, C and L.

Format

ADD-TEXT-MODIFICATION

ADDRESS = <x-string 1..8>(...)

<x-string 1..8>(...)

| **BASE-ADD**RESS = ***MOD** IFICATION **-DEF** AULT / <x-string 1..8>

,**NEW-CONTEN**TS = <x-string 1..100>(...) / <c-string 1..50 with-low>(...)

<x-string 1..100>(...)

| **OLD-CONTEN**TS = ***ANY** / <x-string 1..100> / <c-string 1..50 with-low>

<c-string 1..50 with-low>(...)

| **OLD-CONTEN**TS = ***ANY** / <x-string 1..100> / <c-string 1..50 with-low>

,**MOD**IFICATION**-ID** = ***MOD** IFICATION **-DEF** AULT / ***SPACES** / <c-string 1..12 with-low>

Operands

ADDRESS = <x-string 1..8>(...)
Specifies the address at which the member selected by MODIFY-ELEMENT is to be modified.

BASE-ADDRESS = *MODIFICATION-DEFAULT / <x-string 1..8>
Base address. The default BASE-ADDRESS is 0.
The base address is added to ADDRESS. The resulting correction address is as follows:

For ...	Relative to:
modules	start of CSECT (the desired CSECT is specified via the MODIFY-MODIFICATION-DEFAULTS substatement.)
phases	start of phase
LLMs	start of CSECT if a CSECT has been specified
	start of sub-LLM if a sub-LLM has been specified (the desired sub-LLM is specified using the MODIFY-MODIFICATION-DEFAULTS substatement)

	start of slice if a slice has been specified (the desired slice is specified using the MODIFY-MODIFICATION-DEFAULTS substatement)
	start of LLM if nothing has been specified and the LLM only consists of one slice. If the LLM consists of more than one slice, you must specify a CSECT, a sub-LLM or a slice.

Table 10: Determining the base address

NEW-CONTENTS = <x-string 1..100>(...) / <c-string 1..50 with-low>(...)
Replacement text specified in character or hexadecimal form.

OLD-CONTENTS = *ANY / <x-string 1..100> / <c-string 1..50 with-low>
Original text of the member. The original text must always be specified with the same length as the replacement text.

OLD-CONTENTS = *ANY
Any old contents are replaced.

MODIFICATION-ID = *MODIFICATION-DEFAULT / *SPACES / <c-string 1..12 with-low>
Identification which is held in the correction journal record (TXTP record). If SPACES is specified, blanks are used as the identification. The preset value are blanks or the value set with MODIFY-MODIFICATION-DEFAULTS.

For member types R and C, only 8 characters are allowed.
The identification for types R and C is to have the form 'Annnnnnn', and the form 'Annnnnnn-jjj' for type L, where Annnnnnn is a problem report number and jjj is a Julian date.

7.4.3.3 DELETE-RECORD-TYPE - Exclude record types from input member

The MODIFY-ELEMENT substatement DELETE-RECORD-TYPE excludes the following record types from the input member:

- ISD records (applies only to R-type members)
- LSD records (applies only to R-type members)
- REP records (applies only to R-type members)
- INCLUDE records (applies only to R-type members)
- TXTP records (applies only to R-, C- and L-type members)
- DSDD records (applies only to R-type members)

This substatement may be used only for members of types R, C and L.

Format

DELETE**-REC**ORD**-**
TYPE

TYPE = ***TXTP**(...) / list-poss(5): ***ISD** / ***LSD** / ***REP** / ***DSDD** / ***INCL**UDE

***TXTP**(...)

| **MOD**IFICATION**-ID** = **<u>*ALL</u>** / ***SPACES** / <c-string 1..12 with-low>

Operands

TYPE = *TXTP(...) / list-poss(5): *ISD / *LSD / *REP / *DSDD / *INCLUDE
Defines the record type that is not to be transferred from the input member to the output member.

MODIFICATION-ID = <u>*ALL</u> / *SPACES / <c-string 1..12 with-low>
Only those TXTP records with the specified identification are deleted.
For member types R and C, only 8 characters are allowed.
This identification applies only to this DELETE-RECORD-TYPE.

ℹ Deleted record types cannot be retrieved.

7.4.3.4 END-MODIFY - Terminate input of substatements

Each sequence of substatements is concluded by an END-MODIFY substatement. LMSCONV then checks that all statements are executable and executes the statement string.

Format

END-MODIFY

This statement has no operands.

7.4.3.5 MODIFY-CSECT-ATTRIBUTES - Modify CSECT attributes

The MODIFY-ELEMENT substatement MODIFY-CSECT-ATTRIBUTES modifies CSECT attributes.

This substatement must be used for object modules (R-type members) only.

At the beginning of the MODIFY-ELEMENT statement, the operands are preset to the value immediately following *UNCHANGED.

Format

MODIFY-**CSECT-ATTR**IBUTES
NAME = ***ALL** / <c-string 1..8 with-low> / <text 1..8> ,**VISIBLE** = ***UNCHANGED** / ***Y**ES / ***NO** ,**READ-ONLY** = ***UNCHANGED** / ***Y**ES / ***NO** ,**PAGE-ALIGNMENT** = ***UNCHANGED** / ***Y**ES / ***NO** ,**RESIDENCY-MODE** = ***UNCHANGED** / 24 / ***ANY** ,**ADDRESSING-MODE** = ***UNCHANGED** / 24 / 31 / ***ANY**

Operands

NAME = *ALL / <c-string 1..8 with-low> / <text 1..8>
Name of the CSECT whose attributes are to be modified. All CSECTs or a specific CSECT can be specified.

VISIBLE = *UNCHANGED / *YES / *NO
Masking (visibility) of the program interface. The preset value is *YES.

VISIBLE = *YES
The specified control sections are not masked (see the “Binder Loader / Starter” manual [13 (Related publications)]). A secondary name record is created for these sections, and the names are entered in the directory of secondary names.

VISIBLE = *NO
The specified control sections are masked. No secondary name record is created for these sections, and the names are not entered in the directory of secondary names. Any secondary name record which may exist is deleted.

If all control sections of an object module are masked, a library member without a secondary name entry is created. This object module can be located via the primary name only.

The module name can, however, be derived from the initial control section name with the aid of all ESD records, since masked control sections are also used in this case.

> The linkage editor cannot process object modules which only have masked control sections, e.g. when an object module is excluded with the autolink function. The VISIBLE operand can also be used on ENTRYs.

READ-ONLY = *UNCHANGED / *YES / *NO
Specifies the write protection. The preset value is *YES.

READ-ONLY = *YES
Indicates that only read access to the specified control sections is permitted while the program is executing.

READ-ONLY = *NO
Enables write access to the specified control sections even while the program is executing.

PAGE-ALIGNMENT = *UNCHANGED / *YES / *NO
Specifies the page alignment. The preset value is *YES.

PAGE-ALIGNMENT = *YES
Indicates that the specified control sections are to be aligned on a page boundary, i.e. the load address should be a multiple of decimal 4096 or hexadecimal 1000.

PAGE-ALIGNMENT = *NO
Does not take page boundaries into account. The control sections always start at the next doubleword address produced during the linkage process.

RESIDENCY-MODE = *UNCHANGED / 24 / *ANY
Specifies the residency mode. The preset value is 24.

RESIDENCY-MODE = 24
Indicates that the specified control sections are to be loaded to the address area below the 16-Mbyte limit.

RESIDENCY-MODE = *ANY
No limitation exists.

ADDRESSING-MODE = *UNCHANGED / 24 / 31 / *ANY
Specifies the addressing mode. The preset value is 24.

ADDRESSING-MODE = 24
Indicates that the specified control sections are to be executable in 24-bit mode.

ADDRESSING-MODE = 31
Indicates that the specified control sections are to be executable in 31-bit mode.

ADDRESSING-MODE = *ANY
Any execution mode.

7.4.3.6 MODIFY-MODIFICATION-DEFAULTS - Specify global defaults

The MODIFY-ELEMENT substatement MODIFY-MODIFICATION-DEFAULTS defines the global default values within the MODIFY-ELEMENT statement.

This substatement may be used only for members of types R, C and L.

At the beginning of the MODIFY-ELEMENT statement, the operands are preset to the value immediately following *UNCHANGED.

Format

MODIFY**-MOD**IFICATION**-DEF**AULTS

CSECT-NAME = ***UNCHANGED** / ***NONE** / <c-string 1..32 with-low> / <text 1..32>

,**PHASE-SEG**MENT = ***UNCHANGED** / ***ROOT** / <name 1..8>

,**LLM-PART** = ***UNCHANGED** / ***NONE** / ***SLICE**(...) / ***SUB-LLM**(...)

***SLICE**(...)

| **NAME** = <structured-name 1..32>

***SUB-LLM**(...)

| **PATH-NAME** = <c-string 1..255 with-low> / <text 1..255>

,**MOD**IFICATION**-LOG**GING = ***UNCHANGED** / ***Y**ES(...) / ***NO**

***Y**ES(...)

| **MOD**IFICATION**-ID** = ***UNCHANGED** / ***SPACES** / <c-string 1..12 with-low>

,**BASE-ADD**RESS = ***UNCHANGED** / <x-string 1..8>

Operands

CSECT-NAME = *UNCHANGED / *NONE / <c-string 1..32 with-low> / <text 1..32>
Name of the CSECT to be corrected. (Relevant only for types R and L.)
The preset value is *NONE.

CSECT-NAME = *NONE
If no CSECT name is specified, in the case of R-type modules the first CSECT name is used.

PHASE-SEGMENT = *UNCHANGED / *ROOT / <name 1..8>
Specifies the phase segment to be corrected. If no segment is specified, the first segment (*ROOT) is used. The preset value is *ROOT.

LLM-PART = *UNCHANGED / *NONE / *SLICE(...) / *SUB-LLM(...)
If no LLM part is specified, the entire LLM is used. The preset value is *NONE.

LLM-PART = *SLICE(...)
Specifies the slice to be corrected.

NAME = <structured-name 1..32>
Name of the slice to be corrected.

LLM-PART = *SUB-LLM(...)
Specifies the sub-LLM to be corrected.

PATH-NAME = <c-string 1..255 with-low> / <text 1..255>
The sub-LLM to be corrected is determined by way of its path name.

MODIFICATION-LOGGING = *UNCHANGED / *YES(...) / *NO
Defines TXTP record generation. The preset value is *YES.

MODIFICATION-LOGGING = *YES(...)
TXTP records are to be generated.

MODIFICATION-ID = *UNCHANGED / *SPACES / <c-string 1..12 with-low>
Identification which is held in the correction journal record (TXTP record). If SPACES is specified, blanks are used as the identification. The preset value is *SPACES.

For member types R and C, only 8 characters are allowed.
The identification for types R and C is to have the form 'Annnnnnn', and the form 'Annnnnnn-jjj for type L, where Annnnnnn is a problem report number and jjj a Julian date.

MODIFICATION-LOGGING = *NO
No TXTP records are to be generated.

BASE-ADDRESS = *UNCHANGED / <x-string 1..8>
Hexadecimal specification of the base address. At the beginning of the MODIFY-ELEMENT statement, base address 0 is set.

7.4.3.7 REMOVE-MODIFICATION - Cancel corrections

The MODIFY-ELEMENT substatement REMOVE-MODIFICATION cancels corrections from a previous correction run under the following preconditions:

A correction journal record was created with the MODIFY-ELEMENT substatement ADD-TEXT-MODIFICATION, i. e. the operand MODIFICATION-LOGGING=*YES was set.

This substatement may be used only for members of types R, C and L.

Format

REMOVE**-MOD**IFICATION
MODIFICATION-ID = ***ALL** / ***SPACES** / <c-string 1..12 with-low>

Operands

MODIFICATION-ID = *ALL / *SPACES / <c-string 1..12 with-low>
For member types R and C, only 8 characters are allowed.
Only those corrections with the specified identification are canceled. If an identification is specified, correction journal records for it must exist.

MODIFICATION-ID = *ALL
If no identification is specified, all corrections for which a correction journal record exists are canceled.

7.4.3.8 RENAME-SYMBOLS - Rename symbols

The MODIFY-ELEMENT substatement RENAME-SYMBOLS changes the name of a CSECT, ENTRY, EXTRN or COMMON. Each renaming results in a modification of the ESD records. LMSCONV checks for the uniqueness of names within all ESD records, rejecting a new name if that name already exists.

The MODIFY-ELEMENT substatement RENAME-SYMBOLS may be used only for object modules (R-type members).

Format

RENAME**-SYMB**OLS
SYMBOL-NAME = <text 1..8> **,SYMBOL-TYPE** = ***CSECT** / ***ENTRY** / ***EXTRN** / ***COMMON** **,NEW-NAME** = <text 1..8>

Operands

SYMBOL-NAME = <text 1..8>
Defines the symbol name to be renamed.

SYMBOL-TYPE = *CSECT / *ENTRY / *EXTERN / *COMMON
Defines the type of symbol whose name is to be changed.

NEW-NAME = <text 1..8>
New symbol name.
The name should satisfy the BINDER conventions for the special data type <symbol> (see the “Binder” manual [13 (Related publications)]). LMSCONV does not check for this convention, however.

> Masked (invisible) CSECT/ENTRY names can also be renamed.

7.4.4 Substatements of MODIFY-ELEMENT for text members

These substatements make changes to text members. They are read from the statement stream until the END-MODIFY substatement is encountered.

Overview of LMSCONV substatements

Statement	Function
ADD-RECORD	Add records
END-MODIFY	End modification
REMOVE-RECORD	Remove records

Table 11: MODIFY-ELEMENT substatement for text members

> Standard SDF statements are also permitted as substatements.
> Only member records with lengths <= 251 are processed. Longer records will be truncated. In this case, LMSCONV issues a warning.

Definition of record identification for textual members

Record identification may be a record number or a record ID.

Record no.: The record number indicates the relative position of the member record in relation to the beginning of the member. If the record number specified is greater than the member's highest record number, the changes continue after the last member record, i.e. records are appended to the member.

Record ID: The location and length of the record ID are specified by means of the INPUT-RECORD-ID operand (see the MODIFY-ELEMENT statement).
This is why it is not permissible to specify a record ID in substatements except when INPUT-RECORD-ID has a value other than *NONE. If specified, a record ID must have the length declared in INPUT-RECORD-ID. Only leading zeros may be omitted. If the record ID does not occur in the input member, the changes are inserted in front of the first record with a higher record ID.

Record numbers and record IDs may be mixed within substatements. In substatements and data records, they must always be specified in ascending order.

If an error is detected in interactive mode, the correction must be terminated with END-MODIFY and then restarted. After an ADD-RECORD substatement, an *END must also be entered.

7.4.4.1 ADD-RECORD - Add records

The ADD-RECORD substatement inserts the records following the statement at the specified position. The records to be inserted must be concluded by an *END record.

Format

ADD-RECORD
RECORD**-ID** = **<u>*NONE</u>** / <integer 0..99999999> / <c-string 1..16 with-low>

Operands

RECORD-ID = <u>*NONE</u> / <integer 0..99999999> / <c-string 1..16 with-low>
Specifies the data record to be added.

RECORD-ID = <u>*NONE</u>
If the INPUT-RECORD-ID operand of the MODIFY-ELEMENT statement is set to a value other than *NONE, the data records following the ADD-RECORD substatement are inserted in the member being modified in accordance with their record IDs.
If a specified record ID designates a record which already exists, the data record is written over the existing record. If no record with the specified record ID yet exists, the data record is inserted in front of the first record with a higher record ID. If INPUT-RECORD-ID=*NONE is set or no record ID is specified for a data record, the record is inserted at the current position.

RECORD-ID = <integer 0..99999999> / <c-string 1..16 with-low>
Specifies the member position after which the data records following the statement are to be inserted. If the specified record number or record ID does not exist, the data records are each inserted in front of the first record with a higher record number/record ID.

7.4.4.2 END-MODIFY - Conclude substatements

Each sequence of substatements is concluded by an END-MODIFY substatement.

Format

END-MODIFY

This statement has no operands.

7.4.4.3 REMOVE-RECORD - Delete record or record area in member

The REMOVE-RECORD substatement deletes the specified record or record area from the member.

Format

REMOVE**-REC**ORD

RECORD**-ID** = <integer 0..99999999> / <c-string 1..16 with-low> / ***RANGE**(...)

***RANGE**(...)

| **FROM** = <integer 0..99999999> / <c-string 1..16 with-low>

| ,**TO** = <integer 0..99999999> / <c-string 1..16 with-low>

Operands

RECORD-ID = <integer 0..99999999> / <c-string 1..16 with-low> / *RANGE(...)
Record number or record ID of the record to be deleted.

RECORD-ID = <integer 0..99999999> / <c-string 1..16 with-low>
Record number or record ID of the record to be deleted.

RECORD-ID = *RANGE(...)
Specifies the record area to be deleted.

FROM = <integer 0..99999999> / <c-string 1..16 with-low>
Specifies the first record number or record ID of the area which is to be deleted.

TO = <integer 0..99999999> / <c-string 1..16 with-low>
Specifies the last record number or record ID of the area which is to be deleted.

7.4.5 LMSCONV statements MODIFY-ELEMENT-ATTRIBUTES to WRITE- COMMENT

- MODIFY-ELEMENT-ATTRIBUTES - Modify member attributes
- MODIFY-LOGGING-PARAMETERS - Modify logging settings
- OPEN-LIBRARY - Open global library
- SHOW-DEFAULTS - Output current default values
- SHOW-ELEMENT - Display contents of member
- SHOW-ELEMENT-ATTRIBUTES - Display member attributes
- SHOW-LIBRARY-ATTRIBUTES - Display library attributes
- SHOW-LIBRARY-STATUS - Display library status
- SHOW-LOGGING-PARAMETERS - Display global LMSCONV parameters
- SHOW-TYPE-ATTRIBUTES - Display attributes of a member type
- SHOW-USER-EXITS - Display LMSCONV version
- WRITE-COMMENT - Write comments to output medium

7.4.5.1 MODIFY-ELEMENT-ATTRIBUTES - Modify member attributes

The MODIFY-ELEMENT-ATTRIBUTES statement modifies the Coded Character Set Name (CCSN) for a member.

Format

MODIFY-**ELEM**ENT-**ATTR**IBUTES

ELEMENT = ***LIBRARY-ELEM**ENT(...)

***LIB**RARY-**ELEM**ENT(...)

| **LIB**RARY = ***STD** / <filename 1..54 without-vers> / ***LINK**(...)

| ***LINK**(...)

| | **LINK**-NAME = <structured-name 1..8>

| ,**ELEM**ENT = ***ALL**(...) / <composed-name 1..64 with-under with-wild(132)>(...)

| ***ALL**(...)

| | **VERSION** = ***HIGHEST-EXIST** ING / ***ALL** / ***UP**PER-**LIM**IT /

| | <composed-name 1..24 with-under with-wild(52)>

| <composed-name 1..64 with-under with-wild(132)>(...)

| | **VERSION** = ***HIGHEST-EXIST** ING / ***ALL** / ***UP**PER-**LIM**IT /

| | <composed-name 1..24 with-under with-wild(52)>

| ,**TYPE** = ***DEF**AULT / ***ALL** / <alphanum-name 1..8 with-wild(20)>

| ,**USER-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

| ***INTERVAL**(...)

| | **FROM** = **1900-01-01** / <date 8..10 with-compl>

| | ,**TO** = ***TODAY** / <date 8..10 with-compl>

| ,**CRE**ATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

| ***INTERVAL**(...)

| | **FROM** = **1900-01-01** / <date 8..10 with-compl>

| | ,**TO** = ***TODAY** / <date 8..10 with-compl>

| ,**MOD**IFICATION-**DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)

| ***INTERVAL**(...)

| | **FROM** = **1900-01-01** / <date 8..10 with-compl>

| | ,**TO** = ***TODAY** / <date 8..10 with-compl>

| ,**CODED-CHAR**ACTER-**SET** = ***ANY** / ***NONE** / <name 1..8 with-wild(20)>

,**NEW-ATTR**IBUTES = ***PAR**AMETERS(...)

***PAR**AMETERS(...)

| **CODED-CHAR**ACTER-**SET** = ***BY-SOUR**CE / ***LIB**RARY-**DEF**AULT / ***NONE** / <name 1..8>

,**DIALOG-CONTR**OL = ***DEFAULT** / ***NO** / ***Y**ES / ***ERROR**

Operands

ELEMENT = *LIBRARY-ELEMENT(...)
Specifications for the desired member designation.

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library containing the member.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = <filename 1..54 without-vers>
Name of the library containing the member.

LIBRARY = *LINK(...)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of an ADD-FILE-LINK command prior to calling LMSCONV.

ELEMENT = *ALL(...) / <composed-name 1..64 with-under with-wild(132)>(...)
Name of the members whose attributes are to be modified.

VERSION = *HIGHEST-EXISTING / *ALL / *UPPER-LIMIT /
<composed-name 1..24 with-under with-wild(52)>
Version of the member.

VERSION = *HIGHEST-EXISTING
The attributes of the member with the highest existing version are modified.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' in the library under the specified TYPE and name is used.

VERSION = <composed-name 1..24 with-under with-wild(52)>
Explicitly specifies the version of the member.

TYPE = *DEFAULT / *ALL / <alphanum-name 1..8 with-wild(20)>
Type of the member.

USER-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date given by the user.

USER-DATE = *ANY
The member has any date.

USER-DATE = *TODAY
The member with the current date is used.

USER-DATE = <date 8..10 with-compl>
The member whose date is entered explicitly in the form [YY]YY-MM-DD is used.

USER-DATE = *INTERVAL(...)
All members lying in the specified interval are used.

FROM = 1900-01-01 / <date 8..10 with-compl>
Beginning of interval.

TO = *TODAY / <date 8..10 with-compl>
End of interval.

CREATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Creation date of the member. For a description of the operands, see the USER-DATE operand of this statement.

MODIFICATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date of the last modification to the member. For a description of the operands, see the USER-DATE operand of this statement.

CODED-CHARACTER-SET = *ANY / *NONE / <name 1..8 with-wild(20)>
Character set assigned to the member.

CODED-CHARACTER-SET = *ANY
Selects members without regard to their assigned character set.

CODED-CHARACTER-SET = *NONE
Selects members which have not been assigned a character set.

CODED-CHARACTER-SET = <name 1..8 with-wild(20)>
Selects the members to which the specified character set has been assigned.

NEW-ATTRIBUTES = *PARAMETERS(...)
Specifies the attributes that the selected members are to receive.

CODED-CHARACTER-SET = *BY-SOURCE / *LIBRARY-DEFAULT / *NONE / <name 1..8>
Character set assigned to the members.

CODED-CHARACTER-SET = *BY-SOURCE
The member is assigned the character set of the source members.

CODED-CHARACTER-SET = *LIBRARY-DEFAULT
The members are assigned the character set of the library containing the member.

CODED-CHARACTER-SET = *NONE
No character set is assigned to the members.

CODED-CHARACTER-SET = <name 1..8>
Specifies the character set which is to be assigned to the members.

DIALOG-CONTROL = *DEFAULT / *NO / *YES / *ERROR
This operand specifies whether or not the execution of the statement is carried out interactively with the user.

For more detailed information on dialog control, see the MODIFY-DEFAULTS statement.

DIALOG-CONTROL = *DEFAULT
The default value is *NO or the current value set with MODIFY-DEFAULTS..

DIALOG-CONTROL = *NO
All members are processed without dialog queries, i.e. without the user being able to take control.
Exception: If a member or a library is locked, LMSCONV inquires whether the attempt to access it shall be repeated.

DIALOG-CONTROL = *YES / *ERROR
See the description in the MODIFY-DEFAULTS statement .

Note

In contrast to the CCSN of a file, the NETCCSN, i.e. the name of the character set of SAM node files on Net-Storage, is saved as a file attribute. It can therefore not be changed with MODIFY-ELEMENT-ATTRIBUTES.

7.4.5.2 MODIFY-LOGGING-PARAMETERS - Modify logging settings

The MODIFY-LOGGING-PARAMETERS statement modifies the global settings for the logging scope, output medium and logging format.

If one of these values is changed by the MODIFY-LOGGING-PARAMETERS statement, this new setting becomes the current setting. This remains valid for the LMSCONV run (*UNCHANGED) until a new MODIFY-LOGGING-PARAMETERS statement for this value or RESET-LOGGING-PARAMETERS is issued. At the beginning of the LMSCONV run, the values immediately following *UNCHANGED apply.

Format

MODIFY-**LOG**GING-**PAR**AMETERS

LOGGING = ***UNCHANGED** / ***MIN**IMUM / ***MAX**IMUM

,**TEXT-OUT**PUT = ***UNCHANGED** / ***SYSOUT** / ***SYSLST**(...) / ***NONE** / ***LIB**RARY-**ELEM**ENT(...)

***SYSLST**(...)

| **SYSLST-NUM**BER = ***STD** / <integer 1..99>

***LIB**RARY-**ELEM**ENT(...)

| **LIB**RARY = ***STD** / <filename 1..54 without-vers> / ***LINK**(...)

| ***LINK**(...)

| | **LINK**-NAME = <structured-name 1..8>

| ,**ELEM**ENT = <composed-name 1..64 with-under>(...)

| <composed-name 1..64 with-under>(...)

| | **VERSION** = ***UPPER-LIMIT** / <composed-name 1..24 with-under>

| ,**TYPE** = **P** / <alphanum-name 1..8>

| ,**WR**ITE-**MODE** = ***UNCHANGED** / ***CREATE** / ***REPLACE** / ***EXTEND** / ***ANY**

,**OUT**PUT-**LAYOUT** = ***UNCHANGED** / ***PAR**AMETERS(...)

***PAR**AMETERS(...)

| **LINE**S-**P**ER-**PAGE** = ***UNCHANGED** / <integer 1..9999>

| ,**LINE-SIZE** = ***UNCHANGED** / 132 / 80

| ,**EXTRA-F**ORM-**F**EED = ***UNCHANGED** / ***NO** / ***Y**ES

| ,**HEAD**ER-**LINE**S = ***UNCHANGED** / ***Y**ES / ***NO**

Operands

LOGGING = *UNCHANGED / *MINIMUM / *MAXIMUM
Defines the scope of LMSCNV logging.

LOGGING = *MINIMUM
Only error messages and negative acknowledgments are output.

LOGGING = *MAXIMUM
A complete LMSCONV log is output.

TEXT-OUTPUT = *UNCHANGED / *SYSOUT / *SYSLST(...) / *NONE / *LIBARARY-ELEMENT(...)
This parameter defines the output medium. If the medium is changed or if WRITE-MODE=*EXTEND is entered, page numbering always begins with 1. The preset value is *SYSOUT.

TEXT-OUTPUT = *SYSOUT
The output is written to SYSOUT.

TEXT-OUTPUT = *SYSLST(...)
The output is written to SYSLST.

SYSLST-NUMBER = *STD / <integer 1..99>
Denotes the SYSLST file to which the output is to be written.

SYSLST-NUMBER = *STD
The system file SYSLST is used.

SYSLST-NUMBER = <integer 1..99>
The system file with the specified number from the set SYSLST01 through SYSLST99 is used.

TEXT-OUTPUT = *NONE
Except for error messages, output is suppressed.

TEXT-OUTPUT = *LIBRARY-ELEMENT(...)
The output is stored in a library member.

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library in which the output is to be stored. Either the library set globally by means of OPEN-LIBRARY is used as standard, or the explicitly specified library or the library assigned via the link name is used.

LINK-NAME = <structured-name 1..8>
Link name of the library.

ELEMENT = <composed-name 1..64 with-under>(...)
Specifies the member in which the output is to be stored.

VERSION = *UPPER-LIMIT / <composed-name 1..24 with-under>
Specifies the version that the member is to receive.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' is generated.

VERSION = <composed-name 1..24 with-under>
The text specified here is interpreted as the version designation.

TYPE = P / <alphanum-name 1..8>
Specifies the member type.
By default the member in which the output is stored receives type P for print-edited files.

WRITE-MODE = *UNCHANGED / *CREATE / *REPLACE / *EXTEND / *ANY
Overwriting of a member having the same name. If the member does not exist under this name, it will be created as a new member. The preset value is *CREATE.

WRITE-MODE = *CREATE
The target member must not yet exist and is created as a new member.

WRITE-MODE = *REPLACE
The target member must already exist and is replaced.

WRITE-MODE = *EXTEND
The target member is extended if it already exists. Otherwise it will be created as a new member.

WRITE-MODE = *ANY
The target member is replaced if it already exists. Otherwise it will be created as a new member.

OUTPUT-LAYOUT = *UNCHANGED / *PARAMETERS(...)
This parameter defines the LMSCONV log format.

LINES-PER-PAGE = *UNCHANGED / <integer 1..9999>
This parameter defines the page length.
Default value: 64 lines

LINE-SIZE = *UNCHANGED / 132 / 80
This parameter defines the line length.The preset value is 132.

LINE-SIZE = 132
The line is to be 132 characters long.

LINE-SIZE = 80
The line is to be 80 characters long.

EXTRA-FORM-FEED = *UNCHANGED / *NO / *YES
This parameter controls an extra form feed. The preset value is *NO.

EXTRA-FORM-FEED = *NO
A form feed will only occur when the page is full.

EXTRA-FORM-FEED = *YES
A form feed will occur either when the page is full or when a change of statement or member takes place.

HEADER-LINES = *UNCHANGED / *YES / *NO
This parameter controls the output of headers.The preset value is *YES.

HEADER-LINES = *YES
Headers containing the library and member designations are output.

HEADER-LINES = *NO
No headers are output.

7.4.5.3 OPEN-LIBRARY - Open global library

OPEN-LIBRARY is used to define and open a global library. This is referenced in other statements by means of LIBRARY=*STD.

If two libraries are required in a statement, then the second library must be specified explicitly in the statement or via a link name.

Global libraries remain assigned until they are explicitly closed by means of the CLOSE-LIBRARY statement or until a new OPEN-LIBRARY statement is issued.

Global libraries are opened for reading as standard. If they are to be opened for reading and writing, the operand MODE=*UPDATE must be set. If a new library is set up, it must be generated with MODE=*UPDATE.

Format

OPEN-LIBRARY

LIBRARY = <filename 1..54 without-vers> / ***LINK**(...)

***LINK**(...)

| **LINK**-NAME = <structured-name 1..8>

,**MODE** = **_*READ_** / ***UPD**ATE(...)

***UPD**ATE(...)

| **STATE** = **_*ANY_** / ***OLD** / ***NEW**

Operands

LIBRARY = <filename 1..54 without-vers> / *LINK(...)
Specifies the library that is to be set up and opened as a global library.

LIBRARY = <filename 1..54 without-vers>
The library with the name specified here is set up as a global library and opened.

LIBRARY = *LINK(...)
The library assigned via the link name is set up as a global library and opened.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of /ADD-FILE-LINK prior to calling LMSCONV.

MODE = _*READ_ / *UPDATE(...)
Library open mode.

MODE = _*READ_
The library is opened only for reading. It must already exist.

MODE = *UPDATE(...)
The library is opened for reading and writing.

STATE = *ANY / *OLD / *NEW
Status of the library to be opened.

STATE = *ANY
The library may exist. If it does not exist, it will be created as a new library.

STATE = *OLD
The library must exist.

STATE = *NEW
The library must not exist. It will be created as a new library.

Examples

Assigning an existing library LIB1 as a global library:

```
//open-library library=lib1
```

Assigning an existing library as a global library via the link name:

```
/add-file-link link-name=glob-lib,file-name=lib1
//start-lmsconv
. . .
//open-library library=*link(link-name=glob-lib)
```

Creating a new library and assigning it as a global library:

```
//open-library library=lib1,mode=*update
```

7.4.5.4 SHOW-DEFAULTS - Output current default values

This statement outputs the current values of the LMSCONV defaults. These can be modified by means of the MODIFY-DEFAULTS statement.

Format

SHOW-DEFAULTS
DEFAULTS = ***STD** / ***ALL** / list-poss(2000): ***ELEMENT-ATTR**IBUTES / ***F**ILE**-ATTR**IBUTES / ***DESTROY-DATA** / ***WR**ITE**-MODE** / ***DIALOG-CONTR**OL / ***INF**ORMATION / ***LAYOUT** / ***SORT** / ***OUT**PUT**-FORM** / ***DEL**ETE**-SOUR**CE / ***MAX-ERROR-WEIGHT** / ***RUN-MODE** / ***NEXT-ATTEM**PT / ***TEXT-INF**ORMATION / ***MOD**ULE**-INF**ORMATION / ***PHASE-INF**ORMATION / ***LLM-INF**ORMATION

Operands

DEFAULTS = *STD
Outputs the default values for the following with their current settings: ELEMENT-ATTRIBUTES, FILE-ATTRIBUTES, DESTROY-DATA, WRITE-MODE, DIALOG-CONTROL, INFORMATION, LAYOUT, SORT, OUTPUT-FORM, DELETE-SOURCE, MAX-ERROR-WEIGHT, NEXT-ATTEMPT and PROTECTION.

DEFAULTS = *ALL
Outputs all defaults with their current settings.

DEFAULTS = *ELEMENT-ATTRIBUTES
Outputs the current settings for member type, source and target version, storage form and the file attributes.

DEFAULTS = *FILE-ATTRIBUTES
The current settings for the file access method is output.

DEFAULTS = *DESTROY-DATA
Whether or not data is to be overwritten is output.

DEFAULTS = *WRITE-MODE
The current setting for the write mode is output.

DEFAULTS = *DIALOG-CONTROL
The current setting for the dialog control is output.

DEFAULTS = *INFORMATION
Displays the current setting for the scope of directory information to be output.

DEFAULTS = *LAYOUT
Displays the current setting for the layout of the directory to be output.

DEFAULTS = *SORT
Displays the current setting for the sort criterion of the directory to be output.

DEFAULTS = *OUTPUT-FORM
Displays the current setting for the output form.

DEFAULTS = *DELETE-SOURCE
Outputs whether the source file is to be deleted or kept. Outputs the default setting for source file deletion.

DEFAULTS = *MAX-ERROR-WEIGHT
Outputs the default setting for spin-off control.

DEFAULTS = *RUN-MODE
Outputs the default setting for the run mode in which LMSCONV is to be called.

DEFAULTS = *NEXT-ATTEMPT
Outputs the default setting for the control of attempts to open files.

DEFAULTS = *TEXT-INFORMATION
Outputs the default setting for the scope of information for textual members.

DEFAULTS = *MODULE-INFORMATION
Outputs the default setting for the scope of information for object modules.

DEFAULTS = *PHASE-INFORMATION
Outputs the default setting for the scope of information for phases.

DEFAULTS = *LLM-INFORMATION
Outputs the default setting for the scope of information for link and load modules.

Example

```
//show-defaults   *std
ELEMENT-ATTRIBUTES
     TYPE              = *NONE
     ELEMENT-VERSION   = *ALL
     TO-ELEM-VERSION   = *BY-SOURCE
     STORAGE-FORM      = *STD
     SOURCE-ATTRIBUTES = *STD
FILE-ATTRIBUTES
     ACCESS-METHOD     = *ISAM
DESTROY-DATA           = *NO
WRITE-MODE             = *CREATE
DIALOG-CONTROL         = *NO
INFORMATION            = *MEDIUM
LAYOUT                 = *VARIABLE
SORT                   = *BY-NAME
OUTPUT-FORM            = *STD
DELETE-SOURCE          = *NO
PROTECTION             = *STD
MAX-ERROR-WEIGHT       = *SERIOUS
RUN-MODE               = *STD
NEXT-ATTEMPT           = *NO
```

All LMSCONV default values are output. These values are applicable immediately after the start of LMSCONV.

7.4.5.5 SHOW-ELEMENT - Display contents of member

SHOW-ELEMENT displays the contents of a specified member, depending on its type. The contents of text-oriented members, modules, phases and link and load modules (LLM) can be output. The representation format of the output is controlled by the OUTPUT-FORM operand. The meaning of the attributes with modules and link and load modules is explained in the “BINDER” manual [12 (Related publications)].

The statement is permissible for all member types. User-defined types are handled according to their respective base type. If the base type is unknown to LMSCONV, only the TEXT-INFORMATION and OUTPUT-FORM operands are effective. The scope of information can be restricted for textual members.

Format

SHOW-ELEMENT

ELEMENT = ***LIB**RARY**-ELEM**ENT (...)

***LIB**RARY**-ELEM**ENT(...)

| **LIB**RARY = ***STD** / <filename 1..54 without-vers> / ***LINK**(...)

| ***LINK**(...)

| | **LINK**-NAME = <structured-name 1..8>

| ,**ELEM**ENT = ***ALL**(...) / <composed-name 1..64 with-under with-wild(132)>(...)

| ***ALL**(...)

| | **VERSION** = ***HIGHEST-EXIST**ING / ***ALL** / ***UP**PER**-LIM**IT /

| | <composed-name 1..24 with-under with-wild(52)>

| <composed-name 1..64 with-under with-wild(132)>(...)

| | **VERSION** = ***HIGHEST-EXIST**ING / ***ALL** / ***UP**PER**-LIM**IT /

| | <composed-name 1..24 with-under with-wild(52)>

| ,**TYPE** = ***DEF**AULT / ***ALL** / <alphanum-name 1..8 with-wild(20)>

```
        |   ,USER-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
        |       *INTERVAL(...)
        |           |   FROM = 1900-01-01 / <date 8..10 with-compl>
        |           |   ,TO = *TODAY / <date 8..10 with-compl>
        |   ,CREATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
        |       *INTERVAL(...)
        |           |   FROM = 1900-01-01 / <date 8..10 with-compl>
        |           |   ,TO = *TODAY / <date 8..10 with-compl>
        |   ,MODIFICATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
        |       *INTERVAL(...)
        |           |   FROM = 1900-01-01 / <date 8..10 with-compl>
        |           |   ,TO = *TODAY / <date 8..10 with-compl>
,TEXT-INFORMATION = *DEFAULT / *ALL / *STATISTICS / *FILE-ATTRIBUTES / *PARAMETERS(...)
    *PARAMETERS(...)
        |   INFORMATION = *DEFAULT / *ALL / list-poss(2): *TEXT / *COMMENT
        |   ,RECORD-RANGE = *DEFAULT / *ALL / *RANGE(...)
        |       *RANGE(...)
        |           |   FROM = *DEFAULT / <integer 1..2147483647>
        |           |   ,TO = *DEFAULT / *LAST / <integer 1..2147483647>
        |   ,RECORD-PART = *DEFAULT / *ALL / *PART(...)
        |       *PART(...)
        |           |   START = *DEFAULT / <integer 1..32764>
        |           |   ,LENGTH = *DEFAULT / *REST / <integer 1..32764>
        |   ,RECORD-NUMBER = *DEFAULT / *BY-OUTPUT / *YES / *NO
,MODULE-INFORMATION = *DEFAULT / *ALL / *STATISTICS / *PARAMETERS(...)
    *PARAMETERS(...)
        |   INFORMATION = *DEFAULT / *ALL / *TXT(...) / *TXTP(...) / list-poss(9): *ESD / *ISD /
        |                   *LSD / *RLD / *REP / *INCLUDE / *DSDD / *REF / *END
        |       *TXT(...)
        |           |   CSECT-NAME = *DEFAULT / *ALL / <c-string 1..32 with-low> / <text 1..32>
```

```
        |           |   ,ADDRESS = *DEFAULT (...) / <x-string 1..8>(...)
        |           |       *DEFAULT(...)
        |           |           |   BASE-ADDRESS = *DEFAULT / <x-string 1..8>
        |           |       <x-string 1..8>(...)
        |           |           |   BASE-ADDRESS = *DEFAULT / <x-string 1..8>
        |           |   ,LENGTH = *DEFAULT / *REST / <integer 1..2147483647> / <x-string 1..8>
        |       *TXTP(...)
        |           |   MODIFICATION-ID = *DEFAULT / *ALL / <c-string 1..8 with-low> / *RANGE(...)
        |           |       *RANGE(...)
        |           |           |   FROM = *DEFAULT / *LOWEST / <c-string 1..8 with-low>
        |           |           |   ,TO = *DEFAULT / *HIGHEST / <c-string 1..8 with-low>
,PHASE-INFORMATION = *DEFAULT / *ALL / *STATISTICS / *PARAMETERS(...)
    *PARAMETERS(...)
        |   SEGMENT = *DEFAULT / *ALL / *ROOT / <name 1..8>
        |   ,INFORMATION = *DEFAULT / *ALL / *TXT(...) / *TXTP(...) / list-poss(4): *ESD / *ISD /
        |                  *LSD / *RLD
        |       *TXT(...)
        |           |   ADDRESS = *DEFAULT (...) / <x-string 1..8>(...)
        |           |       *DEFAULT(...)
        |           |           |   BASE-ADDRESS = *DEFAULT / <x-string 1..8>
        |           |       <x-string 1..8>(...)
        |           |           |   BASE-ADDRESS = *DEFAULT / <x-string 1..8>
        |           |   ,LENGTH = *DEFAULT / *REST / <integer 1..2147483647> / <x-string 1..8>
        |       *TXTP(...)
        |           |   MODIFICATION-ID = *DEFAULT / *ALL / <c-string 1..8 with-low> / *RANGE(...)
        |           |       *RANGE(...)
        |           |           |   FROM = *DEFAULT / *LOWEST / <c-string 1..8 with-low>
        |           |           |   ,TO = *DEFAULT / *HIGHEST / <c-string 1..8 with-low>
```

```
,LLM-INFORMATION = *DEFAULT / *ALL / *STATISTICS / *PARAMETERS(...)
    *PARAMETERS(...)
        | LLM-PART = *DEFAULT / *ALL / *SLICE(...) / *SUB-LLM(...)
        |     *SLICE(...)
        |         | NAME = *DEFAULT / <structured-name 1..32>
        |     *SUB-LLM(...)
        |         | PATH-NAME = *DEFAULT / <c-string 1..255 with-low> / <text 1..255>
        | ,INFORMATION = *DEFAULT / *ALL / *TXT(...) / *TXTP(...) / *LOGICAL(...) / *PHYSICAL / *REF /
        |                           list-poss(4): *RELOCATION / *ESVD / *ESVR / *LRLD
        |     *TXT(...)
        |         | CSECT-NAME = *DEFAULT / *ALL / <c-string 1..32 with-low> / <text 1..32>
        |         | ,ADDRESS = *DEFAULT (...) / <x-string 1..8>(...)
        |         |     *DEFAULT(...)
        |         |         | BASE-ADDRESS = *DEFAULT / <x-string 1..8>
        |         |     <x-string 1..8>(...)
        |         |         | BASE-ADDRESS = *DEFAULT / <x-string 1..8>
        |         | ,LENGTH = *DEFAULT / *REST / <integer 1..2147483647> / <x-string 1..8>
        |     *TXTP(...)
        |         | CSECT-NAME = *DEFAULT / *ALL / <c-string 1..32 with-low> / <text 1..32>
        |         | ,MODIFICATION-ID = *DEFAULT / *ALL / <c-string 1..12 with-low> / *RANGE(...)
        |         |     *RANGE(...)
        |         |         | FROM = *DEFAULT / *LOWEST / <c-string 1..12 with-low>
        |         |         | ,TO = *DEFAULT / *HIGHEST / <c-string 1..12 with-low>
        |     *LOGICAL(...)
        |         | LEVEL = *DEFAULT / *ALL / *NEXT
,OUTPUT-FORM = *DEFAULT / *STD / *CHARACTER / *HEXADECIMAL / *DUMP
```

Operands

ELEMENT = *LIBRARY-ELEMENT(...)
Specifications for the desired member designation.

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library containing the member.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = <filename 1..54 without-vers>
Name of the library containing the member.

LIBRARY = *LINK(...)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library.

ELEMENT = *ALL(...) / <composed-name 1..64 with-under with-wild(132)>(...)

ELEMENT = *ALL(...)
Information is output on all members.

ELEMENT = <composed-name 1..64 with-under with-wild(132)>(...)
Name of the member to be displayed.

VERSION = *HIGHEST-EXISTING / *ALL / *UPPER-LIMIT
<composed-name 1..24 with-under with-wild(52)>
Version of the member to be output.

VERSION = *HIGHEST-EXISTING
The member with the highest existing version is output.

VERSION = *UPPER-LIMIT
The highest possible version X'FF' in the library under the specified TYPE and name is displayed.

VERSION = <composed-name 1..24 with-under with-wild(52)>
Explicitly specifies the version of the member to be displayed.

TYPE = *DEFAULT / *ALL / <alphanum-name 1..8 with-wild(20)>
Type of the member to be output. If the value is *DEFAULT and the current value set with MODIFY-DEFAULTS is *NONE, LMSCONV requires a type specification.

USER-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVALL(...)
Date given by the user.

USER-DATE = *ANY
The member to be output has any date.

USER-DATE = *TODAY
The member with the current date is output.

USER-DATE = <date 8..10 with-compl>
The member whose date is entered explicitly in the form [YY]YY-MM-DD is output.

USER-DATE = *INTERVAL(...)
All members lying in the specified interval are output.

FROM = 1900-01-01 / <date 8..10 with-compl>
Beginning of interval.

TO = *TODAY / <date 8..10 with-compl>
End of interval.

CREATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Creation date of the member. For a description of the operands, see the USER-DATE operand of this statement.

MODIFICATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date of the last modification to the member. For a description of the operands, see the USER-DATE operand of this statement.

TEXT-INFORMATION = *DEFAULT / *ALL / *STATISTICS / *FILE-ATTRIBUTES / *PARAMETER(...)
Defines the information scope for all members except member types R, C and L. The default setting is *ALL (see below) or the current value set with MODIFY-DEFAULTS.

TEXT-INFORMATION = *ALL
Everything is output.

TEXT-INFORMATION = *STATISTICS
The number of records per record type and the total number of records are output. For each record type, the total of all record lengths (without record length fields) is output, as is the total record length across all record types.

TEXT-INFORMATION = *FILE-ATTRIBUTES
Only the stored file attributes are output.

> For all PAM members (except L-type members), the output includes the last byte pointer (LBP), if it has been stored as a file attribute. For SAM members, the output includes net coded character set (NETCCSN), if it has been stored as a file attribute.

TEXT-INFORMATION = *PARAMETERS(...)
Defines a member extract to be output.

INFORMATION = *DEFAULT / *ALL / list-poss(2): *TEXT / *COMMENT
The member section to be output. The default setting is *ALL (see below) or the current value set with MODIFY-DEFAULTS.

INFORMATION = *ALL
Outputs all user record types.

INFORMATION = *TEXT
Outputs the text itself, i.e. record type 1.

INFORMATION = *COMMENT
Outputs the separately stored comment, i.e. record type 2.

RECORD-RANGE = *DEFAULT / *ALL / *RANGE(...)
The section of the member to be processed. The default setting is *ALL (see below) or the current value set with MODIFY-DEFAULTS.

RECORD-RANGE = *ALL
Processes all user record types.

RECORD-RANGE = *RANGE(...)
Specifies the range of record numbers which is to be processed. The record numbers refer not to a record type, but to the section of the member designated by means of INFORMATION=. Within this section, the records are numbered consecutively from 1 through n.

FROM = *DEFAULT / <integer 1..2147483647>
Beginning of the range, specified by the first record number. Record number 1 is the default value.

TO = *DEFAULT / *LAST / <integer 1..2147483647>
End of the range, specified by the last record number. The last record number is used as the default value.

RECORD-PART = *DEFAULT / *ALL / *PART(...)
Specifies the part of the record to be processed.

RECORD-PART = *ALL
Processes the entire record.

RECORD-PART = *PART(...)
Specifies the part of the record to be processed. If the default values are not changed, the entire record is processed.

START = *DEFAULT / <integer 1..32764>
Beginning of the record part, specified by the first character in the record. The first character is used as the default value.

LENGTH = *DEFAULT / *REST / <integer 1..32764>
Length of the record part. The remainder of the record is used as the default value.

RECORD-NUMBER = *DEFAULT / *BY-OUTPUT / *YES / *NO
Specifies output of the record numbers. The default value is *BY-OUTPUT (see below) or the current value set with MODIFY-DEFAULTS.

RECORD-NUMBER = *BY-OUTPUT
Only if the output is directed to SYSOUT will no record numbers be output. With any other output medium, the record numbers are included in the output.

RECORD-NUMBER = *YES
The record numbers are also output to SYSOUT.

RECORD-NUMBER = *NO
No record numbers are included in the output..

MODULE-INFORMATION = *DEFAULT / *ALL / *STATISTICS / *PARAMETERS(...)
Defines the information scope for object modules (R-type members). The default setting is *ALL (see below) or the current value set with MODIFY-DEFAULTS.

MODULE-INFORMATION = *ALL
Everything is output.

MODULE-INFORMATION = *STATISTICS
The name, length and address of the CSECTS and also the overall length of the module are output.

MODULE-INFORMATION = *PARAMETERS(...)
This parameter determines whether all record types or only selected record types are output.

INFORMATION = *DEFAULT / *ALL / *TXT(...) / *TXTP(...) / list-poss(9): *ESD / *ISD / *LSD / *RLD / *REP / *INCLUDE / *DSDD / *REF / *END
The record types listed here can be selected. The default setting is *ALL or the current value set with MODIFY-DEFAULTS.

INFORMATION = *TXT(...)
Text records are selected.

CSECT-NAME = *DEFAULT / *ALL / <c-string 1..32 with-low> / <text 1..32>
The text records can be restricted to one CSECT. The default setting is *ALL or the current value set with MODIFY-DEFAULTS.

ADDRESS = *DEFAULT / <x-string 1..8>(...)
Start address of the text. The default setting is X'00000000' or the current value set with MODIFY-DEFAULTS.

BASE-ADDRESS = *DEFAULT / <x-string 1..8>
The base address specified here is added to the start address. The default setting is X'00000000' or the current value set with MODIFY-DEFAULTS.

LENGTH = *DEFAULT / *REST / <integer 1..2147483647> / <x-string 1..8>
Length of the text. The default setting is *REST or the current value set with MODIFY-DEFAULTS.

INFORMATION = *TXTP(...)
TXTP records are output.

MODIFICATION-ID = *DEFAULT / *ALL / <c-string 1..8 with-low> / *RANGE(...)
Those TXTP records with the specified identification are selected. The default setting is *ALL or the current value set with MODIFY-DEFAULTS.

MODIFICATION-ID = *RANGE(...)
A group of TXTP records lying in a range can be selected.

FROM = *DEFAULT / *LOWEST / <c-string 1..8 with-low>
The beginning of the range is by default the lowest identification for the TXTP records, otherwise the value entered here.

TO = *DEFAULT / *HIGHEST / <c-string 1..8 with-low>
The end of the range is by default the highest identification for the TXTP records, otherwise the value entered here.

PHASE-INFORMATION = *DEFAULT / *ALL / *STATISTICS / *PARAMETER(...)
Defines the information scope for phases (C-type members). The default setting is *ALL or the current value set with MODIFY-DEFAULTS.

PHASE-INFORMATION = *ALL
Everything is output.

PHASE-INFORMATION = *STATISTICS
The name, length and address of the segment and also the overall length of the segment are output.

PHASE-INFORMATION = *PARAMETERS(...)
This parameter determines whether all record types or only selected record types are output.

SEGMENT = *DEFAULT / *ALL / *ROOT / <name 1..8>
Phase segment that is selected. The default setting is *ALL or the current value set with MODIFY-DEFAULTS.

INFORMATION = *DEFAULT / *ALL / *TXT(...) / *TXTP(...) / list-poss(4): *ESD / *ISD / *LSD / *RLD
The record types listed here can be selected. The default setting is *ALL or the current value set with MODIFY-DEFAULTS.

INFORMATION = *TXT(...)
Text records are selected.

ADDRESS = *DEFAULT / <x-string 1..8>(...)
Start address of the text. The default setting is X'00000000' or the current value set with MODIFY-DEFAULTS.

BASE-ADDRESS = *DEFAULT / <x-string 1..8>
The base address specified here is added to the start address. The default setting is X'00000000' or the current value set with MODIFY-DEFAULTS.

LENGTH = *DEFAULT / *REST / <integer 1..2147483647> / <x-string 1..8>Length of the text. The default setting is *REST or the current value set with MODIFY-DEFAULTS.

INFORMATION = *TXTP(...)
TXTP records are output.

MODIFICATION-ID = *DEFAULT / *ALL / <c-string 1..8 with-low> / *RANGE(...)Those TXTP records with the specified identification are selected. The default setting is *ALL or the current value set with MODIFY-DEFAULTS.

MODIFICATION-ID = *RANGE(...)
A group of TXTP records lying in a range can be selected.

FROM = *DEFAULT / *LOWEST / <c-string 1..8 with-low>
The beginning of the range is by default the lowest identification for the TXTP records, otherwise the value entered here.

TO = *DEFAULT / *HIGHEST / <c-string 1..8 with-low>
The end of the range is by default the highest identification for the TXTP records, otherwise the value entered here.

LLM-INFORMATION = *DEFAULT / *ALL / *STATISTICS / *PARAMETERS(...)
Defines the information scope for link and load modules (L-type members). The default setting is *ALL (see below) or the current value set with MODIFY-DEFAULTS.

LLM-INFORMATION = *ALL
Everything is output.

LLM-INFORMATION = *STATISTICS
General information on the link and load module (name, copyright, ...) is output.

LLM-INFORMATION = *PARAMETERS(...)
This parameter determines whether all record types or only selected record types are output.

LLM-PART = *DEFAULT / *ALL / *SLICE(...) / *SUB-LLM(...)
Specifies the LLM part to be selected. By default the entire LLM is selected.

LLM-PART = *SLICE(...)
Specifies the slice to be output.

NAME = <structured-name 1..32>
Name of the slice to be output.

LLM-PART = *SUB-LLM(...)
Specifies the sub-LLM to be output.

PATH-NAME = *DEFAULT / <c-string 1..255 with-low> / <text 1..255>
The sub-LLM to be output is determined by way of its path name. The default setting is *ALL or the current value set with MODIFY-DEFAULTS.

INFORMATION = *DEFAULT / *ALL / *TXT(...) / *TXTP(...) / *LOGICAL(...) / *PHYSICAL / *REF / list-poss (4): *ESVD / *ESVR / *LRLD / *RELOCATION
The record types listed here can be selected. The default setting is *ALL or the current value set with MODIFY-DEFAULTS.

INFORMATION = *TXT(...)
Text records are selected.

CSECT-NAME = *DEFAULT / *ALL / <c-string 1..32 with-low> / <text 1..32>The text records can be restricted to one CSECT.

ADDRESS = *DEFAULT / <x-string 1..8>(...)
Start address of the text. The default setting is X'00000000' or the current value set with MODIFY-DEFAULTS.

BASE-ADDRESS = *DEFAULT / <x-string 1..8>
The base address specified here is added to the start address. The default setting is X'00000000' or the current value set with MODIFY-DEFAULTS.

LENGTH = *DEFAULT / *REST / <integer 1..2147483647> / <x-string 1..8>
Length of the text. The default setting is *REST or the current value set with MODIFY-DEFAULTS.

INFORMATION = *TXTP(...)
TXTP records are output.

CSECT-NAME = *DEFAULT / *ALL / <c-string 1..32 with-low> / <text 1..32>
The TXTP records can be restricted to one CSECT. The default setting is *ALL or the current value set with MODIFY-DEFAULTS.

MODIFICATION-ID = *DEFAULT / *ALL / <c-string 1..12 with-low> / *RANGE(...)
Those TXTP records with the specified identification are selected. The default setting is *ALL or the current value set with MODIFY-DEFAULTS.

MODIFICATION-ID = *RANGE(...)
A group of TXTP records lying in a range can be selected.

FROM = *DEFAULT / *LOWEST / <c-string 1..12 with-low>
The beginning of the range is by default the lowest identification for the TXTP records, otherwise the value entered here.

TO = *DEFAULT / *HIGHEST / <c-string 1..12 with-low>
The end of the range is by default the highest identification for the TXTP records, otherwise the value entered here.

INFORMATION = *LOGICAL(...)
Outputs the logical structure of the LLM.

LEVEL = *DEFAULT / *ALL / *NEXT
Outputs all substructures by default; otherwise, only the next substructure.

INFORMATION = *PHYSICAL
Outputs the physical structure of the LLM.

OUTPUT-FORM = *DEFAULT / *STD / *CHARACTER / *HEXADECIMAL / *DUMP
Specifies the form of representation for the output. The default setting is *STD or the current value set with MODIFY-DEFAULTS.

OUTPUT-FORM = *STD
The form of representation is selected dependent on the member type.

OUTPUT-FORM = *CHARACTER
The output is in alphanumeric form.

OUTPUT-FORM = *HEXADECIMAL
The output is in alphanumeric and hexadecimal form, one above the other.

OUTPUT-FORM = *DUMP
The output is in alphanumeric and hexadecimal form, side by side. For member types S, P, D, J and M, this operand has the same effect as OUTPUT-FORM=*HEXADECIMAL.

Examples

The member LETTER.A, which contains the text “Dear ...”, is to be output.

```
//show-element (element=letter.a,type=d)
INPUT  LIBRARY= :N:$USER.TEST.LIB
INPUT  ELEMENT= (D)LETTER.A/ (0001)/<date>
Dear ...
...
...
Yours sincerely, ...
NUMBER OF PROCESSED RECORDS IS    123
```

The stored file attributes of a member with LBP are to be output.

```
//show-element (element=net.lbp.2dd,type=x),text-information=file-attributes
INPUT  LIBRARY= :SQGB:$TSOS.BIB.ALI
INPUT  ELEMENT= (X)PTF04/§(0001)/<date>
ORIGINAL FILE ATTRIBUTES :
FILENAME= :CK33:$TSOS.PTF04
FILSIZE = 0000009   FCBTYPE = PAM       2ND ALLO= 00006     LBP     = 13312
SHARE   = NO        ACCESS  = WRITE
BLKCTRL = DATA      BLKSIZE = 014336    RECFORM = U         RECSIZE = 00000
PERFORM = STANDARD  USAGE   = NOT-SPEC
```

The stored file attributes of a member with NETCCSN are to be output.

```
//show-element (element=stf03,type=x),text-information=file-attributes
INPUT  LIBRARY= :SQGB:$TSOS.BIB.ALI
INPUT  ELEMENT= (X)STF03/@(0001)/<date>
ORIGINAL FILE ATTRIBUTES :
FILENAME= :CK33:$DMS01.STF03
FILSIZE = 0000032   FCBTYPE = SAM       2ND ALLO= 00016
SHARE   = NO        ACCESS  = WRITE
BLKCTRL = DATA      BLKSIZE = 032768    RECFORM = V         RECSIZE = 00000
PERFORM = STANDARD  USAGE   = NOT-SPEC  NETCCSN = ISO88591
```

7.4.5.6 SHOW-ELEMENT-ATTRIBUTES - Display member attributes

SHOW-ELEMENT-ATTRIBUTES outputs the directory entries of the specified members or of the entire library. The entries are output on the medium specified by the MODIFY-LOGGING-PARAMETERS statement.

The directory is always output sorted by type. The remainder of the sort sequence is determined by the SORT operand. The default sort sequence is type, name and version.

The INFORMATION and LAYOUT operands are used to specify the scope and format of the directory output. By default, the type, name, version, variant number and date are output.

With the aid of the SECONDARY-NAME and SECONDARY-ATTRIBUTE operands, the directory can be limited to the members containing a certain reference entry.

> i In order to obtain the entire contents of a library (all members with all versions), it is sufficient to specify only SHOW-ELEMENT-ATTRIBUTES without any operands, provided no specific member type or version was set using MODIFY-DEFAULTS.

Format

SHOW-ELEMENT**-ATTR**IBUTES

ELEMENT = ***LIB**RARY**-ELEM**ENT (...)

***LIB**RARY**-ELEM**ENT(...)

| **LIB**RARY = ***STD** / <filename 1..54 without-vers> / ***LINK**(...)

| ***LINK**(...)

| | **LINK**-NAME = <structured-name 1..8>

| ,**ELEM**ENT = ***ALL** (...) / <composed-name 1..64 with-under with-wild(132)>(...)

| ***ALL**(...)

| | **VERSION** = ***DEFAULT** / ***ALL** / ***HIGH**EST**-EXIST**ING / ***UP**PER**-LIM**IT /

| | <composed-name 1..24 with-under with-wild(52)>

| <composed-name 1..64 with-under with-wild(132)>(...)

| | **VERSION** = ***DEFAULT** / ***ALL** / ***HIGH**EST**-EXIST**ING / ***UP**PER**-LIM**IT /

| | <composed-name 1..24 with-under with-wild(52)>

| ,**TYPE** = ***DEF**AULT / ***ALL** / <alphanum-name 1..8 with-wild(20)>

| ,**USER-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)
| ***INTERVAL**(...)
| | **FROM** = **1900-01-01** / <date 8..10 with-compl>
| | ,**TO** = ***TODAY** / <date 8..10 with-compl>
| ,**CRE**ATION**-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)
| ***INTERVAL**(...)
| | **FROM** = **1900-01-01** / <date 8..10 with-compl>
| | ,**TO** = ***TODAY** / <date 8..10 with-compl>
| ,**MOD**IFICATION**-DATE** = ***ANY** / ***TODAY** / <date 8..10 with-compl> / ***INTERVAL**(...)
| ***INTERVAL**(...)
| | **FROM** = **1900-01-01** / <date 8..10 with-compl>
| | ,**TO** = ***TODAY** / <date 8..10 with-compl>
| ,**USER-TIME** = ***ANY** / <time 1..8> / ***INTERVAL**(...)
| ***INTERVAL**(...)
| | **FROM** = **00:00:00** / <time 1..8>
| | ,**TO** = **23:59:59** / <time 1..8>
| ,**CRE**ATION**-TIME** = ***ANY** / <time 1..8> / ***INTERVAL**(...)
| ***INTERVAL**(...)
| | **FROM** = **00:00:00** / <time 1..8>
| | ,**TO** = **23:59:59** / <time 1..8>
| ,**MOD**IFICATION**-TIME** = ***ANY** / <time 1..8> / ***INTERVAL**(...)
| ***INTERVAL**(...)
| | **FROM** = **00:00:00** / <time 1..8>
| | ,**TO** = **23:59:59** / <time 1..8>
| ,**CODED-CHAR**ACTER**-SET** = ***ANY** / ***NONE** / <name 1..8> / <composed-name 1..8 with-wild(20)>
| ,**SEC**ONDARY**-NAME** = ***ANY** / <alphanum-name 1..32 with-wild(68)>
| ,**SEC**ONDARY**-ATTR**IBUTE = ***ANY** / ***CSECT** / ***ENTRY**

,**INF**ORMATION = ***DEFAULT** / ***MED**IUM / ***MIN**IMUM / ***MAX**IMUM / ***SUMM**ARY

,**LAYOUT** = ***DEFAULT** / ***VAR**IABLE / ***FIXED**

,**SORT** = ***DEFAULT** / ***BY-NAME** / ***BY-VERS**ION / ***BY-USER-DATE** / ***BY-SEC**ONDARY**-NAME**

Operands

ELEMENT = *LIBRARY-ELEMENT(...)
Specifications for the desired member designation.

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library for which the directory is to be output.

LIBRARY = *STD
The library opened by OPEN-LIBRARY.

LIBRARY = <filename 1..54 without-vers>
Outputs the directory of the library specified here.

LIBRARY = *LINK(...)
The library assigned via the link name.

LINK-NAME = <structured-name 1..8>
Link name of the library.

ELEMENT = *ALL(...) /
<composed-name 1..64 with-under with-wild(132)>(...)
Name of the member for which the library entry is to be output.
If the default value “*ALL” is entered, LMSCONV outputs the library entries for all of the members with the corresponding version and type.

VERSION = *DEFAULT / *ALL / *HIGHEST-EXISTING / *UPPER-LIMIT /<composed-name 1..24 with-under with-wild(52)>
Version of the member. The default setting is *ALL or the current value set with MODIFY-DEFAULTS.

VERSION = *ALL
Outputs the library entries of all members selected above, regardless of their respective versions.

VERSION = *HIGHEST-EXISTING
The library entries of all members selected above are output with the highest existing version.

VERSION = *UPPER-LIMIT
The library entries of all members selected above are output with the version X'FF'.

VERSION = <composed-name 1..24 with-under with-wild(52)>
The library entries of all members selected above are output with the version specified here.

TYPE = *DEFAULT / <alphanum-name 1..8 with-wild(20)>
Type of the member which is to be output.
If the LMSCONV default value for TYPE is *NONE, then *DEFAULT has the same effect as *ALL.

USER-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date given by the user.

USER-DATE = *ANY
The member has any date.

USER-DATE = *TODAY
The library entries of all members with the current date are output.

USER-DATE = <date 8..10 with-compl>
The library entries of all members with the date specified here in the form [YY]YY-MM-DD are output.

USER-DATE = *INTERVAL(...)
The library entries of all members lying in the specified interval are output.

FROM = 1900-01-01 / <date 8..10 with-compl>
Beginning of interval.

TO = *TODAY / <date 8..10 with-compl>
End of interval.

CREATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Creation date of the member. For a description of the operands, see the USER-DATE operand of this statement.

MODIFICATION-DATE = *ANY / *TODAY / <date 8..10 with-compl> / *INTERVAL(...)
Date of the last modification to the member. For a description of the operands, see the USER-DATE operand of this statement.

USER-TIME = *ANY / <time 1..8> / *INTERVAL(...)
Time given by the user.

USER-TIME = *ANY
The library entries of all members are output, regardless of the time.

USER-TIME = <time 1..8>
The library entries of all members with the time specified in the form HH:MM:SS are output.

USER-TIME = *INTERVAL(...)
The library entries of all members lying in the specified interval are output.

FROM = 00:00:00 / <time 1..8>
Beginning of interval.

TO = 23:59:59 / <time 1..8>
End of interval.

CREATION-TIME = *ANY / <time 1..8> / *INTERVAL(...)
Creation time of the member. For a description of the operands, see USER-TIME of this statement

MODIFICATION-TIME = *ANY / <time 1..8> / *INTERVAL(...)
Time of the last modification to the member. For a description of the operands, see USER-TIME of this statement.

CODED-CHARACTER-SET = *ANY / *NONE / <name 1..8> / <composed-name 1..8 with-wild(20)>
Character set assigned to the member.

CODED-CHARACTER-SET = *ANY
Selects members without regard to their assigned character set.

CODED-CHARACTER-SET = *NONE
Selects members which have not been assigned a character set.

CODED-CHARACTER-SET = <name 1..8> / <composed-name 1..8 with-wild(20)>
Selects the members to which the specified character set has been assigned.

SECONDARY-NAME = *ANY / <alphanum-name1...32 with-wild(68) >
Secondary name.
If anything other than *ANY is specified here, the selection is made via the secondary directory of the library. If wildcards are specified, only the first 32 characters of the secondary name are used to determine the selection.

SECONDARY-ATTRIBUTE = *ANY / *CSECT / *ENTRY
Secondary attribute.
If anything other than *ANY is specified here, the selection is made via the secondary directory of the library.

INFORMATION = *DEFAULT / *MEDIUM / *MINIMUM / *MAXIMUM / *SUMMARY
This parameter defines the scope of directory information to be output. It also specifies the scope of the structure output (see “Parameter dependencies”). The default setting is *MEDIUM or the current value set with MODIFY-DEFAULTS.

INFORMATION = *MEDIUM
Outputs the type, name, version, variant number and the date or the member size.

INFORMATION = *MINIMUM
Outputs only the type, name and version.

INFORMATION = *MAXIMUM
All the information is output.

INFORMATION = *SUMMARY
Outputs only the number of members per type.

LAYOUT = *DEFAULT / *VARIABLE / *FIXED
This parameter defines the format of the directory to be output. The default setting is *VARIABLE or the current value set with MODIFY-DEFAULTS.

LAYOUT = *VARIABLE
The number of print columns depends on the longest member designation within a member type.

LAYOUT = *FIXED
The directory is printed in a single column in fixed format. Single column means that the entries in the directory appear one beneath the other.

SORT = *DEFAULT / *BY-NAME / *BY-VERSION / *BY-USER-DATE / *BY-SECONDARY-NAME
Sort criterion for the directory entries of the selected members. The type is always used as the first sort criterion. The default setting is *BY-NAME or the current value set with MODIFY-DEFAULTS.

SORT = *BY-NAME
The directory entries of the selected members are sorted on the basis of the following sequence: type, name and version.

SORT = *BY-VERSION
The directory entries of the selected members are sorted on the basis of the following sequence: type, version and name.

SORT = *BY-USER-DATE
The directory entries of the selected members are sorted on the basis of the following sequence: type, user date, name and version.

SORT = *BY-SECONDARY-NAME
The directory entries of the selected members are sorted on the basis of the following sequence: type, secondary name, secondary attribute, name and version.

> If the SECONDARY-NAME or SECONDARY-ATTRIBUTE operand is specified and if the value of this operand is other than *ANY, then a header will also be included in the directory output, providing information on the secondary name and the secondary attribute. The secondary names, however, are not displayed at maximum length.

Parameter dependencies

The following dependencies exist between the INFORMATION, SORT and LAYOUT operands:

- The current date or the member size only influences the sorting if it, too, is to be included in the directory output. In this case, the INFORMATION operand must have a setting other than *MINIMUM.
- The LAYOUT operand is effective only if INFORMATION =*MEDIUM/*MINIMUM is specified. For all other settings, the directory is always output in fixed format.
- If INFORMATION=*MAXIMUM is specified, the information for each member will be too long for a single line. The information will then be output in a format independent of layout control.
- Output of the directory is accelerated if the INFORMATION operand is set to *MINIMUM and the selection is restricted to the ELEMENT, VERSION and TYPE operands. All other operands should be set to the value *ANY.

The SORT and INFORMATION operands (except for INFORMATION=*SUMMARY) influence the sort sequence of the directory. The following table shows these dependencies:

SORT	INFORMATION	
	***MINIMUM**	***MEDIUM / *MAXIMUM**
*BY-NAME	1. Type 2. Name 3. Version	1. Type 2. Name 3. Version
*BY-VERSION	1. Type 2. Version 3. Name	1. Type 2. Version 3. Name
*BY-USER-DATE	1. Type 2. Name 3. Version	1. Type 2. User date 3. Name 4. Version
*BY-SECONDARY-NAME with reference[1)]	1. Type 2. Secondary name 3. Secondary attribute 4. Name 5. Version	1. Type 2. Secondary name 3. Secondary attribute 4. Name 5. Version
*BY-SECONDARY-NAME without reference[2)]	1. Type 2. Name 3. Version	1. Type 2. Name 3. Version

[1)] with reference means that either the secondary name or the secondary attribute specified for the member name is not equal to the default value *ANY, i.e.:

SECONDARY-NAME = <alphanum-name...> and/or SECONDARY-ATTRIBUTE = *CSECT or *ENTRY

[2)] without reference means that neither a secondary name nor a secondary attribute was specified for member selection

Example

Outputting the directory for the library USER.BSPLIB. The library contains precisely one member which is displayed with all its attributes..

```
//SHOW-ELEMENT-ATTRIBUTES -
(LIBRARY=USER.BSPLIB,ELEMENT=*(VERSION=*)),INFORMATION=*MAXIMUM
INPUT LIBRARY= :N:$USER.USER.BSPLIB
TYPE = D
NAME = TEST
VERSION = @ VARIANT = 0001
--------------------------------GENERAL-----------------------------------
ELEM-SIZE = 12
STORAGEW = *FULL
STATE = *IN-HOLD HOLDER = MUBF
--------------------------------HISTORY-----------------------------------
USER-DATE = <date> CRE-DATE = <date> MOD-DATE = <date>
USER-TIME = <time> CRE-TIME = <time> MOD-TIME = <time>
ACC-DATE = <date>
ACC-TIME = <time>
--------------------------------SECURITY----------------------------------
READ-PASS = *NONE READ-USER = *OWNER *GROUP -
WR-PASS = *NONE WR-PASS = *OWNER - -
```

7.4.5.7 SHOW-LIBRARY-ATTRIBUTES - Display library attributes

This statement displays all attributes set for the library. These are as follows:

- library size in 2-K units
- number of 2-K units available (can be removed by copying)
- library format (NK2/NK4)
- UPAM protection (Y/N)

Format

SHOW-LIBRARY**-ATTR**IBUTES

LIBRARY = ***STD** / <filename 1..54 without-vers> / ***LINK**(...)

***LINK**(...)

| **LINK**-NAME = <structured-name 1..8>

Operands

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
The library whose attributes are to be displayed.

LIBRARY = *STD
The global library opened by OPEN-LIBRARY is displayed.

LIBRARY = <filename 1..54 without-vers>
Name of the library whose status is to be displayed.

LIBRARY = *LINK(...)
The status of the library assigned via a link name is displayed.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of /ADD-FILE-LINK prior to calling LMSCONV.

Example

```
//show-library-attributes
INPUT  LIBRARY= :X:$USER.LIB
INIT-ELEM-P= *NONE
ADMINISTRAT= *NONE
FILE-SIZE  = 291       FREE-SIZE  = 62       FORMAT = NK2   UPAM-PROT = N
ACCESS-DATE= *NONE     WR-CONTROL = *NONE    STORAGE= *NONE
```

7.4.5.8 SHOW-LIBRARY-STATUS - Display library status

This statement displays the status of the libraries used. LMSCONV outputs the following information on execution of the statement:

- name of the library/libraries
- status of the library/libraries (opened or closed)
- assigned link name, if any

Format

SHOW-LIBRARY**-STA**TUS

LIBRARY = ***ALL** / ***STD** / <filename 1..54 without-vers> / ***LINK**(...)

***LINK**(...)

| **LINK**-NAME = <structured-name 1..8>

Operands

LIBRARY = *ALL / *STD / <filename 1..54 without-vers> / *LINK(...)
The library whose status is to be displayed.

LIBRARY = *ALL
All libraries used are displayed.

LIBRARY = *STD
The global library opened by OPEN-LIBRARY is displayed.

LIBRARY = <filename 1..54 without-vers>
Name of the library whose status is to be displayed.

LIBRARY = *LINK(...)
The status of the library assigned via a link name is displayed.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of /ADD-FILE-LINK prior to calling LMSCONV.

Example

Five different libraries were used during the LMSCONV session. One library was addressed via the link name LIB1:

```
//show-library-status
STATUS FILENAME                                          MODE    LINK
OPEN   :N:$USER.LMSPL.LIB                                UPDATE
CLOSED :N:$USER.MODUL.LIB                                        LIB1
CLOSED :N:$USER.MACRO.LIB
CLOSED :N:$USER.QUELL.LIB
CLOSED :N:$USER.TEST.LIB
```

7.4.5.9 SHOW-LOGGING-PARAMETERS - Display global LMSCONV parameters

This statement outputs the global LMSCONV option values currently in force. These values are modified by means of the MODIFY-LOGGING-PARAMETERS statement.

If this statement is specified without operands, the preset values for all of the parameters are output (see example).

Format

SHOW-LOGGING-PARAMETERS
PARAMETERS = ***ALL** / list-poss(2000): ***LOG**GING / ***OUTPUT** / ***TEXT-OUT**PUT / ***OUT**PUT**-LAYOUT**

Operands

PARAMETERS = *ALL / *LOGGING / *OUTPUT / *TEXT-OUTPUT / *OUTPUT-LAYOUT
The current settings of all the parameters are output

PARAMETERS = *ALL
The current settings of all the parameters are output.

PARAMETERS = *LOGGING / *OUTPUT / *TEXT-OUTPUT
The output medium setting is output.

PARAMETERS = *OUTPUT-LAYOUT
The parameter settings for the LMSCONV log format are output.

Example

```
//show-logging-parameters
LOGGING                 = *MINIMUM
TEXT-OUTPUT             = *SYSOUT
OUTPUT-LAYOUT
  LINES-PER-PAGE        = 60
  LINE-SIZE             = 132
  EXTRA-FORM-FEED       = *NO
  HEADER-LINES          = *YES
```

All global LMSCONV options are output. These values are applicable immediately after the start of LMSCONV.

7.4.5.10 SHOW-TYPE-ATTRIBUTES - Display attributes of a member type

This statement outputs all the attributes set for a given member type.

Format

SHOW-TYPE-ATTRIBUTES

LIBRARY = ***STD** / <filename 1..54 without-vers> / ***LINK**(...)

***LINK**(...)

| **LINK**-NAME = <structured-name 1..8>

,**TYPE** = ***DEF**AULT / ***ALL** / <alphanum-name 1..8 with-wild(20)>

Operands

LIBRARY = *STD / <filename 1..54 without-vers> / *LINK(...)
Specifies the library containing the type whose attributes are to be displayed.

LIBRARY = *STD
The library opened globally is used.

LIBRARY = <filename 1..54 without-vers>
Name of the library whose type attributes are to be displayed.

LIBRARY = *LINK(...)
The type attributes of a library assigned via a link name are displayed.

LINK-NAME = <structured-name 1..8>
Link name of the library; this name was defined by means of /ADD-FILE-LINK prior to calling LMSCONV.

TYPE = *DEFAULT / *ALL / <alphanum-name 1..8 with-wild(20)>
Member type whose attributes are to be output. If the LMSCONV default value is set to *NONE, *DEFAULT has the same effect as *ALL.

> If TYPE=* is specified, the statement outputs the attributes for all explicitly declared types and for types which are embodied by existing members.

Example

The type attributes for member type S are set and then displayed. The library used is the globally set library.

```
//show-type-attributes type=s
INPUT  LIBRARY= :X:$USER.TESTLIB
TYPE       = S
SUPER-TYPE = *NONE
BASE-TYPE  = S
CONVENTION = *NONE
INIT-ELEM-P= *NONE
ADMINISTRAT= *NONE
STORAGE    = *NONE         WR-CONTROL = *NONE
```

7.4.5.11 SHOW-USER-EXITS - Display LMSCONV version

This statement displays the active user exits.

Format

SHOW-USER-EXITS

Operands

This statement has no operands.

7.4.5.12 WRITE-COMMENT - Write comments to output medium

This statement is used to write comments to the output medium defined by the statement MODIFY-LOGGING-PARAMETERS TEXT-OUTPUT=..., unlike the SDF standard statement WRITE-TEXT, which always outputs to SYSOUT or SYSLST.

Format

WRITE- **COMMENT**
COMMENT = "BLANK" / <c-string 1..1024 with-low>

Operands

COMMENT = "BLANK" / <c-string 1..1024 with-low>
Comment text. If nothing is specified, a blank "BLANK" is used as the default value, i.e. a blank line is generated.

7.5 Example: Modifying a link load module

```
/start-lmsconv                                                          1.
%  LMC0310 LMSCONV VERSION '<version>' STARTED
//modify-logging-parameters logging=*maximum                            2.
//open-library library=tst.bib,mode=*update                             3.
//modify-element (element=tst.ele,type=l)                               4.
//modify-modification-defaults csect=test                               5.
//add-text-modification address=x'e0',
       new-contents='ZZ'(old-contents='aa')                             6.
//end-modify                                                            7.
INPUT  LIBRARY= :2OSX:$USER.TST.BIB
OUTPUT LIBRARY= :2OSX:$USER.TST.BIB
INPUT  ELEMENT= (L)TST.ELE/031(0001)/<date>
OUTPUT ELEMENT= (L)TST.ELE/031(0002)/<date>
TEXT-ADR:            000000E0
TEXT BEFORE CHANGE:  a a
8181
TEXT AFTER  CHANGE:  Z Z
E9E9
CORRECT (L)TST.ELE/031(0001)/<date> AS
(L)TST.ELE/031(0002)/<date> , OUTPUT REPLACED
//END                                                                   8.
```

1. LMSCONV is called.
2. The entire LMSCONV log is output.
3. The TST.BIB library is assigned. It must be opened for reading and writing; otherwise, no modification of the link load module is possible.
4. The link load module TST.ELE is to be modified.
5. This substatement specifies the CSECT to be corrected.
6. This substatement replaces the original text “aa” at address E0 of the CSECT TEST with the replacement text 'ZZ'.
7. The input of the substatement is terminated. The substatements are then executed.
8. LMSCONV is terminated.

7.6 Comparison between LMSCONV and LMS

Global restrictions in LMSCONV compared to LMS:

- No creation of delta-stored members
- No structured output, i.e. no output to S variables
- No modification of library and type attributes
- No support for a start file

The following LMS statements are not supported:

```
ACTIVATE-USER-EXIT
BEGIN-MAKE
CALL-EDT
COMPARE-ELEMENT
DEACTIVATE-USER-EXIT
EDIT-ELEMENT
EXECUTE-SYSTEM-COMMAND
FIND-ELEMENT
MODIFY-ELEMENT-PROTECTION
MODIFY-LIBRARY-ATTRIBUTES
MODIFY-TYPE-ATTRIBUTES
PROVIDE-ELEMENT
REORGANIZE-LIBRARY
RESET-Anweisungen
RETURN-ELEMENT
SHOW-STATISTICS
```

The following table shows the syntax restrictions compared to LMS:

Statement	Restriction
ADD-ELEMENT	No *HIGH,*INCR,BASE,STORAGE-FORM for TO-ELEMENT No PROTECTION
CLOSE-LIBRARY	./.
COPY-ELEMENT	No selection via BASE for ELEMENT No EXCEPT No *HIGH,*INCR,BASE,STORAGE-FORM for TO-ELEMENT No PROTECTION
COPY-LIBRARY	./.
DELETE-ELEMENT	No selection via BASE for ELEMENT No EXCEPT
END	./.

EXTRACT-ELEMENT	No selection via BASE for ELEMENT No EXCEPT No INFORMATION No PROTECTION

Statement	Restriction
MODIFY-DEFAULTS	No STORAGE-FORM for ELEMENT-ATTRIBUTES No DELTA-STRUCTURE for INFORMATION No PROTECTION No COMPARE-PARAMETERS
MODIFY-ELEMENT	No selection via BASE for ELEMENT No EXCEPT No *HIGH,*INCR,BASE for TO-ELEMENT
MODIFY-ELEMENT-ATTRIBUTES	No selection via BASE for ELEMENT No EXCEPT No selection via STATE No WRITE-MODE Only CODED-CHARACTER-SET exists for NEW-ATTRIBUTES
MODIFY-LOGGING-PARAMETERS	No EDT(...) for TEXT-OUTPUT No BASE for ELEMENT
OPEN-LIBRARY	./.
SHOW-DEFAULTS	No PROTECTION No COMPARE-PARAMETERS
SHOW-ELEMENT	No selection via BASE for ELEMENT No EXCEPT
SHOW-ELEMENT-ATTRIBUTES	No selection via BASE for ELEMENT No EXCEPT No selection via ACCESS-DATE/TIME No selection via STATE No selection via STORAGE-FORM No selection via ELEMENT-SIZE No selection via PROTECTION No selection via DELTA-STRUCTURE for INFORMATION No *BY-CREATION-DATE, *BY-MODIFICATION-DATE, *BY-ACCESS-DATE, *BY-ELEMENT-SIZE for SORT No TEXT-OUTPUT No STRUCTURE-OUTPUT
SHOW-LIBRARY-ATTRIBUTES	No TEXT-OUTPUT No STRUCTURE-OUTPUT
SHOW-LIBRARY-STATUS	./.
SHOW-LOGGING-PARAMETERS	./.

Statement	Restriction
SHOW-TYPE-ATTRIBUTES	No TEXT-OUTPUT No STRUCTURE-OUTPUT
SHOW-USER-EXITS	./.
WRITE-COMMENT	./.

Converting from LMSCONV to LMS

All LMSCONV statements and operands are valid in LMS, provided that the full statement names or the guaranteed abbreviations are used. Thus, conversion from LMSCONV to LMS presents no problem to the user.

8 MSGMAKER Processing of BS2000 Message Files

Version: MSGMAKER V1.2B

MSGMAKER offers the following functions:

- Create message file
- Build a message file from message units
  - Enter new message units (ADD-MSG function)
  - Modify existing message units (MODIFY-MSG function)
  - Delete existing message units (DELETE-MSG function)
  - Display the contents of a message file (SHOW function)
  - Copy various components of a message file (message units, documentation lines, component identification) from one message file to another or to a different location within the same file. The source area of the message file is either retained (COPY function) or deleted (MOVE function)
  - Document messages of a designated message range (ADD-DOCUMENTATION function)
  - Modify existing documentation lines (MODIFY-DOCUMENTATION function)
  - Delete existing documentation lines (DELETE-DOCUMENTATION function)
  - Message texts, meaning texts and response texts can be defined in up to eight languages
  - Variable text sections (inserts) can be defined in the message text. Each insert is assigned a number and a name and may be assigned a default text
- Merge the contents of several message files into one message file

> Information on the “old” format (MSGEDIT/MSGLIB) of the message file has been removed from this MSGMAKER description. However, the “old” format can still be used for reasons of compatibility. The MSGEDIT and MSGLIB programs have been withdrawn.

MSGMAKER is controlled

- by means of statements (batch jobs and procedures)
- interactively by means of screen masks (menu mode)
  It is also possible to enter statements in the command area of the mask (with the exception of the //OPEN-MSG-FILE statement).

The following overview lists the functions provided by MSGMAKER and the statements and screen masks that implement them.

Function	Mask	Statement
Modify existing message units	MODIFY-MSG	//MODIFY-MSG
Modify existing documentation lines	MODIFY-DOCUMENTATION	//MODIFY-DOCUMENTATION
Display contents of a message file	SHOW	//SHOW
Add or modify a meaning text	MEANING /RESPONSE	MEANING operand of the //ADD-MSG and //MODIFY-MSG statements
Add or modify response text	MEANING /RESPONSE	RESPONSE operand of the //ADD-MSG and //MODIFY-MSG statements
Add or modify message text	MSG-TEXT	MSG-TEXT operand of the //ADD-MSG and //MODIFY-MSG statements
Create message file	MSG-FILE-ATTRIBUTES	//OPEN-MSG-FILE
Add new message units	ADD-MSG	//ADD-MSG
Add documentation lines	ADD-DOCUMENTATION	//ADD-DOCUMENTATION
Define inserts	INSERT-ATTRIBUTES	INSERT-ATTRIBUTES operand of the //ADD-MSG and //MODIFY-MSG statements
Copy various components of message file	COPY	//COPY
Copy and delete various components of a message file	MOVE	//MOVE
Delete existing message units	DELETE-MSG	//DELETE-MSG
Delete documentation lines	DELETE-DOCUMENTATION	//DELETE-DOCUMENTATION
Merge message files	- - -	//MERGE-MSG-FILES

8.1 Execution of MSGMAKER

It can be installed under any desired user ID with the aid of the installation monitor IMON.

The masks and messages of MSGMAKER are output in the language set for the system component MIP.

The command `/MODIFY-MSG-ATTRIBUTES TASK-LANGUAGE=<lang>` is used to specify the output language: English (TASK-LANGUAGE = E) or German (TASK-LANGUAGE = D).

This command cannot be entered in the command area of the masks; it may be issued only at command level (/MOD...). If no task language is defined before the routine is started, MSGMAKER uses the system default value.

8.1.1 Starting the routine

The MSGMAKER routine is started using the /START-MSGMAKER command.

START-MSGMAKER	Alias: **MSGMAKER**
VERSION = ***STD** / <product-version>	
,MONJV = ***NONE** / <filename 1..54 without-gen-vers>	
,CPU-LIMIT = ***JOB-RE**ST / <integer 1..32767 *seconds*>	

MSGMAKER can also be called with:

```
/START-EXECUTABLE-PROGRAM FROM-FILE=*LIB-ELEM(
                          LIBRARY= $<userid>.SYSLNK.MSGMAKER.<version>,
                            ELEMENT=MSGMAKER)
```

<version> is the current version of MSGMAKER, e.g. 012.
When the routine is started, the operands CPU-LIMIT, TEST-OPTIONS, MONJV, RESIDENT-PAGES and VIRTUAL-PAGES of the /START-EXECUTABLE-PROGRAM command are also available to the user, e.g. to be able to monitor the program run. For a description of these operands, see the /START-EXECUTABLE-PROGRAM command in the “Commands” manual [1 (Related publications)].

The routine runs under any user ID.

MSGMAKER has no subroutine interface, i.e. it cannot be called from another program.

8.1.2 Defining a monitoring job variable

A monitoring job variable (MONJV) can be defined when the MSGMAKER routine is started.

While the routine is executing, the operating system sets the status indicator (the first three bytes of the job variable) to the value

- $R: The routine is running.

Once the routine has terminated, the status indicator can assume the following values:

- $T: The routine was terminated successfully.
- $A: The routine was terminated abnormally due to a program error or a defined error exit.

On termination of the routine, one of the following values is assigned to the fourth byte of the job variable:

- 0: The routine executed without errors or warnings.
- 1: The routine executed without errors but with warnings.
- 3: The routine executed with errors.

MSGMAKER does not assign a value to bytes five through seven of the job variable.

For further information on monitoring job variables, see the “JV” manual [16 (Related publications)].

8.1.3 MSGMAKER operating modes

There are two operating modes for the MSGMAKER routine:

1. Interactive mode, referred to below as menu mode (see "Menu mode").
   The routine is controlled by means of screen masks or statements entered in the command area of the masks.
2. Batch and procedure mode - see "Statements". This includes batch mode, batch procedure mode and interactive procedure mode.
   The routine is controlled by means of statements.

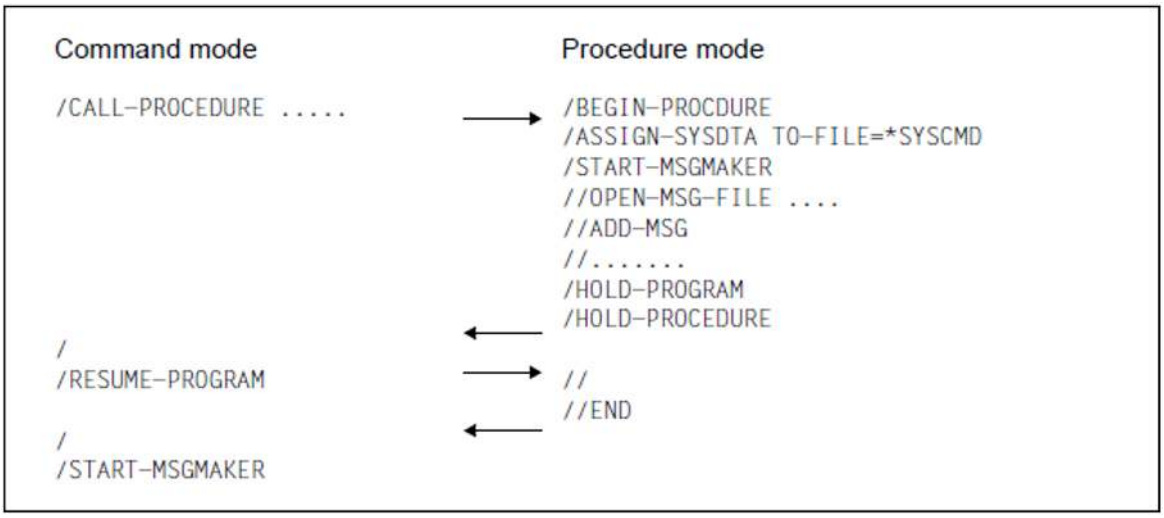
The MSGMAKER operating mode is defined by the call environment when the routine is started. You cannot change the operating mode while the routine is running. To be able to use the routine in another mode, you must enter /START-MSGMAKER again. This is illustrated in the following example.

Example

Let us assume the following situation:
MSGMAKER has been called in the middle of a procedure. The MSGMAKER statements are also located in this procedure. MSGMAKER is therefore also started in procedure mode. The routine or procedure is interrupted by /HOLD-PROGRAM or /HOLD-PROCEDURE. You are now at command level.

If you return to the routine using /RESUME-PROGRAM, MSGMAKER waits for a statement to be input (you are shown the prompt //). You are back in guided dialog. Although procedure mode is no longer active, this does not change the fact that the routine has been called in a procedure.

When this occurs, it is best to abort with K2 or terminate the MSGMAKER routine with the END statement. By restarting the routine, this time at command level, you will be able to use MSGMAKER in menu mode.



8.1.4 Special character sets

In menu mode, messages can be created with special characters from extended character sets (e.g. 8-bit character sets). The relevant library must be assigned for the screen display before MSGMAKER is started.

```
/ADD-FILE-LINK LINK-NAME=MAPLIB,FILE-NAME=SYSFHS.MSGMAKER.<version>.<code>
```

`<version>` contains the current MSGMAKER version and `<code>` the name of the required character set.

The terminal must also be working in 8-bit mode. The 8-bit code table of the user entry is activated by means of the `/MODIFY-TERMINAL-OPTIONS CODED-CHARACTER-SET=8-BIT-DEFAULT` command.

8.1.5 Messages of MSGMAKER

The messages of the MSGMAKER utility routine have the message class MSM.

See also the section “Messages and their meaning” (Notational conventions).

8.2 Menu mode

In menu mode, MSGMAKER provides one or two masks for each function. In each mask, the user can

- select a function by entering alphabetic or numeric characters or marking it with “X” and start the function by pressing the DUE key
- initiate control functions by pressing the function keys F2, F3, K1, K2 and K3
- deviate from the defined processing sequence of the masks and call any function by entering statements in the command area. Selective control of the individual masks enables the user to move through the masks more quickly, thus reducing the time taken to process a message file (see also "//GO-TO - Branch to specified mask").
- call any statement with the help of SDF. SDF is in EXPERT mode; if you enter a question mark, SDF displays all the available MSGMAKER statements.

8.2.1 Mask overview

Function	Brief description
ADD-DOCUMENTATION	Add documentation (person responsible, comment) for messages of a specific range
ADD-MSG	Add message units
COPY	Copy message units and documentation lines
DELETE-DOCUMENTATION	Delete documentation (person responsible, comment) for messages of a specific range
DELETE-MSG	Delete message units
INSERT-ATTRIBUTES	Add, modify or delete insert attributes
MEANING/RESPONSE	Add, modify or delete meaning and response text
MODIFY-DOCUMENTATION	Modify documentation (person responsible, comment) for messages of a specific range
MODIFY-MSG	Modify message units
MOVE	Copy and delete message units and documentation lines
MSG-FILE-ATTRIBUTES	Add or modify message file attributes
MSG-TEXT	Add, modify or delete message text (and inserts)
SHOW	Display the contents of a message file (message unit and documentation lines)

8.2.2 Mask sequence

The figure 11 shows the order in which the masks appear when the user enters certain operation numbers or presses certain function keys. The sequence in which the masks appear is governed by the mask hierarchy:

- The main mask MENU appears as soon as MSGMAKER is called.
- The MSG-FILE-ATTRIBUTES, ADD-MSG, MODIFY-MSG, DELETE-MSG, COPY, MOVE, SHOW, ADD-DOCUMENTATION, MODIFY-DOCUMENTATION and DELETE-DOCUMENTATION masks are called from the main mask MENU. The SHOW mask allows the user to select the message units to be displayed; a second mask with the name SHOW-OUTPUT is called to display them. Similarly, the DELETE-MSG mask allows the user to select the message unit to be deleted. A second mask with the same name then displays this message unit before the DELETE function is initiated.
- The MSG-TEXT, MEANING-RESPONSE and INSERT-ATTRIBUTES masks follow the ADD-MSG and MODIFY-MSG masks automatically

The numbers in the overview indicate the functions selected in the MENU mask.

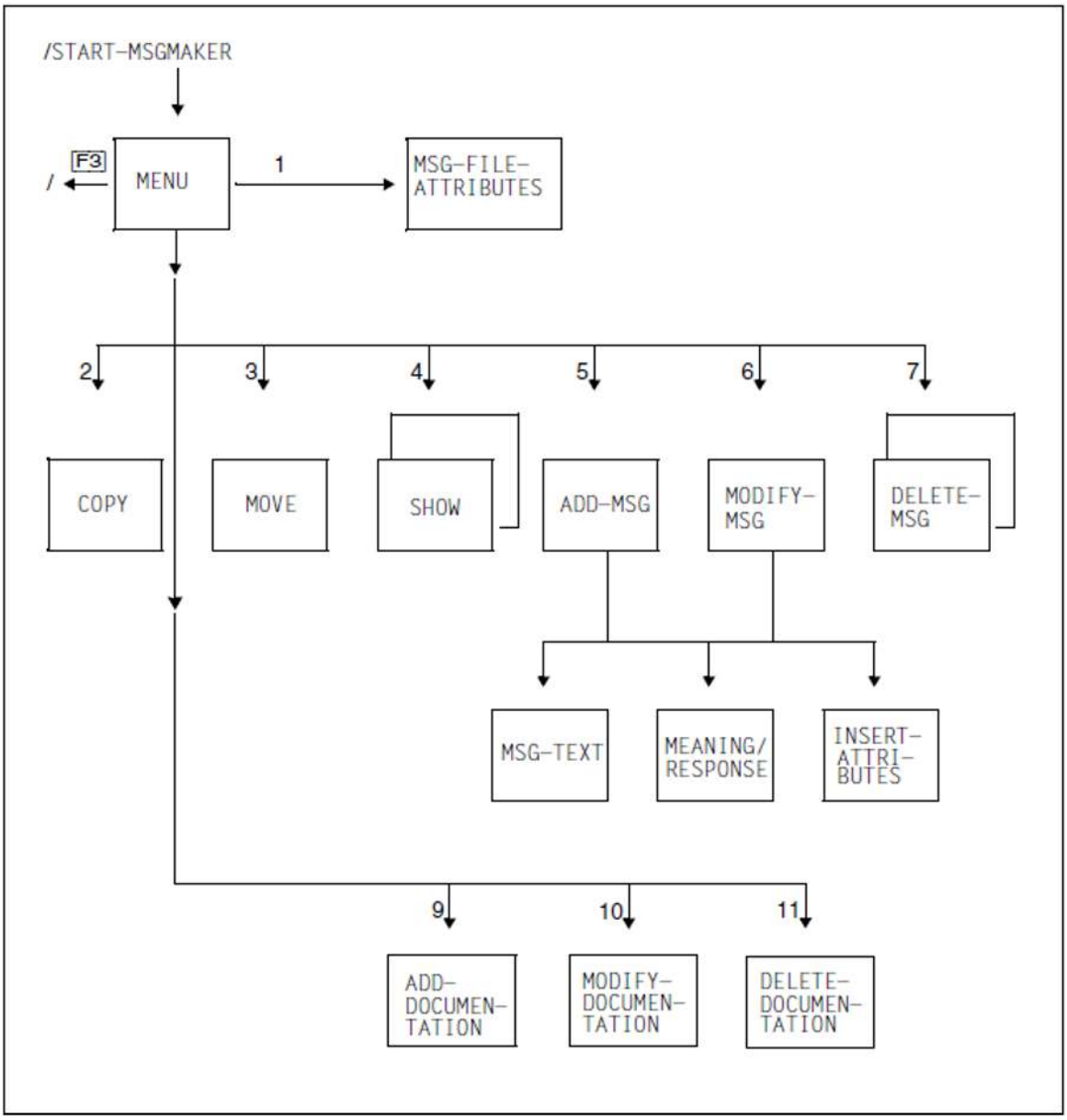
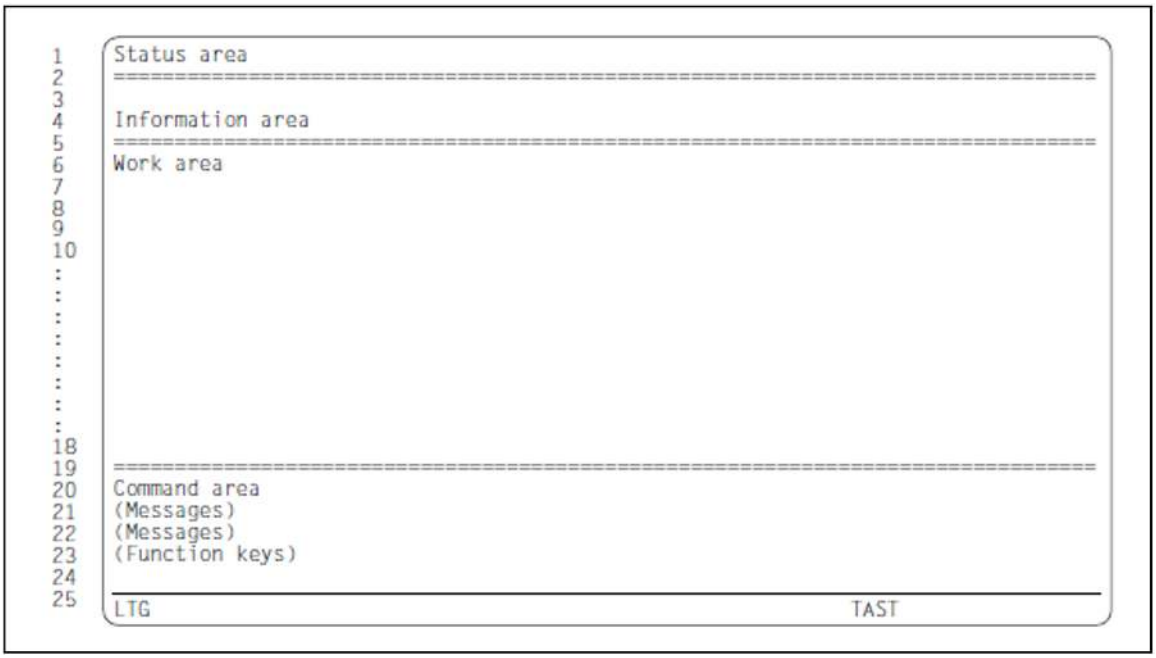


Figure 11: Overview of MSGMAKER masks

8.2.3 General mask format



Status area

The status area contains the mask title that describes the function of the mask. The statement that corresponds to this mask also has this name.

Exceptions:
There are no statements called MENU or MSG-FILE-ATTRIBUTES. The functions of these two masks are combined in the statement //OPEN-MSG-FILE.

Information area

The information area provides any information that is available on the message file or message unit currently being processed (message code, language, etc.).

Work area

The work area is the area in which the user performs operations. In this area, the user can select functions or enter values.

Command area / message output area

In the three-line command area, the user can enter a statement or the character “+” or “-”. The function key assignments are displayed in the last line of the mask.

The second and third lines of the command area also serve as the output area for messages issued by the routine or by the system.

More (- +)

- **+** In each of the masks MSG-TEXT, INSERT-ATTRIBUTES and SHOW-OUTPUT, a second mask can be called by entering “+”. The second mask displays all the remaining message texts or inserts that have been defined.

- The user can enter “-” to return to the first mask.

Command ==>

The user can enter statements, as described in the sections “Statements” and “Special features of statements in menu mode”.

> i A statement entered in the command area is always executed before the mask function displayed. Exception: in the MENU mask.

If a complete statement is entered, the function is executed immediately in the background.

In the case of an incomplete statement, i.e. one in which all or some of the operands are missing, the mask that corresponds to the statement is displayed. Operands that have already been assigned values in the statement call are transferred to the fields in the mask. Missing operands can be added in the mask. The user then presses the DUE key to execute the statement and return to the calling mask. The command area of the mask is now empty and all other mask areas are as they were before the statement was entered.

If the prompt function is applied to a statement, the statement is not executed immediately; Instead the mask that corresponds to the statement is displayed. The user is able to add any missing operand values. Finally, the user presses the DUE key to execute the statement. For more information, see the section “Function keys”.

The //GO-TO statement allows the user to call any mask directly (see "GO-TO - Branch to specified mask").

Combined input of “+”/ “-” and statements is not possible.
"Command ==>" appears in the command area only if it is not possible to call a corresponding mask.

If a question mark “?”is entered in the command line and then the DUE key is pressed, a selection menu with MSGMAKER statements and SDF standard statements appears. By entering the number preceding the statement in the input field NEXT, the operand form associated with the statement is displayed. Entering *CANCEL, *EXIT or *EXIT-ALL returns to the last displayed mask function.

Function keys

The function keys F2, F3, K1, K2 and K3 allow the user to perform operations easily and quickly.

F2 = prompt

provides extensive information on a statement that has been entered in the command area.
Pressing the F2 key after making an entry in the command area initiates the prompt function. If the prompt function is applied to a statement, the mask that corresponds to the statement is called and the operands that have already been specified are transferred to the mask. If all the operands have been entered, the user presses the DUE key to execute the statement and return to the calling mask. If some or all of the operands are missing, pressing the DUE key causes MSGMAKER to branch to the appropriate masks so that the user can enter the missing operands. MSGMAKER returns to the calling mask.

The statement is no longer displayed in the mask; all other mask areas are as they were before the prompt function was called.

F3 = Exit

The user presses the F3 key to exit the current function. The function is **not** executed and any entries made are lost.

Function of F3

ADD-MSG MODIFY-MSG DELETE-MSG COPY MOVE SHOW MSG-TEXT INSERT-ATTRIBUTES	F3 = exit mask
MENU	F3 = end MSGMAKER

The F3 key is **not** available in the following masks:

- MEANING/RESPONSE
- MSG-FILE-ATTRIBUTES
- ADD-DOCUMENTATION
- MODIFY-DOCUMENTATION
- DELETE-DOCUMENTATION

In SDF guided mode, F3 causes a statement to be executed immediately.

K1 = cancel/skip

returns the user to the previous mask. All entries made in the mask between DUE and pressing the K1 key are lost. (K1 = cancel)
If several message units are being processed (in the ADD-/MODIFY-/DELETE-MSG masks), K1 = skip allows the user to exit the message unit currently being processed and proceed to the next. As in the case with K1 = cancel, any entries made since the last DUE and pressing the K1 key are lost.
If the skipped message unit is the last one within a set of messages being processed, the function (ADD-/MODIFY-/DELETE-MSG) is terminated.

K2 = interrupt

interrupts the MSGMAKER routine and switches to BS2000 command mode. MSGMAKER can be resumed by means of the /RESUME-PROGRAM command.

K3 = refresh

saves the mask contents that were committed the last time the DUE key was pressed. All entries made in the mask between DUE and K3 since then are lost.

ℹ Entering a question mark in the command line causes MSGMAKER to switch to guided SDF interactive mode. During guided mode the function key assignment set for SDF applies.

Message output area

This area is used to display error messages, warnings and information which can overwrite the bottom one or two lines of the command area. The final part of a long statement may be lost.

Messages issued by MSGMAKER start with the message class MSM. If a warning is output, the message text starts with “WARNING”; in the case of an error message, the cursor appears in the first incorrect mask field.

8.2.4 Making entries in masks

Each input field is identified by a name, which appears to the left of the field. The name of input and output fields in this part of the manual is in **bold**.

The user can make the following entries in the fields:

Input of	Example	Input in field
Text	Mask: MENU	**Work message file**: testfile
Letters	Mask: MENU	**Open mode**: U
Numbers	Mask: MENU	**Select operation**: 5
“X” character	Mask: MOVE	**Information**: X messages

The inputs into the fields have to be neither left- nor right-aligned; however, any gaps should be filled with blanks or null characters.

Symbolic names <name> or actual specifications to the right of the input field tell the user what entries can be made in the field.

Example

(COPY mask)

Field: **First msg-id**: .....(<msg-id> / <partial msg-id>* / *=all)

The “X” character is used to mark and select one or more items in a list.

The user can cancel a selection by overwriting the marked fields with a blank or null character.

Example

(COPY mask)

Field: **Information**: X messages

X documentation

component identification

8.2.5 Description of fields that occur frequently

First msg-id and **Last msg-id**

A message unit cannot be processed until the message code has been specified. There are various ways of selecting one or more message units.

Explicit specification of the message code

The user enters a full seven-character message code, e.g. AAA0001, in the **First msg-id** field.
If the **Last msg-id** field also contains a full message code, e.g. AAA0005, a range with the lower limit AAA0001 and the upper limit AAA0005 is defined.
The four-character message number can consist of both digits and letters. Note that letters appear before digits in a sequence, e.g. CCC01AA, CCC01AB, ..., CCC01AZ, CCC01A0, ..., CCC01A9.

Implicit specification of the message code

A message code that begins with an invariable section and ends in one of the wildcard characters * or # is entered in the **First msg-id** field. The asterisk (*) indicates that the message code can end in letters or digits; the # character (which may be specified for the ADD-MSG and MODIFY-MSG functions) stands for digits only.
A full message code is entered in the **Last msg-id** field to specify the upper limit of a message range. Any partially qualified message code specified in this field is ignored.

Examples

CCC*	stands for	CCCAAAA,	CCCAAAB,	...,	CCCAAAZ
		CCCAAA0,	CCCAAA1,	...,	CCCAAA9
		CCCAABA,	CCCAABB,	...,	CCCAABZ
		...			CCC9999
CC*	stands for	CCA...,	CCB...,	...,	CCZ...
*	stands for	selects all message units			
CCC01A#	stands for	CCC01A0,	CCC01A1,	...,	CCC01A9
CCC#	stands for	CCC0000,	CCC0002,	...,	CCC0009
		CCC0010,	CCC0011,	...,	CCC0019
		CCC9990,	CCC9991,	...,	CCC9999
CC# or C# or # not possible					

Special keywords

(same)

The word (same) is displayed for information purposes in the **Last msg-id** or **To msgid** field. If the user accepts (same) as the field contents or overwrites it with blanks or null characters, the first and last message codes are identical (**Last msg-id**: (same)) or the message codes are not renamed (**To msg-id**: (same)).

(list)

The word (list) is displayed in the **First msg-id** and **Last msg-id** fields if a statement contains several message codes that cannot be grouped together in a range. (list) is displayed for information purposes only and cannot be entered in a field.

Language(s)

The word (list) is displayed in the **Language(s)** fields if a statement contains several message codes that cannot be grouped together in a range. Each language is identified by a single letter. The letter D is used for German and E for English. All other languages (up to eight) may have any abbreviation.
No entry means that the entire message unit is to be processed, including the message attributes and texts in all languages.

Message file

This field contains the name of a message file.
Usually the name in this field is the currently open message file; in some masks this default can be overwritten.

8.2.6 Description of the masks

- MENU mask - MSGMAKER main mask
- MSG-FILE-ATTRIBUTES mask - Enter and modify message file attributes
- COPY mask - Copy message units
- MOVE mask - Copy and delete message units
- SHOW mask - Display message file contents
- SHOW-OUTPUT mask - Output message units and additional information
- ADD-MSG mask - Add message unit
- MODIFY-MSG mask - Modify message unit
- MSG-TEXT mask - Add or modify message text
- MEANING/RESPONSE mask - Add or modify meaning and response text
- INSERT-ATTRIBUTES mask - Add or modify insert attributes
- DELETE-MSG mask - Delete message unit
- ADD-DOCUMENTATION mask - Add documentation lines
- MODIFY-DOCUMENTATION mask - Modify, add and delete documentation lines
- DELETE-DOCUMENTATION mask - Delete documentation lines

8.2.6.1 MENU mask - MSGMAKER main mask

Mask sequence

/START-MSG-MAKER
|
|

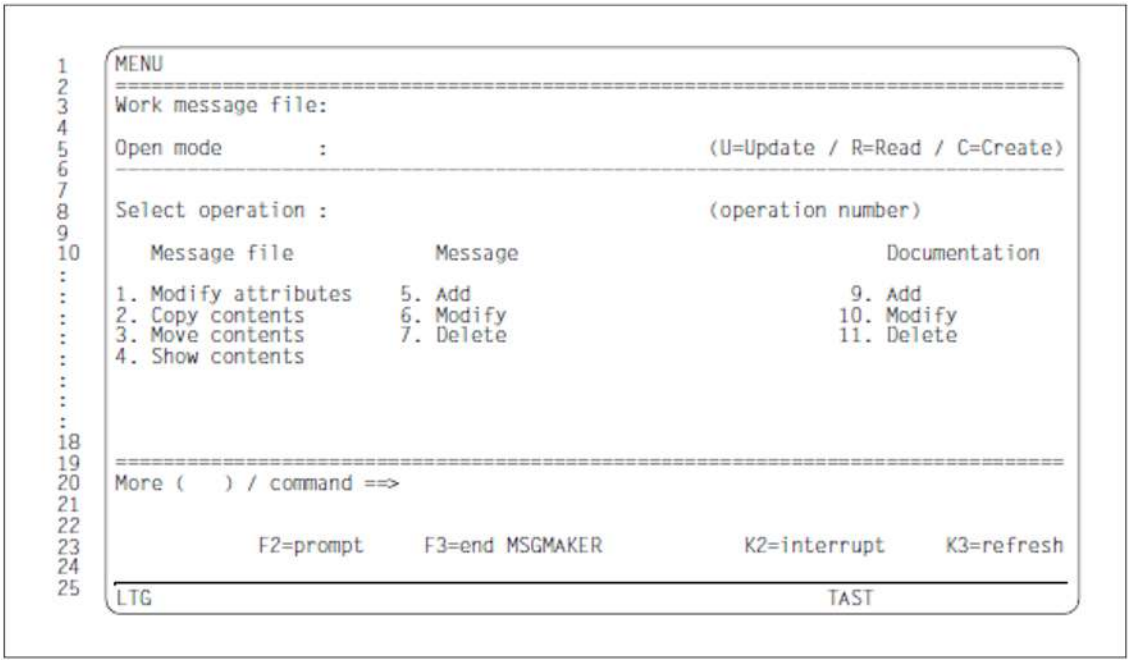
MENU

This mask is available to the user as soon as the MSGMAKER utility routine has been called.

Function

In this mask, the user enters the message file to be processed. At the same time, the user can specify a number to select the next operation to be performed on this file.
Operations 1, 5, 6, 7, 9, 10 and 11 cannot be performed unless a message file is specified. Operations 2, 3 and 4 do not require a message file to be entered in the MENU mask.

Mask



Input fields

Message file

Name of a message file. If the input is confirmed with DUE, the fully qualified name appears in the message file.

If a new message file is to be opened, the name of the file that is already open must be overwritten and confirmed with DUE.

Validity criteria:
Data type: <filename 1..54>

Open mode

U (Update): Opens an existing message file so that it can be updated.

R (Read): Opens a message file for read access only.

C (Create): Creates a new message file which becomes the current work file. The MSG-FILE-ATTRIBUTES mask (see "MSG-FILE-ATTRIBUTES mask - Enter and modify message file attributes") is called automatically; in it, the user can enter the attributes of the new message file.

If the name of a file that does not yet exist is specified together with U, MSGMAKER outputs error message `MSMDJ01`.

Select operation

The user can select one of the following operations by entering the appropriate number:

1 (Message file - modify attributes)

calls the **MSG-FILE-ATTRIBUTES** mask.

This mask allows the user to modify the attributes of the message file.

File attributes include

- the file type; a distinction is made between customer message files and the BS2000 standard message files
- the name of the software product for which the message file is created
- the version number of this software product

> ¡ If a new message file is created (C = create), the MSG-FILE-ATTRIBUTES mask is called **automatically**. The user must enter 1 in order to modify the attributes of an existing message file.

2 (Message file - copy contents)

calls the **COPY** mask.

This mask allows the user to select components of a message file via their message codes and copy them from one file to another or to a different location within the same file. The message file currently open does not have to be the source or target file.

The components of a message file include

- Message unit(s)
- Documentation lines
- component identification and correction information (internal)

3 (Message file - move contents)

calls the **MOVE** mask.

This mask allows the user to select components of a message file via their message codes and move them from one file to another or to a different location within the same file. Unlike the COPY function, the MOVE function deletes the source area.
The message file currently open does not have to be the source or target file.

The components of a message file include

- Message unit(s)
- Documentation lines
- component identification and correction information (internal)

4 (Message file - show contents)

calls the **SHOW** mask.

This mask allows the user to select components of a message file via their message codes and output them to SYSOUT or to a SYSLST file.

The components of a message file include

- message unit(s), subdivided into
  - Message attributes
  - message text in the defined languages
  - meaning and response text in the defined languages
  - inserts and insert attributes
- Documentation lines
- component identification and correction information (internal)

5 (Message - Add)

calls the **ADD-MSG** mask.

This mask allows the user to add new message units to the current work file. The message codes determine the order in which the new message units appear in the message file.

A message file can be assigned the following attributes:

- MIP access method
- Output destination
- Routing code
- Weight code
- Warranty
- Language identifier

For each defined language, the user must write a message text and may optionally write a meaning and response text. In addition, inserts may be defined in the message text.

6 (Message - modify)

calls the **MODIFY-MSG** mask.

This mask allows the user to modify message units in the current work file.

The following attributes of a message file can be modified:

- MIP access method
- Output destination
- Routing code
- Weight code
- Warranty
- Language identifier

In each defined language, the user can modify the message text, meaning text and response text. The same applies to the inserts and their attributes.

7 (Message - delete)

calls the **DELETE-MSG** mask.

This mask allows the user to delete message units from the current message file.
The message units are selected via their message codes.

The user can

- delete texts in different languages
- delete one or more message units
- display the message units in another mask of the same name before deleting them.

9 (Documentation - add)

calls the **ADD-DOCUMENTATION** mask.

Messages can be documented in this mask.

The following specifications are possible:

- Message code/message range
- name of the person responsible for the message
- name of the team responsible
- Comment

10 (Documentation - modify)

calls the **MODIFY-DOCUMENTATION** mask.

The documentation of messages can be modified in this mask.

The following specifications are possible:

- modification of the message code
- modification of the persons responsible for the message
- modification of the team responsible
- modification of the comment

Modified message codes are automatically sorted anew into the list.

11 (Documentation - delete)

calls the **DELETE-DOCUMENTATION** mask.

The documentation forone or more messages can be deleted in this mask.

Command

Special point:

If a message file has been entered in the **Work message file** field, this file is always opened before any statement is executed.

> A plus sign “+” is displayed in the command area if a standard message file of the manufacturer is opened. Hitting “+” scrolls to a second MENU mask which offers functions for internal use.

For further information on

- entering statements, see "General mask format"
- function key assignment, see "General mask format"

Follow-on operations:

MENU	-> ?

Operation no. 1 - 11: Branch to different input masks.

F3 End MSGMAKER.

8.2.6.2 MSG-FILE-ATTRIBUTES mask - Enter and modify message file attributes

Mask sequence

MENU -> 1 -> MSG-FILE-ATTRIBUTES

Function

The user can enter the message file attributes or modify them by overwriting existing values. By pressing the DUE key, the user confirms the input and exits the mask.

The user switches to the MSG-FILE-ATTRIBUTES mask by specifying

- a new file with **Open mode**: C or
- **Select operation**: 1 for an existing file (**Open mode**: U) in the MENU mask.

When the mask appears, the cursor is positioned in the **Type** field.

Mask

```
MSG-FILE-ATTRIBUTES
================================================================================



File name: :N:$USER0001.TESTFILE


Type     : C               (C=Customer: msg file attached to a customer product
                            S=Standard: msg file attached to an official
                                        product developed within SNI)


Product                    (product whose message file belongs to)
  name   : PROGRAM         (empty field = none)
  version: V1.2A           (empty field = none)


================================================================================
Command ==>

           F2=prompt                K1=cancel     K2=interrupt     K3=refresh

LTG                                                          TAST
```

Output fields

File name (name of the message file)

The name entered in the main mask MENU is transferred to the MSG-FILE-ATTRIBUTES mask. This name must not be modified.

Input fields

Type (type of message file)

If the message file is associated with a software product developed by the manufacturer, the file type S (standard) must be entered. The default value C (customer) means that the message file was written for a customer's own product.

> It is **not** possible to convert a standard message file into a customer message file by changing the S entry to C. However, the user may open a new message file of type C and copy the contents of the standard message file into this new file. A customer message file can be converted into a standard message file without restriction. Default setting/display: C (customer)

Product name (name of the software product)

Name of the software product for which the message file is created.

Validity criteria:
Data type: <structured-name 1..15>
A product name is required only if a version number is specified.
The name entered appears in uppercase letters within the message file and the next time the mask is called.

Product version (version number of the product)

Version number of the software product for which the message file is created.

Validity criteria:
<composed-name 3..8> or <c-string 1..8>
If the version number contains letters, these are converted into uppercase letters within the message file and the next time the mask is called.
If the version number is specified, a product name is also required.

Command

For detailed information on

- entering statements, see "General mask format"
- function key assignment, see "General mask format"

Follow-on operations:

MSG-FILE-ATTRIBUTES	-> ?

DUE	Inputs are confirmed; return to the MENU mask.
K1	Inputs are not confirmed and are lost; return to the MENU mask.

8.2.6.3 COPY mask - Copy message units

Mask sequence

MENU -> 2 -> COPY

Function

This mask allows the user to select components of a message file by means of their message codes and copy them to another file or to a different location within the same file. The current message file does not have to be the source or target file.

It is possible to call the COPY mask directly from MENU without entering a file name in the main mask. If a message file name is entered in MENU, this name is transferred to the **From file** and **To file**fields of the COPY mask.

The COPY function is executed when the inputs are confirmed with DUE. If the copying procedure is successful, MSGMAKER outputs message `MSMN100`.

Special points concerning the copying procedure

- If a specified target message unit does not yet exist, a message unit is created with the message and insert attributes of the source message unit.
- If the target message unit already contains a text with the same message code and language identifier, the text of the target message unit is overwritten with the text of the source message unit.
- If the only difference between the texts is the language identifier (e.g. ABC0000D, ABC0000E), the text is appended to the contents of the target message unit.
- If, as a result of the copying procedure, the target message unit contains a new, undefined insert number, the corresponding insert attributes of the source unit are copied to the target message unit. Conversely, the insert attributes of inserts that no longer exist are deleted.

> If the entire contents of a message file are copied to another file, it is better to use the //MERGE-MSG-FILE statement rather than the COPY function. The copying procedure is then actually much more efficient.

Mask

Input fields

First msg-id (first message code of the message range)

The specified message code identifies a message unit or the first message unit of a message range.

Validity criteria:
Seven-character message code or partially qualified message code in which the asterisk (*) can stand for between one and seven characters.
The first three characters must be letters (message class).

Example

> Valid specifications for <partial msg-id>* are:
> A*, AB*, ABC*, ABC0*, ABC00*, ABC000*

The entry * selects all defined message codes. For further information on message codes, see the “Introduction to System Administration” [5 (Related publications)].

Last msg-id (last message code of the message range)

The specified message code identifies the last message unit of a message range. Only one message unit is selected if this message code matches the entry in the **First msg-id** field or the keyword (same) is transferred.

Validity criteria:
Seven-character message code; if the field is empty, the message code is transferred from the **First msg-id** field.
For further information on message codes, see the “Introduction to System Administration” [5 (Related publications)].

Display: (same)

Information

Message units and documentation lines can be selected from the defined message range and copied to another file or to a different location within the same file.

Validity criteria:
The file contents must be marked with the character “X”.

Display: X messages

Messages
Message units in the specified languages (see the **Lang** field) within the defined message range are copied. If the **Lang** field is empty, all message units are copied.

documentation
All documentation lines defined within the specified message range are copied. If other defined message ranges overlap the specified message range, their documentation lines are also copied.

Example

see the description of the //COPY statement.

component identification / correction information
This additional information is available only for BS2000 standard message files.

Language(s) (1-letter language identifier)

Entering the language identifier causes the texts (message, meaning and response) that are entered under the identifier to be copied to the specified target message unit. The target area may be located either in the source file or in another message file (see field **To file**).

Validity criteria: A letter from A through Z can be entered. To ensure correct selection, the letter entered here must match the identifier defined when the message unit was created.

From file (source file)

Name of the message file whose contents are to be copied. It is possible to overwrite the current message file, i.e. the file that was opened in the MENU mask and whose name was transferred to the **From file** field of the COPY mask. Any existing message file may be specified as the source file.

Validity criteria:
Data type: <filename 1..54>

To msg-id(s) (message code of the target file)

The message area reserved in the target file must be at least as large as the selected area of the source file, as specified in the **First msg-id** and **Last msg-id** fields.
Leaving the **To msg-id(s)** field empty has the same effect as the entry (same) and causes the message code of the source file to be transferred unchanged to the target file.
(same) must not be specified if the source file is also the target file. (same) **must** be specified if

- the **First msg-id** field contains
  - the keyword (list)
  - the entry *
  - a partially qualified specification containing the wildcard character * (e.g.: AB*)

- a message range across a number of message classes was defined in the **First msgid** and **Last msg-id** fields (e.g.: ABC0000 - ABC9999).

Validity criteria:
Seven-character message code or partially qualified message code in which the asterisk (*) can stand for between one and four characters.

Example

Valid specifications for <partial msg-id>* are: ABC*, ABC0*, ABC00*, ABC000*

The entry * selects all defined message codes. For further information on message codes, see the “Introduction to System Administration” [5 (Related publications)].

Display: (same)

To file (name of the target file)

Name of the message file to which the contents of the source file are to be copied. The current message file, i.e. the message file that was opened in the MENU mask and whose name was transferred to the **To file** field, can be overwritten in the COPY mask. Any existing message file can be specified as the target file.
If the message file specified does not yet exist, it is created. The file type (Customer or Standard), product name and version number are defined as for the source file.

Validity criteria:
Data type: <filename 1..54>

Command

For detailed information on

- entering statements, see "General mask format"
- function key assignment, see "General mask format"

Follow-on operations:

COPY	-> ?

DUE	The COPY function is started; the COPY mask is displayed again with its original status. The function can be called again by entering new values in the now empty “msg-id” fields.
F3	Quits the copy mask without the COPY function being executed. The original MENU mask is displayed.

8.2.6.4 MOVE mask - Copy and delete message units

Mask sequence

MENU -> 3 -> MOVE

Function

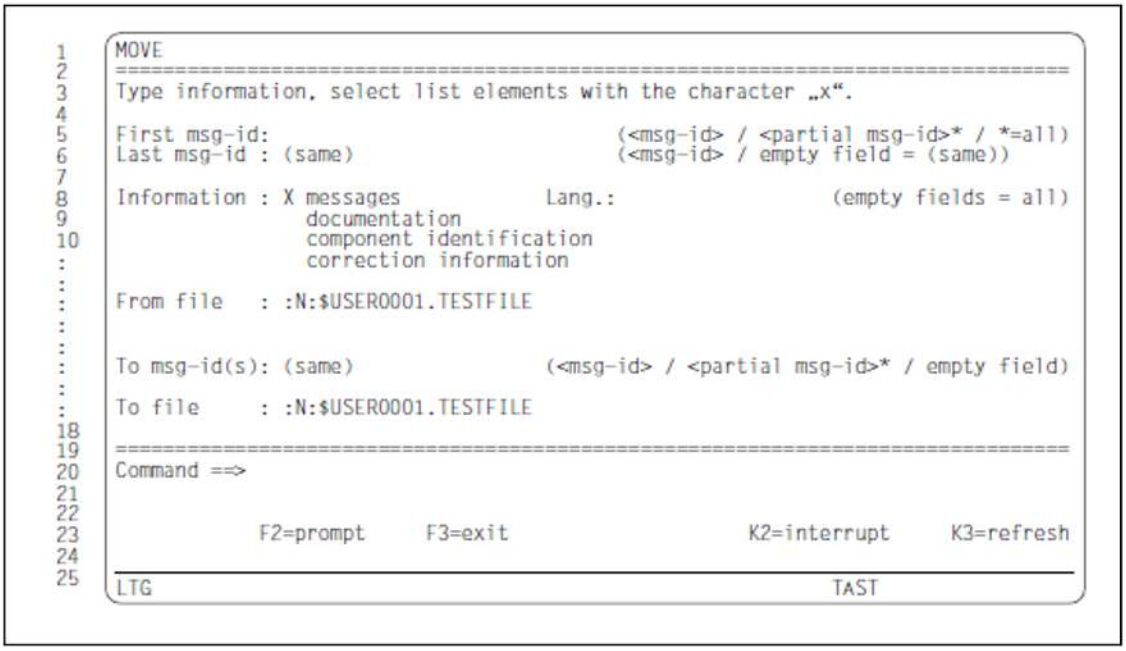
This mask allows the user to select message file components by specifying the message code and move them to another file or to a different location within the same file. Unlike the COPY function, the MOVE function deletes the source area. The current message file does not have to be the source or target file. The MOVE mask can be called directly from MENU without entering a file name. If a message file is entered in MENU, the file name is transferred to the **From file** and **To file**fields of the MOVE mask.

The MOVE function is executed when the input is confirmed with DUE. After a successful move procedure, MSGMAKER outputs message `MSMN100`.

Special points regarding transfer (copying + deletion)

- If the specified target message unit does not exist, a message unit is created with the message attributes of the source message unit.
- If the target message unit already contains a text with the same message code and language identifier, the text of the target message unit is overwritten with the text of the source message unit.
- If the only difference between the texts is the language identifier (e.g. ABC0000D, ABC0000E), the text is appended to the contents of the target message unit.
- If, as a result of transfer, the target message unit contains a new, undefined insert number, the corresponding insert attributes of the source unit are copied to the target message unit. Conversely, the insert attributes of inserts that no longer exist are deleted.
- If the MOVE function has deleted all the language-dependent parts of a message unit, the remaining message attributes and thus the message unit itself are deleted automatically.

Mask



Input fields

First msg-id (first message code of the message range)

The specified message code identifies a message unit or the first message unit of a message range.

Validity criteria:
Seven-character message code or partially qualified message code in which the asterisk (*) can stand for between one and seven characters.

Example

> Valid specifications for <partial msg-id>* are:
> A*, AB*, ABC*, ABC0*, ABC00*, ABC000*

The entry * selects all defined message codes. For further information on message codes, see the “Introduction to System Administration” [5 (Related publications)].

Last msg-id (last message code of the message range)

The specified message code identifies the last message unit of a message range. Only one message unit is selected if the message code matches the specification in the **First msgid** field or the keyword (same) is transferred.

Validity criteria:
Seven-character message code; if the field is empty, the message code is transferred from the **First msg-id** field.
For further information on message codes, see the “Introduction to System Administration” [5 (Related publications)].

Display: (same)

Information (selects the message file components that are to be moved)

Message units and documentation lines can be selected from the defined message range and copied to another file or to a different location within the same file. Unlike the COPY function, the MOVE function deletes the source area.

Validity criteria:
The file contents must be marked with the character “X”.

Display: X messages

Messages
Message units in the specified languages (see the **Lang** field) that are included in the defined message range are copied to the target area and deleted from the source area. If the **Lang** field is empty, all message units are copied.

documentation
All documentation lines defined in the specified message range are copied and then deleted from the source area. If other defined message ranges overlap the specified message range, their documentation lines are also copied and then deleted from the source area.

Example

See the description of the //MOVE statement.

component identification / correction information
This additional information is available only for BS2000 standard message files.

Language(s) (1-letter language identifier)

Entering the language identifier causes the texts (message, meaning and response) that are entered under the identifier to be copied to the specified target message unit. Once the texts have been copied, the source area is deleted.

Validity criteria:
A letter from A through Z can be entered. To ensure correct selection, the letter entered here must match the identifier defined when the message unit was created.

From file (source file)

Name of the message file whose contents are to be copied. The source area of this file will then be deleted. The current message file, i.e. the file that was opened in the MENU mask and whose name was transferred to the **From file** field, can be overwritten in the MOVE mask. Any existing message file may be specified as the source file.

Validity criteria:
Data type: <filename 1..54>

To msg-id(s) (message code of the target file)

The message area reserved in the target file must be at least as large as the selected area of the source file, as specified in the **First msg-id** and **Last msg-id** fields.
Leaving the **To msg-id(s)** field empty has the same effect as the entry (same) and causes the message code of the source file to be transferred unchanged to the target file.
(same) must not be specified if the source file is also the target file. (same) **must** be specified if

- the **First msg-id** field contains
  - the keyword (list)
  - the entry *
  - a partially qualified specification containing the wildcard character * (e.g.: AB*)
- a message range across a number of message classes was defined in the **First msgid** and **Last msg-id** fields (e.g.: ABC0000 - ABC9999).

Validity criteria:
Seven-character message code or partially qualified message code in which the asterisk (*) can stand for between one and four characters.

Example

> Valid specifications for <partial msg-id>* are:
> ABC*, ABC0*, ABC00*, ABC000*

The entry * selects all defined message codes. For further information on message codes, see the “Introduction to System Administration” [5 (Related publications)].

Display: (same)

To file (name of the target file)

Name of the message file to which the contents of the source file are to be transferred. The current message file, i. e. the message file that was opened in the MENU mask and whose name was transferred to the **To file** field, can be overwritten in the MOVE mask. Any existing message file can be specified as the target file.
If the message file specified does not yet exist, it is created. The file type (Customer or Standard), product name and version number are defined as for the source file.

Validity criteria:
Data type: <filename 1..54>

Command

For detailed information on

- entering statements, see "General mask format"
- function key assignment, see "General mask format"

Follow-on operations:

MOVE	-> ?

DUE	The MOVE function is initiated; the MOVE mask is displayed again with its original status. The function can be called again by entering new values in the now empty “msg-id” fields.
F3	Quits the MOVE mask without executing the MOVE function. The original MENU mask is displayed.

8.2.6.5 SHOW mask - Display message file contents

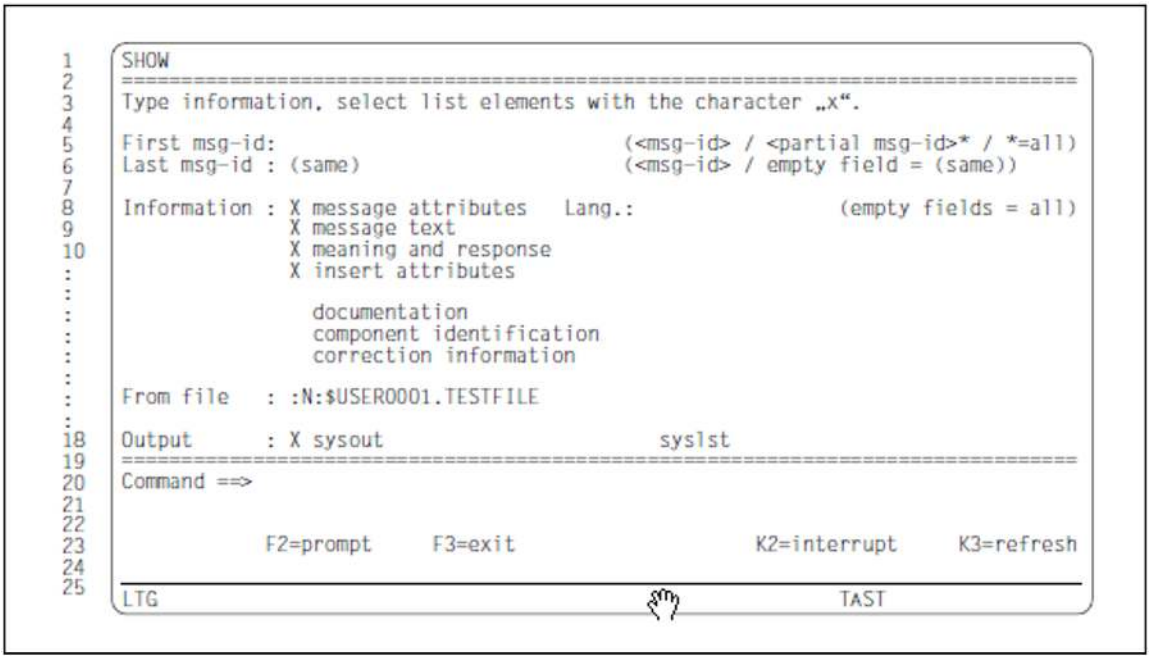
Mask sequence

MENU -> 4 -> SHOW

Function

This mask allows the user to select message file components by specifying the message code and output them to SYSOUT or to a SYSLST file. The message units are displayed classified according to their attributes, their message, meaning and response texts in the specified languages and their insert attributes. Documentation lines may also be displayed. These message file components are sorted according to message class. A separate mask is used for output to SYSOUT (see "SHOW-OUTPUT mask - Output message units and additional information", SHOW-OUTPUT mask).

Mask



Input fields

First msg-id (first message code of the message range)

The specified message code identifies a message unit or the first message unit of a message range.

Validity criteria:
Seven-character message code or partially qualified message code in which the asterisk (*) can stand for between one and seven characters.
The first three characters must be letters (message class).

Example

Valid specifications for <partial msg-id>* are:
A*, AB*, ABC*, ABC0*, ABC00*, ABC000*

The entry * selects all defined message codes. For further information on message codes, see the “Introduction to System Administration” [5 (Related publications)].

Last msg-id (last message code of the message range)

The specified message code identifies the last message unit of a message range. Only one message unit is selected if this message code matches the entry in the **First msg-id** field or the keyword (same) is transferred.

Validity criteria:
Seven-character message code; if the field is empty, the message code is transferred from the **First msg-id** field.
For further information on message codes, see the “Introduction to System Administration” [5 (Related publications)].

Display: (same)

Information (message file components to be output to SYSOUT or SYSLST)

Message units and documentation lines can be selected from the defined message range and output to SYSOUT or SYSLST.

Validity criteria:
The file contents must be marked with the character “X”.

Display:

- X message attributes
- X msg-text
- X meaning and response
- X Insert attributes

message attributes

The message attributes of the message units defined in the specified message range are output.
The message attributes include the access method, the output destination, the routing code, the format of the output text, the weight code and the message warranty.

msg text
The message text is output in the specified languages (see the **Lang** field). If the **Lang** field is empty, the message texts are output in all defined languages.

meaning and response
The meaning and response texts are output in the specified languages. If no language is specified in the **Lang** field, the meaning and response texts are output in all defined languages.

insert attributes
The insert attributes are output.

documentation
All documentation lines defined in the specified message range are output. If other defined message ranges overlap the specified message range, their documentation lines are also output. See the description of the //SHOW statement.

component identification / correction information
This additional information is available only for BS2000 standard message files.

Lang. (1-letter language identifier)

Entering the language identifier selects the message units that contain texts in this language. The texts are output in the same order as the language identifiers were input. If no language is specified, all message units are selected. If this is the case, the texts are output in the alphabetical order of the language identifiers.
If an undefined language is specified for the selected message unit, the message attributes are not displayed, even if selected (X message attributes).

Validity criteria:
A letter from A through Z can be entered. To ensure correct selection, the letter entered here must match the identifier defined when the message unit was created.

From file (message file)

Name of the message file whose components are to be output.
The file name transferred from the MENU mask can be overwritten.

Validity criteria:
<filename 1..54 without-gen-vers>

Output (output destination)

The selected message file contents can be output to SYSOUT and/or SYSLST.

Validity criteria:
Must be marked with the character “X”.

Display: X sysout

sysout
The selected message file components are output to the user’s data display terminal in the SHOW-OUTPUT mask (see "SHOW-OUTPUT mask - Output message units and additional information").

syslst
The selected message file components are output to the SYSLST system file. Lines per page = 60.

Command

For detailed information on

- entering statements, see "General mask format"
- function key assignment, see "General mask format"

Follow-on operations:

SHOW	-> ?

DUE	The SHOW function is initiated; the requested message units are output.
F3	Quits the SHOW mask without executing the SHOW function. The original MENU mask is displayed.

8.2.6.6 SHOW-OUTPUT mask - Output message units and additional information

Function

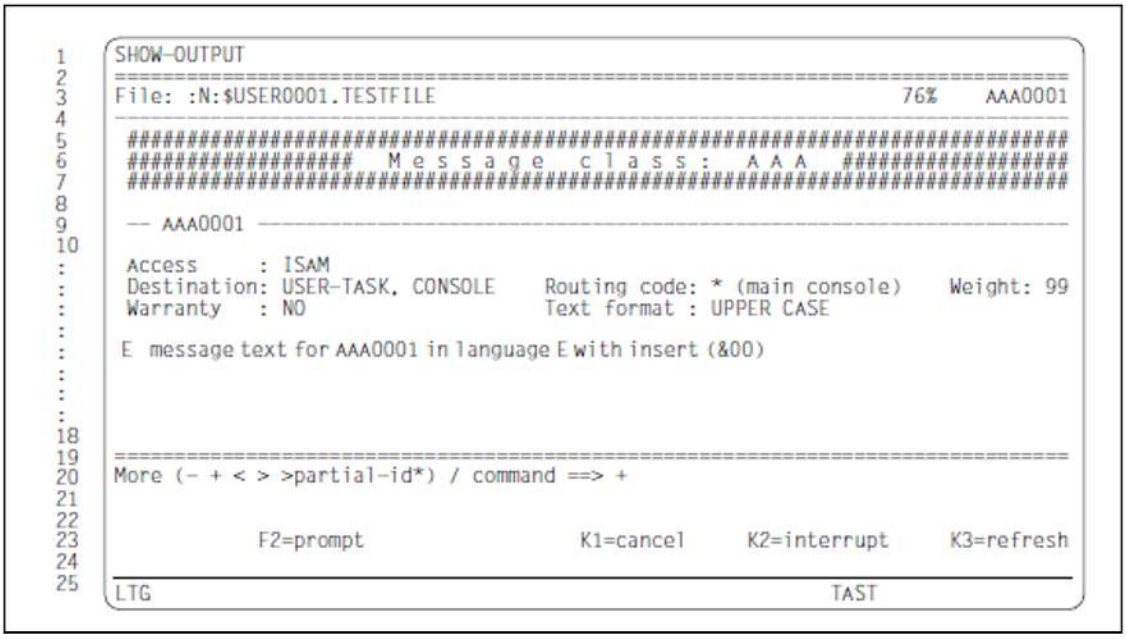
The SHOW-OUTPUT mask is called after the SHOW mask. The information on the selected message units is output together with the documentation lines to SYSOUT (and SYSLST) in the order shown below.
If the information is output to SYSOUT, it is possible to page backwards and forwards through the message file being displayed (see "SHOW-OUTPUT mask - Output message units and additional information").

> If message units from several message classes are displayed, the message classes appear in alphabetical order.

1. Message class
   A new message class is indicated by an appropriate text, e.g. “Message Class: This text appears in a three-line area, enclosed by “#” characters.
2. component identification
   Appears only for standard messages (internal).
3. message units
   The contents of a message unit are displayed in the following order:
   - Message attributes
   - Insert attributes
     Language identifier Message texts (and, if defined, meaning and response texts) are displayed for each language.
   - Language identifier
     Message texts (and, if defined, meaning and response texts) are displayed for each language.
     The “^” separators in the displayed message text have already been converted, i.e. the text that follows “^” starts on a new line.
     The sequence of the languages depends on the order in which the language identifiers were entered in the SHOW mask; if no language has been entered, the language identifiers are displayed in alphabetical order.
4. Documentation lines
5. Correction information
   Only displayed for standard messages (internal).

Mask *(example)*

Three message units (AAA0001 to AAA0003) were defined in message file :N:$USER0001.TESTFILE with message, meaning and response texts.
The three message units are to be output to SYSOUT. The message units are edited for output and the total edited information scaled at 100%. The subsequent screen output is the third of a total of four outputs. The information area of the screen indicates that 76% of the edited information volume has already been output.



Output fields

In the information area

File (name of the message file)

Name of the message file whose contents are displayed.

XX% (information volume in %)

The selected message range of a message file is formatted before being displayed. 0% indicates the start and 100% the end of the edited information.

AAA (message class of the displayed information)

The message class is displayed on output of the first message code within a message class or when documentation lines are output.

AAAXXXX (message code of the first message unit visible in the work area)

A seven-character message code refers to the message unit that is still partially visible in the upper work area.

In the work area

If a new message class is displayed, it is preceded by a three-line comment (see “Function”).

AAAXXXX (message code of the information being displayed)

The letters AAA stand for the message class, XXXX for the message number.

Access	(MIP access method)
Destination	(message output destinations)
Warranty	(message warranty)
Routing Code	(routing code)

Text Format	(format of the text when output via the MSG7X macro or the /HELP-MSG-INFORMATION command)
Weight	(message priority)
Insert attributes	(name and default text of the defined inserts)

In addition to the insert number, the insert name is output in the **Name** field and the default text in the **Default Value** field, enclosed in single quotes.

The single quotes are **not** part of the default text. Entry of two successive single quotes '' in the **Default Value** field corresponds to the specification of DEFAULT-VALUE=*EMPTY-STRING.

The insert name is always output in uppercase letters. This name is a component of the S variable which the user can declare. The default text forms the default contents of this S variable. Further information is provided in the "Introduction to System Administration" [5 (Related publications)].

The field **Automatic Help** indicates whether the automatic help function is available. If a message code or part of a message code is output using an insert (e.g. in the case of DMS errors only the message number is output), the automatic help function causes MIP to output the associated message text in addition to the message code. For further information refer to "ADD-MSG - Add message unit".

In addition to the language identifier, the message text is displayed. The inserts are **not** replaced in the message text.

This is followed by the meaning text (indicated by ?) and response text (indicated by !).

Documentation:

AAAXXXX - AAAXXX (message range)

Owner name	(person responsible for these messages)
Team	(team responsible for these messages)

The comment line is the end of the documentation for every message range.

For standard message files of the manufacturer, the component identification is also output.

Component	(component name)
Domain	(area of application)
Version	(component version)
Owner	(person or team responsible for this component)
Date	(release date of the message file)

In the command area

More (- + <>> partial-id*)

The following options are available for paging through the information displayed:

- or +	Displays the previous or next mask.
-i or +i	Scrolls the screen contents backwards or forwards by i lines.
- or ++	Pages to the start or end of the formatted information.

<	Pages to the start of an item of information (message unit, documentation) if another part of this information item is currently being displayed. If the start of the information item is already displayed, “<” pages to the start of the previous item of information.
>	pages to the start of the next information item.
<<	Pages to the start of the message class currently being displayed. If the start of the message class is already displayed, “<<” pages to the start of the previous message class.
>>	Pages to the start of the next message class or, if there are no more message classes, to the end of the formatted information.
>partial-id*	*If partial-id is more than 3 characters:* Pages to the first message unit whose message code matches the partial message code specified. If no message code matches this partial message code, the first message unit following the partial message code is displayed. This message unit may also belong to a new message class. *If partial-id is less than or equal to 3 characters:* Pages to the first message unit of the specified message class (e.g. >TST*) or to the following class (e. g. TSU). If there are no more message classes, the end of the formatted text is displayed. ℹ The character * is not mandatory.

If a statement is entered in the command area to modify a message unit displayed (e.g. DELETE-MSG), the formatted information on this message unit is **not** updated, i.e. modifications made within the message unit do not appear on the screen. The display is not updated until the next time the SHOW mask is called.

For further information on

- entering statements, see "General mask format"
- function key assignment, see "General mask format"

Follow-on operations:

SHOW-OUTPUT -> ?

+/- DUE	Pages backward and forward in the formatted information (for more details, see "SHOW-OUTPUT mask - Output message units and additional information")
K1	Quits the SHOW-OUTPUT mask; the original SHOW mask is displayed.

8.2.6.7 ADD-MSG mask - Add message unit

Mask sequence

MENU -> 5 -> ADD-MSG

Function

The ADD-MSG mask is the first of a series of four masks that can be called to add a new message unit to the current message file. The message range specified determines the number of new message units and the position at which they are to be inserted in the message file. The inputs in the ADD-MSG mask must be confirmed with DUE; the routine then branches to the other masks.

A new message unit is not saved until all the necessary specifications (message text, access method, etc.) have been made for all the defined languages.

It is not possible to add texts in new languages to an existing message unit. If the user enters a message code that already exists, MSGMAKER assumes that he/she wishes to modify a message unit and automatically displays the MODIFY-MSG mask.

Mask

```
ADD-MSG
================================================================================
Message file: :N:$USER0001.TESTFILE
--------------------------------------------------------------------------------
Type information, select list elements with the character „x“.

First msg-id    :                          (<msg-id> / <partial msg-id>* or #)
Last msg-id     :                          (<msg-id> / empty field = (same))


Access method(s): X ISAM       DLAM       LOCAL DLAM       MINIMIP       BAMR
Destination(s)  : X user-task
                    console  —>  Routing code:          (1 character / *=main)
Weight          :                                  (00-99 / empty field = none)
Warranty        : N                                          (Y=Yes / N=No)

Language(s)     :
Edit            : X msg text      meaning + response      insert attributes

================================================================================
Command ==>


           F2=prompt     F3=exit                  K2=interrupt     K3=refresh

LTG                                                                  TAST
```

Output fields

File (name of the message file)

The file name entered in the main mask MENU is transferred to the ADD-MSG mask. This name must not be modified.

Input fields

First msg-id (first message code of the message range)

If a message range (or a wildcard) is specified in the **First msg-id** field, MSGMAKER finds all the message codes in this range that are as yet undefined and displays them in turn. It is useful to define a message range if the same message attributes are to apply to all message units.
The attributes are entered in the ADD-MSG mask at the start of the procedure and remain valid for all further message units of the defined range until the user modifies them. Each time the ADD-MSG, MSG-TEXT, MEANING /RESPONSE, INSERT-ATTRIBUTES masks have been filled in, MSGMAKER always returns to the ADD-MSG mask.

MSGMAKER continues with the message code in the defined range that follows the message code just entered by the user. The message attributes either apply to all further message units on the basis of the entry in the first ADD-MSG mask or may also be modified as required.
Each message unit is saved to the message file immediately; MSGMAKER does not wait until the entire message range has been processed.
If no specification is necessary for a message code offered by MSGMAKER, the user can skip that code by pressing K1.

Validity criteria:
Seven-character message code or partially qualified message code in which the asterisk * can stand for between one and seven characters and the # character can stand for between one and four digits. The first three characters must be letters (message class).

Example

> Valid specifications for <partial msg-id>* or # are: A*, AB*, ABC*, ABC0*, ABC00*, ABC000*
> ABC#, ABC0#, ABC00#, ABC000#

The entry * selects all defined message codes. For further information on message codes, see the “Introduction to System Administration” [5 (Related publications)].

Last msg-id (last message code of the message range)

The specified message code identifies the last message unit of a message range. Only one message unit is selected if this message code matches the entry in the **First msg-id** field or the keyword (same) is transferred.

Validity criteria:
Seven-character message code; if the field is empty, the message code is transferred from the **First msg-id** field.
For further information on message codes, see the “Introduction to System Administration” [5 (Related publications)].

Access method(s) (MIP message access methods)

Indicates the methods used by the system component MIP to access the messages. It is possible to select more than one access method; however, the ISAM and DLAM methods and the ISAM and LOCAL-DLAM methods must not be combined.

Validity criteria:
The access methods must be marked with the character “X”.

Default setting/display: X ISAM

> **ISAM**
> MIP searches for messages via the ISAM key.

DLAM
This access method is used for particularly frequent messages. If a message file that contains a DLAM message is activated, the DLAM message is loaded into main memory. MIP can output the DLAM message directly without accessing the message file.

LOCAL-DLAM / MINIMIP / BAMR
These access methods are reserved for internal use with the manufacturer.

Destination(s) (message output destinations)

This specification allows the user to document the output destinations. The MSG7X macro and the /HELP-MSG-INFORMATION command do **not** evaluate this specification.

Validity criteria:
The output destination must be marked with the character “X”.

Default setting/display: X user-task

user-task
The message output destination is SYSOUT, SYSLST or a user-specific memory area.

console
The message output destination is a console. The routing code is evaluated as a destination specification if subsidiary consoles are used in addition to the main console as message destinations.

Routing code

A routing code must be specified if in the **Destination(s)** field, the output destination **console** was selected. Further information is provided in the "Introduction to System Administration” [5 (Related publications)].

Validity criteria:
Data type: <alphanum-name 1..1>

Weight (message priority)

A message weight must be specified if in the **Destination(s)** field, the output destination **console** was selected. Further information is provided in the "Introduction to System Administration” [5 (Related publications)].

Validity criteria:
Data type: <integer 0..99>

Warranty (message warranty)

The message attribute “Warranty” is evaluated by MIP. The warranty declaration indicates that specific parts of the message will not be modified in future BS2000 versions.

The following message components are guaranteed:

- message code
- numbering and meaning of inserts

The message text is **not** guaranteed.

MIP creates S variables for warranty messages. Further information is provided in the "Introduction to System Administration” [5 (Related publications)].

Language(s) (1-letter language identifier)

A 1-letter identifier is used as an abbreviation for each language. D should be used for German and E for English.

Validity criteria:
The letters “A” through “Z”.

The order in which entries are made here influences the order in which they are displayed in subsequent masks.

Edit (text selection)

At least one message text **must** be defined for each message unit. Meaning and response texts can be entered as required.

Validity criteria:
The texts must be marked with the character “X”.

Default setting/display: X msg text

msg text
The ADD-MSG mask is followed by the MSG-TEXT mask, which allows the user to enter the message texts in the different languages. For a description of the MSG-TEXT mask, see "MSG-TEXT mask - Add or modify message text".

meaning + response
The MSG-TEXT mask is followed by the MEANING/RESPONSE mask, which allows the user to enter the meaning and response texts. For a description of the MEANING/RESPONSE mask, see "MEANING /RESPONSE mask - Add or modify meaning and response text".

Insert attributes
The MSG-TEXT and MEANING/RESPONSE masks are followed by the INSERT-ATTRIBUTES mask, in which names and default texts are assigned to the inserts that were defined in the message text. For a description of the INSERT-ATTRIBUTES mask, see "INSERT-ATTRIBUTES mask - Add or modify insert attributes".

Command

For detailed information on

- entering statements, see "General mask format"
- function key assignment, see "General mask format"

Follow-on operations:

ADD-MSG -> ?

DUE	The ADD-MSG function is initiated; MSGMAKER branches to the MSG-TEXT mask.
K1	If a message range is entered in the ADD-MSG mask, MSGMAKER offers the skip function K1 as of the second message code of the range. The user can then skip this message code. Following this, an ADD-MSG mask appears with the next message code.
F3	The ADD-MSG function is aborted; MSGMAKER returns to the original MENU mask.

8.2.6.8 MODIFY-MSG mask - Modify message unit

Mask sequence

MENU -> 6 -> MODIFY-MSG

Function

The MODIFY-MSG mask is the first of a series of four masks that can be called to modify a message unit in the current message file.
Once the message file to be processed has been entered in the main mask MENU and function 6 (Modify) has been selected, MSGMAKER displays the MODIFY-MSG mask.
The first message unit is displayed once a message range has been entered in the **First msg-id** (and, if applicable, the **Last msg-id**) field and confirmed by pressing the DUE key.Only then can the message attributes of the message unit be modified in the MODIFY-MSG mask. If the texts are also to be modified (by inserting, appending, deleting or replacing lines of text), highlighting the following fields (and then pressing DUE) calls the relevant masks.

X msg-text	MSG-TEXT mask
X meaning + response	MEANING / RESPONSE mask
X Insert attributes	INSERT-ATTRIBUTES mask

If the modifications only affect the message attributes, they must be confirmed in the MODIFY-MSG mask with DUE and saved to the message file. More complex modifications within a message unit (message text, inserts, ...) are only saved to the message file once the user has run through all the necessary masks.
If a message unit is not to be modified within the selected message range, the user can skip that message code with K1. The function key is offered in the mask as soon as the user has read the message units into the message file with DUE.

Mask

```
MODIFY-MSG
================================================================================
Message file: :N:$USER0001.TESTFILE
--------------------------------------------------------------------------------
Type information, select list elements with the character „x“.

First msg-id    :                          (<msg-id> / <partial msg-id>* or #)
Last msg-id     : (same)                   (<msg-id> / empty field = (same))


Access method(s):   ISAM       DLAM       LOCAL DLAM       MINIMIP       BAMR
Destination(s)  :   user-task
                    console  -->  Routing code:          (1 character / *=main)
Weight          :                                 (00-99 / empty field = none)
Warranty        :                                              (Y=Yes / N=No)

Language(s)     :
Edit            :   msg text      meaning + response      insert attributes

================================================================================
Command ==>


            F2=prompt     F3=exit                   K2=interrupt     K3=refresh

--------------------------------------------------------------------------------
LTG                                                              TAST
```

Output fields

File (name of the message file)

The name entered in the main mask MENU is transferred to the MODIFY-MSG mask. This name must not be modified.

The selected message range is displayed at the right-hand edge of the information area. (*Note: this is not shown in the mask*)

Input fields

First msg-id (first message code of the message range)

If a message range is entered in the **First msg-id** field and confirmed by pressing the DUE key, all defined message codes of this message range are displayed in turn. The cursor is positioned on the **Access method(s)** field to enable the message unit to be processed.

Validity criteria:
Seven-character message code or partially qualified message code in which the asterisk * can stand for between one and seven characters and the # character can stand for between one and four digits. The first three characters must be letters (message class).

Example

Valid specifications for <partial msg-id>* or # are:

```
A*, AB*, ABC*, ABC0*, ABC00*, ABC000*

ABC#, ABC0#, ABC00#, ABC000#
```

The entry * selects all defined message codes. For further information on message codes, see the “Introduction to System Administration” [5 (Related publications)].

Last msg-id (last message code of the message range)

The specified message code identifies the last message unit of a message range. Only one message unit is selected if the message code matches the specification in the **First msgid** field or the keyword (same) is transferred.

Validity criteria:
Seven-character message code; if the field is empty, the message code is transferred from the **First msg-id** field. For further information on message codes, see the “Introduction to System Administration” [5 (Related publications)].

Display: (same)

Access method(s) (MIP message access methods)

Indicates the methods used by the system component MIP to access the messages. It is possible to select more than one access method; however, the ISAM and DLAM methods and the ISAM and LOCAL-DLAM methods must not be combined.

Validity criteria:
The access methods must be marked with the character “X”.

An access method can be deselected by overwriting the “X” with a blank or a null character. A new access method must be selected again using “X”. If no further changes to the message attributes or texts are required, the user can confirm the changes made by pressing the DUE key. The next message unit within the specified message range is displayed.

ISAM
MIP searches for messages via the ISAM key.

DLAM
This access method is used for particularly frequent messages. If a message file containing a DLAM message is activated, the DLAM message is loaded into main memory. MIP can output the DLAM message directly without accessing the message file.

LOCAL-DLAM / **MINIMIP** / **BAMR**
These access methods are reserved for internal use with the manufacturer.

Destination(s) (message output destinations)

This specification allows the user to document the output destinations. The MSG7X macro and the /HELP-MSG-INFORMATION command do **not** evaluate this specification.

Validity criteria:
The output destination must be marked with the character “X”.

Entries can be deleted by overwriting the fields with blanks or null characters, and selected again using “X”.

user-task
The message output destination is SYSOUT, SYSLST or a user-specific memory area.

console
The message output destination is a console. The routing code is evaluated as a destination specification.

Routing code

A routing code must be specified if in the **Destination(s)** field, the output destination **console** was selected. Further information is provided in the "Introduction to System Administration” [5 (Related publications)].

Validity criteria:
Data type: <alphanum-name 1..1>

Weight (message priority)

A message weight must be specified if in the **Destination(s)** field, the output destination **console** was selected. Further information is provided in the "Introduction to System Administration” [5 (Related publications)].

Validity criteria:
Data type: <integer 0..99>

Warranty (message warranty)

The message attribute “Warranty” is evaluated by MIP.
The warranty declaration indicates that specific parts of the message will not be modified in future BS2000 versions.

The following message components are guaranteed:

- message code
- numbering and meaning of inserts

The message text is **not** guaranteed.

MIP creates S variables for warranty messages. Further information is provided in the "Introduction to System Administration” [5 (Related publications)].

Language(s) (1-letter language identifier)

A 1-letter identifier is used as an abbreviation for each language. The letter D stands for German and E for English. If a language identifier is displayed in the MODIFY-MSG mask, the user can neither modify it by overwriting it with another letter nor delete it by overwriting it with a blank.

The language identifier can be modified only in the MSG-TEXT mask (see the description on "MSG-TEXT mask - Add or modify message text"). The MSG-TEXT and DELETE-MSG masks allow the user to delete the language identifier and thus also the message text.

Validity criteria:
The letters “A” through “Z”.

Edit (text selection)

Selects the texts to be modified.

Validity criteria:
The texts must be marked with the character “X”.

msg text
The MODIFY-MSG mask is followed by the MSG-TEXT mask, which allows the user to modify the message texts in the different languages. For a description of the MSG-TEXT mask, see "MSG-TEXT mask - Add or modify message text".

meaning + response
The MSG-TEXT mask is followed by the MEANING/RESPONSE mask, which allows the user to modify the meaning and response texts. For a description of the MEANING/RESPONSE mask, see "MEANING /RESPONSE mask - Add or modify meaning and response text".

insert attributes
The MSG-TEXT and MEANING/RESPONSE masks are followed by the INSERT-ATTRIBUTES mask, which allows the user to modify the insert attributes. For a description of the INSERT-ATTRIBUTES mask, see "INSERT-ATTRIBUTES mask - Add or modify insert attributes".

Command

For detailed information on

- entering statements, see "General mask format"
- function key assignment, see "General mask format"

Follow-on operations:

MODIFY-MSG -> ?

DUE The MODIFY-MSG function is initiated; modifications to message attributes are saved to the message file; for text modifications MSGMAKER branches to the selected masks.

K1 If a message range is entered in the MODIFY-MSG mask and confirmed with DUE, MSGMAKER offers the skip option K1. The user can then skip this message code. After that, the MODIFY-MSG mask appears with the next message code.

F3 The MODIFY-MSG function is aborted; modifications to previously edited messages are stored in the message file and modifications to the current message code are lost. MSGMAKER returns to the original MENU mask.

8.2.6.9 MSG-TEXT mask - Add or modify message text

Mask sequence

MENU -> 5/6 -> ADD-MSG/MODIFY-MSG -> MSG-TEXT

Function

The user calls the MSG-TEXT mask by selecting the field “**Edit**: **X** msg text” in the ADD-MSG or MODIFY-MSG mask. The MSG-TEXT mask allows the user to enter, modify or delete message texts in defined languages.

The message code of the current message unit is displayed in the **Msg-id** field, but cannot be modified. If several languages have been defined, the corresponding message texts can be entered one after the other in the mask. The MEANING/RESPONSE and INSERT-ATTRIBUTES masks are then called if they were also selected in the ADD-MSG or MODIFY-MSG mask. It is also possible to enter the message text in only one language in the MSG-TEXT mask and then branch immediately to the other two masks. Once these masks have been processed, MSGMAKER automatically redisplays the MSG-TEXT mask, in which the message texts can be entered in the remaining languages.
The message units that have been added or modified are not saved to the message file until the user has made all the necessary entries in the MSG-TEXT, MEANING/RESPONSE or INSERT-ATTRIBUTES mask and pressed DUE after completing the last of these masks.

Mask

```
1   MSG-TEXT
2   ================================================================================
3   Msg-id  : AAA0001       Text output format : U     (U=Upper case / L=Lower case)
4
5   Language:     Text:
6
7
8
9   Language:     Text:
10
:
:   Language:     Text:
:
:
:   Language:     Text:
:
18
19  ================================================================================
20  More (  +) / Command ==>
21
22
23             F2=prompt     F3=exit      K1=cancel     K2=interrupt     K3=refresh
24
25  LTG                                                              TAST
```

Output fields

Msg-id (message code of the message unit to be processed)

The message code entered in the ADD-MSG or MODIFY-MSG mask is transferred to the MSG-TEXT mask and **cannot** be modified.

Input fields

Text output format (output format of the message text)

Specifies the format of the message text when output via the MSG7X macro or the /HELP-MSG-INFORMATION command. The default texts for inserts and the texts that are specified via the MSG7X macro are likewise adapted to this format.

Validity criteria:
The letter “U” or “L”.
“U” stands for uppercase (the message text is output in uppercase letters) and “L” stands for lowercase (the message text is output as it is entered in the MSG-TEXT mask).

> The message text format selected in the MSG-TEXT mask is also the format in which the message text is stored in the message file.

Default setting/display: U

Language (1-letter language identifier)

Previous mask was ADD-MSG:
If language identifiers were entered in the **Language(s)** field in the ADD-MSG mask, they are transferred to the MSG-TEXT mask. If not, the language identifiers can be entered directly in the MSG-TEXT mask.

Previous mask was MODIFY-MSG:
All languages displayed in the MODIFY-MSG mask are transferred to the MSG-TEXT mask. The following options are now available:

- The language identifier remains unchanged.
  The user may modify the message text.
- The user may redefine the language identifier by overwriting it.
  The corresponding message text is then stored under a different language.
- The user may delete the language identifier by overwriting it with a blank.The message text must also be deleted. If only one language was defined in the message unit, MSGMAKER prompts the user to enter another language (and message text).

Text (message text)

The message text is stored internally as a single line (not three lines, as in the screen display). The separator “^” can be used to split the text over several lines when it is output via the MSG7X macro or the /HELP-MSG-INFORMATION command. Null characters within the text are replaced with blanks. For further information on the structure of the message text, see the “Introduction to System Administration” [5 (Related publications)].

Validity criteria:
Up to 220 characters of pure message text, including the unreplaced strings ((&00) ... (&29).

More (+ -) Displays another mask.

+ If more than four languages are stored in the message unit or if a fifth language is to be added, the user can enter “+” to call a second MSG-TEXT mask for displaying or entering further texts.

- The user enters “-” to return to the first MSG-TEXT mask.

Validity criteria:
The character “-” or “+”.

Command

For further information on

- entering statements, see "General mask format"
- function key assignment, see "General mask format"

Follow-on operations:

MSG-TEXT	-> ?

DUE The inputs in the MSG-TEXT mask are confirmed and depending on what was previously selected, the routine branches to the MEANING/RESPONSE, INSERT-ATTRIBUTES masks or returns to the original ADD-MSG or MODIFY-MSG mask.

K1 Returns to the original ADD-MSG or MODIFY-MSG mask. Modifications which were confirmed in the MSG-TEXT mask with DUE are stored and can be used again. The previously modified data is displayed when the user returns to the MSG-TEXT mask with DUE.
In this mask K1 has the effect of K3 = refresh.

F3 Processing of the current message texts is aborted. Modifications to previously edited messages are stored in the message file and modifications to the current message code are lost. MSGMAKER returns to the original ADD-MSG or MODIFY-MSG mask.

8.2.6.10 MEANING/RESPONSE mask - Add or modify meaning and response text

Mask sequence

MENU -> 5/6 -> ADD-MSG/MODIFY-MSG -> MEANING/RESPONSE

Function

When the user selects the field “**Edit**:**X** meaning + response” in the ADD-MSG or MODIFY-MSG mask, MSGMAKER branches to the EDT subroutine and calls a modified EDT work screen as the MEANING/RESPONSE mask. For detailed information on EDT refer to the manual “EDT“ [14 (Related publications)].
The user can add, modify or delete meaning and response texts using one or more masks.All EDT statement codes can be used to create meaning and response texts. These statement codes must be entered in the mark column (column 1) of the screen. EDT statements can also be entered in the last line (statement line) of the screen.
The screen is divided into two areas. The meaning text is entered in the upper area and the response text in the lower area. If the meaning text contains more than nine lines, new lines must be added to the upper area, since any text entered in the lower area is interpreted automatically as response text.
The information lines that appear on the screen by default cannot be overwritten. These lines are not part of the meaning or response text.

The meaning and response texts for all other defined languages can be entered in the EDT screens numbered 0 through 7 and confirmed with DUE, F1 or F2. All EDT screens that are not used for text input are available as work screens. If texts are defined for eight languages, EDT screens 8 and 9 may be used as work screens.
The @RET[URN] or RETURN statement entered in the statement line terminates EDT. All processed languages (= EDT screens) are transferred together to the MSGMAKER routine. The meaning and response texts that have been entered are not saved until the user has completed all the masks required for processing the message unit.

Mask

```
1  Language: D                                        Language E : EDT screen 1, 2
2  Msg text:
3  < M E A N I N G    (maximum 256 lines)    Msg-id: AAA0001    Language: D >!!!!!!
4
5
6
7
8
9
10
:
:
:  < R E S P O N S E  (maximum 256 lines)    Msg-id: AAA0001    Language: D >!!!!!!
:
:
:
:
18
19
20
21
22       To set overwrite mode: <F2>. To quit EDT: <K1> (=cancel) or RETURN
23                                                                 0001.00:001(0)
24
25 ____________________________________________________________________________
   LTG                                                                    TAST
```

Output fields

Language (1-letter identifier for the current language)

Language (identifiers for all other defined languages)

EDT screen (EDT screens in which the user can enter texts for all other language identifiers)

Msg text (current message text for the language identifier entered under **Language**)

Input fields

> Before text can be entered or modified, the displayed lines must be made overwritable using the function key F2. It is possible to select individual lines for processing by entering an “x” in the mark column.

The separator “^”

If this character is used within the meaning or response text, any text that follows it is written on a new line after the DUE, F1 and F2 keys have been pressed. This separator is valid only in EDT and is **not** part of the meaning and response text. It therefore differs from the separator “^” that can be used in the message text.

M E A N I N G (maximum 256 lines) (meaning text)

Validity criteria:
The meaning text may consist of up to 256 lines of 74 characters each. Characters entered in columns 75 through 80 are truncated automatically after the DUE key is pressed and the message EDT2267 is output. Lines consisting of null characters or blanks are suppressed. Null characters within the text are replaced with blanks. If the meaning text is to be longer than nine lines, new lines must be added to the upper screen area.

Display: **Msg-id** (current message code)

Language (1-letter identifier for the current language)

The message code of the current message unit and the current language identifier (as displayed in the **Language** field) are displayed.

R E S P O N S E (maximum 256 lines) (response text)

Validity criteria:
The response text may consist of up to 256 lines of 74 characters each. Characters entered in columns 75 through 80 are truncated automatically after the DUE key is pressed and the message EDT2267 is output. Lines consisting of null characters or blanks are suppressed. Null characters within the text are replaced with blanks.

Display: **Msg-id** (current message code)

Language (1-letter identifier for the current language)

The message code of the current message unit and the current language identifier (as displayed in the **Language** field) are displayed.

Command area

Function keys

The keys F1, F2 and F3 do not have the same function in EDT as they have in the MSGMAKER routine.

F1 corresponds to the DUE key.

F2 If the user confirms the screen contents by pressing F2, all the lines become overwritable, i.e. text can be entered or modified in the upper and lower screen areas. Information lines are protected against overwriting.

F3 If a search operation is started with the “@ON .... FIND” statement, the F3 key can be used to jump to the next or previous search string.

For further information on the K2 and K3 keys, see "General mask format".

For further information on the editing functions, see the “EDT” manual [14 (Related publications)].

Follow-on operations:

MEANING/RESPONSE	-> ?

@RET[URN]	The inputs are confirmed and, depending on what was previously selected, the routine branches to the MSG-TEXT, INSERT-ATTRIBUTES masks or returns to the original ADD-MSG or MODIFY-MSG mask.
K1	Aborts screen input in EDT. No modifications (new or modified text) are passed on to the MSGMAKER routine unless the inputs were first confirmed by pressing DUE, F1, F2 or F3. After pressing K1, the last mask displayed in the MSGMAKER routine is called again.

For all other operations, see “Function keys” above.

8.2.6.11 INSERT-ATTRIBUTES mask - Add or modify insert attributes

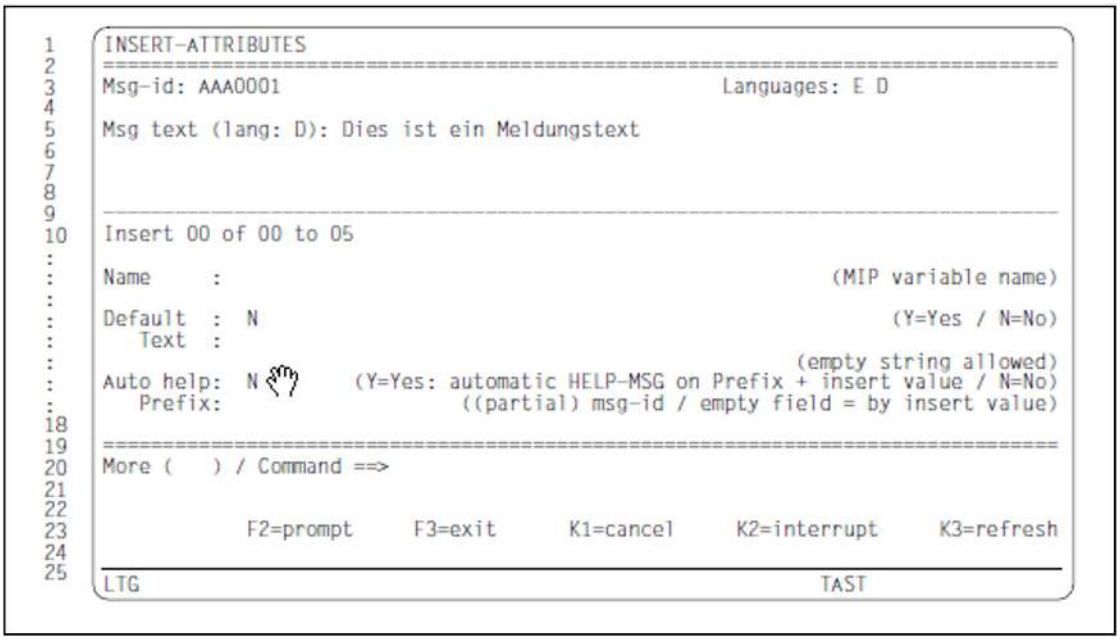
Mask sequence

MENU -> 5/6 -> ADD-MSG/MODIFY-MSG -> INSERT-ATTRIBUTES

Function

The user calls the INSERT-ATTRIBUTES mask by selecting the field „**Edit**: **X** insert attributes” in the ADD-MSG or MODIFY-MSG mask. The INSERT-ATTRIBUTES mask allows the user to add, modify or delete names and default texts for the inserts defined in the message texts. The modifications are confirmed by hitting DUE. The inserts defined in the message unit are common to all the defined languages.

Mask



Output fields

Msg-id (message code of the current message unit)

Language(s) (identifiers for all languages defined in the message unit)

Msg text (In the “language sequence” the first message text in which inserts are defined)

The message text in which one or more inserts have been defined is output as a reference text. Unlike the language identifier, the message text cannot be modified. The user can define a new language for the message text displayed by overwriting the letter in the **Lang** field.

Input fields

Insert .. of .. to .. (Insert number)

Insert numbers can be modified, deleted or added only in the message text itself (see the MODIFY-MSG mask on "MODIFY-MSG mask - Modify message unit").

The scope xx to yy indicates the lowest and highest insert numbers defined in the message text. Entering a new insert number causes a further insert to be output. If individual insert numbers from this range are not defined, they are not output.

Name (Insert name)

Name of the inserts defined in the message texts. Insert names are evaluated by MIP and used to generate S variables. Further information is provided in the "Introduction to System Administration" [5 (Related publications)].

Default text (Default Text)

This default text is inserted in the message text in place of the insert, provided that no current text has been defined in the MSG7X macro.

Default text: Y	There is a default text for an insert text
Default text: N	There is no default text for an insert text

If the ADD-MSG function is being used, it is possible to specify whether a default text is to be displayed or whether an empty string is output. For information on entering an empty string, refer to the operand description for DEFAULT-VALUE = *EMPTY-STRING on "ADD-MSG - Add message unit".
If the MODIFY function is being used, default texts that have already been defined are output.

Validity criteria:
data type: <c-string 1..54 with-low>

The default text in the INSERT-ATTRIBUTES mask is not enclosed in single quotes. Single quotes within the default text do not have to be specified twice. The text entered in the input field is the input text. The input text may also be a null string or blank characters. So the default text can be deleted by overwriting with null characters not with blanks.

Output format of the default text depends on the MSG-TEXT-OUTPUT operand.

Automatic help (Automatic help function)

If a message code or part of a message code is output using an insert (e.g. in the case of DMS errors only the message number is output), the automatic help function causes MIP to output the associated message text in addition to this message code; see also "ADD-MSG - Add message unit".

Auto help: Y	The message code and the message text are output
Prefix:	Message class, message number, etc., refer to PREFIX operand on "ADD-MSG - Add message unit"
Auto help: N	Only the message code is output. The Prefix field may contain only null characters or blanks

Entering “+” or “-” scrolls (pages) to the next insert number or previous insert number defined for this message.

Entry of a new insert number takes precedence over the “paging” function.

Entry of new insert attributes or modification of insert attributes takes precedence over the “paging” function or modification of the insert number.

Command

For further information on

- entering statements, see "General mask format"
- function key assignment, see "General mask format"

Follow-on operations:

INSERT-ATTRIBUTES -> ?

DUE The inputs are confirmed and, depending on what was previously selected, the routine branches to the MSG-TEXT, MEANING/RESPONSE masks or returns to the original ADD-MSG or MODIFY-MSG mask.

F3 The function is aborted. Modifications to previously edited messages are stored in the message file. Modifications to the current message (message/meaning/response text) are lost. The routine returns to the original ADD-MSG or MODIFY-MSG mask, even if there are message units still waiting to be processed.

K1 Returns to the MSG-TEXT or MEANING/RESPONSE mask. Entries in the INSERT-ATTRIBUTES are lost. On returning to the INSERT-ATTRIBUTES mask, new values must be entered.

8.2.6.12 DELETE-MSG mask - Delete message unit

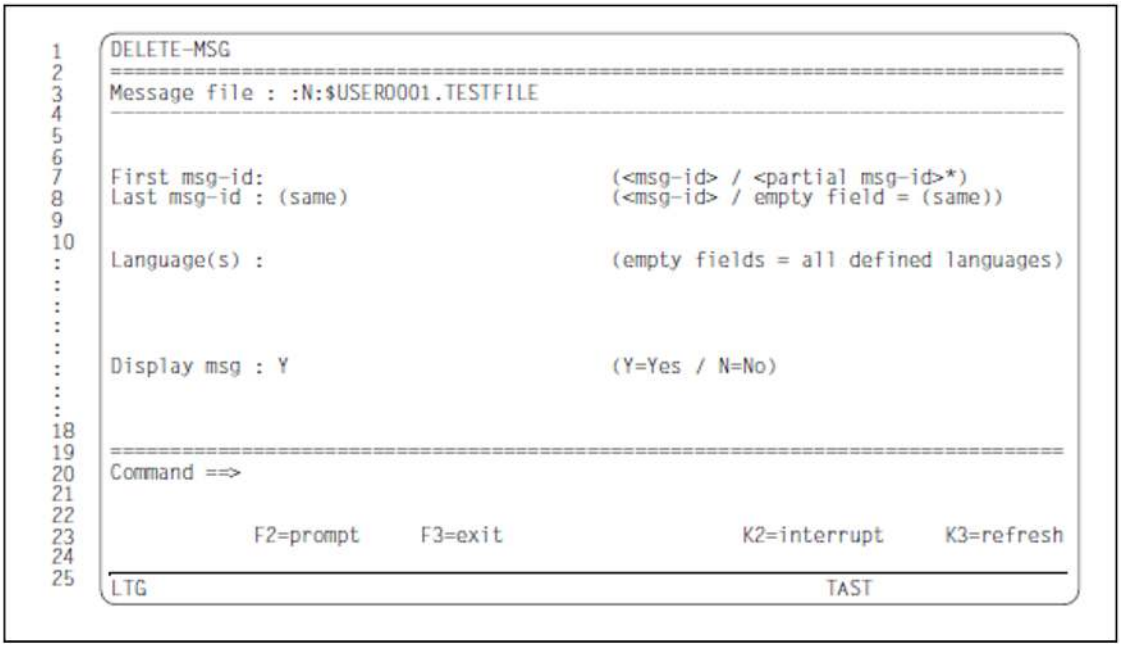
Mask sequence

MENU -> 7 -> DELETE-MSG

Function

This mask allows the user to select message units to be deleted by specifying the message code and the language identifier.
A second mask with the same name is called if the message attributes and message texts of the selected message units are to be displayed once again before the message units are deleted.

Mask 1



Output fields

Message file (current message file)

The current message file is displayed. It was opened in the MENU mask. The file name cannot be modified in the DELETE-MSG mask.

Input fields

First msg-id (first message code of the message range)

The specified message code identifies a message unit or the first message unit of a message range.

Validity criteria:
Seven-character message code or partially qualified message code in which the asterisk (*) can stand for between one and four characters.
The first three characters must be letters (message class).

Example

> Valid specifications for <partial msg-id>* are:
> ABC*, ABC0*, ABC00*, ABC000*

Last msg-id (last message code of the message range)

The specified message code identifies the last message unit of a message range. This message code must belong to the same message class as the first message code. Only one message unit is selected if this message code matches the entry in the **First msg-id** field or the keyword (same) is transferred.

Validity criteria:
Seven-character message code; if the field is empty, the message code is transferred from the **First msg-id** field.
For further information on message codes, see "Description of fields that occur frequently".

Display: (same)

Language(s) (1-letter language identifier)

The identifiers to be entered here correspond to the abbreviations defined when the message unit was created.
If no language is entered in the **Language(s)** field, the texts in all languages and the attributes of the message unit are deleted.
If all the message units of a message file have been deleted in this way, the message file itself is not automatically deleted, since its file attributes and any additional information (e.g. documentation lines) remain intact.

Validity criteria:
The letters “A” through “Z”.

Display msg (displays the message unit)

Entering “Y” (yes) in the **Display msg** field calls the second DELETE-MSG mask, in which the message attributes and message texts of the message unit to be deleted are displayed.If “N” (no) is entered, all the specified message units or message texts of a message unit are deleted immediately without being displayed again.

Validity criteria:
The letters “N” and “Y”.

Default setting/display: Y

Command

For detailed information on

- entering statements, see "General mask format"
- function key assignment, see "General mask format"

Follow-on operations:

DELETE-MSG 1 -> ?

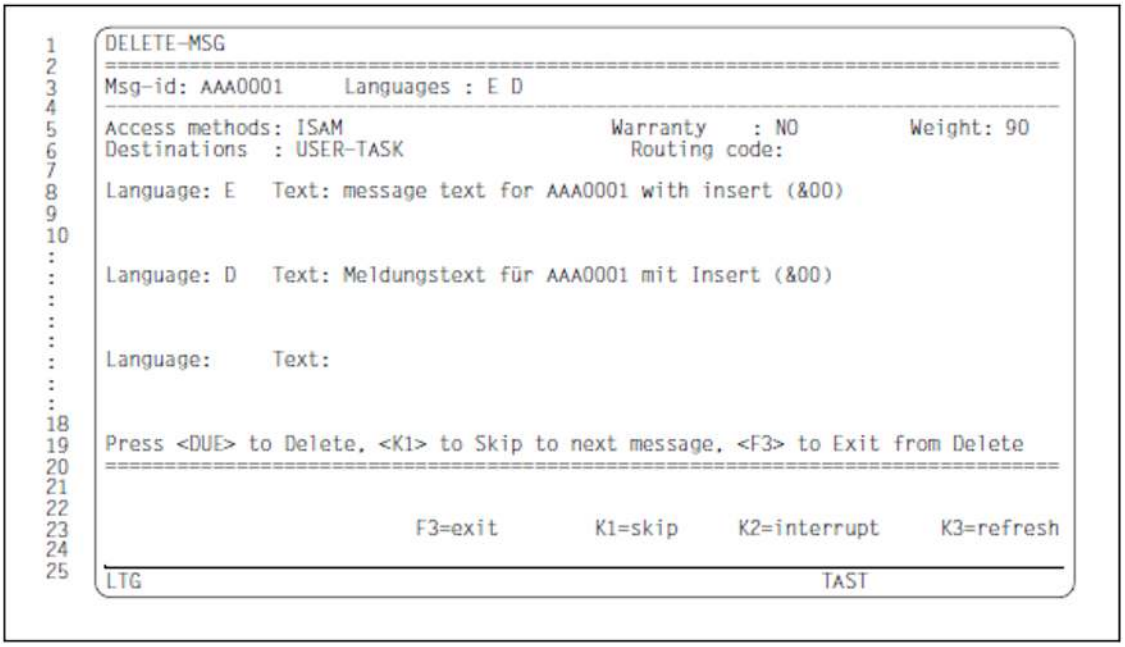
DUE The inputs are confirmed and

- for **Display msg**: N = message unit is deleted
- for **Display msg**: Y routine branches to second DELETE-MSG mask.

F3 The DELETE-MSG function is aborted.

Mask 2

This mask is called if “**Meldung anzeigen**: Y” was specified in the first DELETE-MSG mask.The message code and the letters identifying the selected languages are displayed in the information area of the mask.
The work area of the mask contains the message attributes and no more than the first three message texts in the selected languages.



Output fields

In the information area

Msg-id (message code of the current message unit)

Language (identifiers for all languages selected in the previous mask)

In the work area

Language (1-letter language identifier)

Text (message text in the relevant language)

Command

For further information on assignment of the function keys K2 and K3, see "General mask format".

Follow-on operations:

DELETE-MSG 2 -> ?

The last line of the work area offers the user the following three options for continuing with mask processing:

The following actions can be performed:

DUE The message unit currently displayed (message texts in all selected languages) is deleted immediately. If a message range has been defined, the next message unit is then displayed.

K1 The message unit currently displayed is not deleted. If a message range has been defined, the next message unit is displayed.

F3 The DELETE function is terminated. The message unit currently displayed and all the other message units specified in the message range are not deleted. Control returns to the DELETE-MSG 1 mask.

8.2.6.13 ADD-DOCUMENTATION mask - Add documentation lines

Mask sequence

MENU -> 9 -> ADD-DOCUMENTATION

Function

This mask allows message units to be documented. The documentation is assigned to the message units via message codes (e.g. AAA0001) or via a message range (e.g. AAA0002-AAA0015).

The following specifications are possible:

- Message code/message range
- Name of person responsible for these messages (20 characters)
- Name of the team responsible for these messages (15 characters)
- Comment (60 characters)

The message codes can be entered in the mask in any order. If they are confirmed with DUE, the codes are sorted alphabetically and the documentation lines are stored in the message file.

In this mask, by contrast with MODIFY-DOCUMENTATION, the documentation lines must be appended to already existing ones. The latter being overwrite-protected.

> The specified message codes and the contents of the documentation lines are not checked for errors.

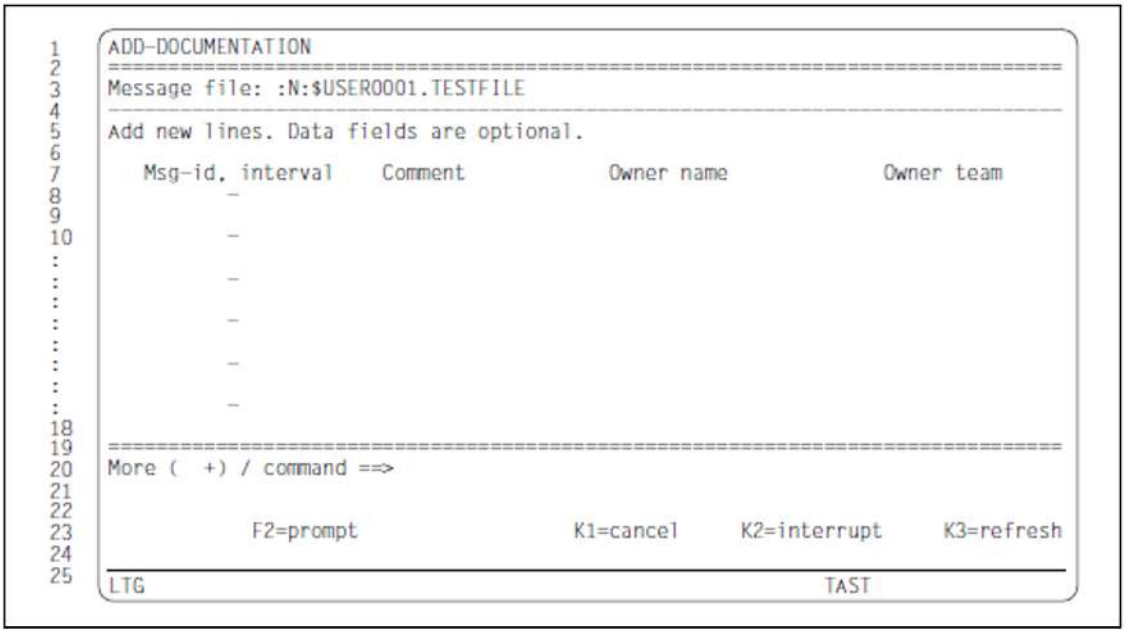
If an already documented message range is input, MSGMAKER outputs the following message:

```
MSMQL00 DOCUMENTATION LINE '<message range>' ALREADY EXISTS
(COMMENT='<comment>'). OVERWRITE? REPLY (Y=YES; N=NO)? N
```

If the user responds with N, the documentation line is displayed again and can be modified.

Using the COPY or MOVE function, documentation lines can be copied or moved within a message file or between two files.
The SHOW function can be used to output to SYSOUT or SYSLST.

Mask



Output fields

Message file

The current message file is displayed. It was opened in the MENU mask. The file name cannot be modified in the ADD-DOCUMENTATION mask.

If documentation lines have already been written for this message file, the last message range, alphabetically speaking, is displayed in the first line of the documentation area. The cursor is located at the beginning of the next line.
The “-” character (in the command line) can be used to page back to message ranges that have already been documented.

Input fields

Msg-id, interval (message code/message range)

This field is used to input a message code or message range which is documented in the subsequent fields. When the input is confirmed with DUE, a single message code is repeated as the upper range limit (e.g. AAA0001-AAA0001).

Validity criteria:
Seven-character message code, of which the first three characters must be letters (message class). The limit values for a message range must belong to the same message class.

The tab key can be used to jump to the next field.

Comment (comment on the messages)

Validity criteria:
Maximum 60 characters, including spaces.

The tab key can be used to jump to the next field.

Owner (person responsible for these messages)

Validity criteria:
Maximum 20 characters, including spaces.

The tab key can be used to jump to the next field.

Owner team (team responsible for these messages)

Validity criteria:
Maximum 15 characters, including spaces.

The tab key can be used to jump to the next field.

Command

For detailed information on

- entering statements, see "General mask format"
- function key assignment, see "General mask format"

Follow-on operations:

ADD-DOCUMENTATION	-> ?

DUE	The inputs are confirmed and the routine returns to the MENU mask.
K1	The ADD-DOCUMENTATION function is aborted. The inputs are lost, even if the user previously paged to another ADD-DOCUMENTATION mask with “+” and then pressed DUE.

8.2.6.14 MODIFY-DOCUMENTATION mask - Modify, add and delete documentation lines

Mask sequence

MENU -> 10 -> MODIFY-DOCUMENTATION

Function

This mask allows the user to modify or delete documentation lines that were written for one or more message units. New lines can also be added. By contrast with ADD-DOCUMENTATION, here the user can access all documentation lines.

The documentation lines are displayed in alphabetical order. Each line can be modified as follows:

- modified by overwriting
- deleted by typing in spaces or null characters.

Modifications in the code field are not saved to the message file until the input is finally acknowledged with DUE. The message codes and message ranges are then re-sorted in the list.

DUE together with “+” or “-” pages forward and backward in the mask. Modifications to the documentation lines are not saved.

Mask

```
MODIFY-DOCUMENTATION
================================================================================
Message file: :N:$USER0001.TESTFILE
--------------------------------------------------------------------------------
Modify existing lines, type new ones. Data fields are optional.

   Msg-id, interval    Comment            Owner name            Owner team
   AAA0001-AAA0100                        John Miller           ABC 111
                    messages for test product
   AAA0001-AAA0010
                    messages for system commands
   AAA0011-AAA0020
                    messages for user commands
   AAA0021-AAA0030
                    device errors
   AAA0031-AAA0040
                    reserved messages
   AAA0041-AAA0041
                    output error
================================================================================
More (  +) / command ==>


           F2=prompt                K1=cancel     K2=interrupt     K3=refresh

LTG                                                              TAST
```

Output fields

Message file

The current message file is displayed. It was opened in the MENU mask. The file name cannot be modified in the MODIFY-DOCUMENTATION mask.

Input fields

Msg-id, interval (message code/message range)

A message code can be modified, deleted or added in this field.
If the code field is deleted without the documentation line being deleted, MSGMAKER outputs the following message: `MSMIF01`

Only when the documentation line is completely deleted is no further error message output. A message range without documentation does not result in an error message.

Validity criteria:
Seven-character message code, of which the first three characters must be letters (message class). The limit values for a message range must belong to the same message class.

The tab key can be used to jump to the next field.

Comment (comment on the messages)

Validity criteria:
Maximum 60 characters, including spaces.

The tab key can be used to jump to the next field.

Owner (person responsible for these messages)

Validity criteria:
Maximum 20 characters, including spaces.

The tab key can be used to jump to the next field.

Owner team (team responsible for these messages)

Validity criteria:
Maximum 15 characters, including spaces.

The tab key can be used to jump to the next field.

Command

For detailed information on

- entering statements, see "General mask format"
- function key assignment, see "General mask format"

Follow-on operations:

MODIFY-DOCUMENTATION	-> ?

DUE	The inputs are confirmed and the routine returns to the MENU mask.
K1	The MODIFY-DOCUMENTATION function is aborted. The inputs are lost, even if the user previously paged to another MODIFY-DOCUMENTATION mask with “+” or “-” followed by DUE.

8.2.6.15 DELETE-DOCUMENTATION mask - Delete documentation lines

Mask sequence

MENU -> 11 -> DELETE-DOCUMENTATION

Function

This mask allows documentation lines that were written for one or more message units to be deleted. Unlike the MODIFY-DOCUMENTATION mask, this mask only permits lines to be deleted that are marked with an “x” in the S column. The documentation lines themselves are overwrite-protected.

When DUE is pressed, the selected lines are deleted without the user being asked for confirmation.

> DUE together with “+” or “-” pages forward or backward in the mask. A selected documentation line is **not** deleted.

Mask

```
DELETE-DOCUMENTATION
------------------------------------------------------------------------------
Message file: :N:$USER0001.TESTFILE
------------------------------------------------------------------------------
Select lines to remove with the character 'x'.

S  Msg-id. interval    Comment            Owner name            Owner team
   AAA0001-AAA0100                       John Miller           ABC 111
                   messages for test product
   AAA0001-AAA0010
                   messages for system commands
   AAA0011-AAA0020
                   messages for user commands
   AAA0021-AAA0030
                   device errors
x  AAA0031-AAA0040
                   reserved messages
   AAA0041-AAA0041
                   output error
------------------------------------------------------------------------------
More (    ) / command ==>  +


            F2=prompt                K1=cancel     K2=interrupt     K3=refresh

LTG                                                              TAST
```

> *Note on the mask*
>
> Here, before finally deleting documentation line AAA0030-AAA0030, the '+' in the command line must be deleted. Only then will DUE also delete "Delete documentation line".

Output fields

- **Message file**
- **Msg-id, interval** (message code/message range)
- **Comment** (comment on the messages)
- **Owner** (person responsible for these messages)
- **Owner team** (team responsible for these messages)

Description see "MODIFY-DOCUMENTATION mask - Modify, add and delete documentation lines".

Input fields

S (mark column)

Validity criteria:
Letter 'x'.

Command

For detailed information on

- entering statements, see "General mask format"
- function key assignment, see "General mask format"

Follow-on operations:

DELETE-DOCUMENTATION	-> ?

DUE	The inputs are confirmed and the DELETE-DOCUMENTATION function is started; the routine returns to the MENU mask.
K1	The DELETE-DOCUMENTATION function is aborted and the routine returns to the MENU mask.

8.3 Statements

In batch jobs and procedures, message processing is controlled by means of statements. The operands and their possible values correspond in most cases to the mask fields and their entries.
Most statement names are the same as the corresponding mask titles.

Once MSGMAKER has been called, message processing usually begins with the //OPEN-MSG-FILE statement. Once opened, the message file can be processed using all the other functions. The COPY, MOVE, SHOW and MERGE-MSG-FILES functions can be called directly for processing the message file.

8.3.1 Overview of statements

The following overview contains all the statements that the MSGMAKER utility routine offers for message processing:

MSGMAKER statements	Function
//ADD-DOCUMENTATION	Add documentation (person responsible for messages, comment) for messages
//ADD-MSG	Add one or more message units to the current message file.
//COPY	Copy components of a message file (message units, documentation lines) from one message file to another or to a different location within the same message file.
//DELETE-DOCUMENTATION	Delete the documentation (person responsible for messages, comment) for messages
//DELETE-MSG	Delete one or more message units from the current message file.
//END	Terminate MSGMAKER; the open message files are closed.
//MERGE-MSG-FILES	Merge the entire contents of two or more message files in one message file
//MODIFY-DOCUMENTATION	Modify the documentation for messages
//MODIFY-MSG	Modify one or more message units in the current message file.
//MODIFY-OPTION	Specify whether (parts of) message units may be overwritten.
//MOVE	copy components of a message file (message units and documentation lines) from one message file to another or to a different location within the same message file. Unlike the //COPY statement, this statement also deletes the source area.
//OPEN-MSG-FILE	Open a message file.
//SHOW	Output message units and the corresponding documentation lines to SYSOUT or SYSLST.

In addition, the general SDF statements may be used. These statements are described in detail in the “SDF Dialog Interface” [20 (Related publications)].

8.3.2 Description of the statements

- ADD-DOCUMENTATION - Add documentation lines
- ADD-MSG - Add message unit
- COPY - Copy message unit
- DELETE-DOCUMENTATION - Delete documentation lines
- DELETE-MSG - Delete message unit
- END - Terminate MSGMAKER
- MERGE-MSG-FILES - Merge message files
- MODIFY-DOCUMENTATION - Modify and delete documentation lines
- MODIFY-MSG - Modify message unit
- MODIFY-OPTION - Overwrite message unit
- MOVE - Copy and delete message unit
- OPEN-MSG-FILE - Open message file
- SHOW - Display message file contents

8.3.2.1 ADD-DOCUMENTATION - Add documentation lines

Function

The //ADD-DOCUMENTATION statement allows message units to be documented. The documentation lines are assigned to the message units via the message code. In addition to the persons responsible for the message file, a 60-character comment can also be entered.
The correctness of the documentation and its relation to the messages are not checked.

If messages that have already been documented are documented again, the default setting in the //MODIFY-OPTION statement determines whether the documentation can be overwritten.

Differences compared with the //ADD-DOCUMENTATION statement in menu mode

The //ADD-DOCUMENTATION statement which can be entered in the command area of the screen mask differs from the //ADD-DOCUMENTATION statement in command procedures in that in menu mode, the operand value *PANEL-REQUEST can be assigned to any operand.

For further details, see section “Special features of statements in menu mode”.

Format

ADD-DOCUMENTATION

```
MSG-ID = *INTERVAL(...) / <name 7..7>

    *INTERVAL(...)

        |    FROM = <name 7..7>

        |    ,TO = <name 7..7>

,OWNER = *NONE / *PARAMETERS(...)

    *PARAMETERS(...)

        |    NAME = *NONE / <c-string 1..20>

        |    ,TEAM = *NONE / <c-string 1..15>

,COMMENTS = *NONE / <c-string 1..60>
```

Operands

MSG-ID = *INTERVAL(...) / <name 7..7>
Name of one or more message units to be documented.

MSG-ID = <name 7..7>
The full message code, consisting of the three-character message class and the four-digit message number, must be entered.

MSG-ID = *INTERVAL(...)
Designates a message range. The limit values of the range must belong to the same message class.

FROM = <name 7..7>
Designates the lower limit value of the message range (full message code).

TO = <name 7..7>
Designates the upper limit value of the message range (full message code).

OWNER = *NONE / *PARAMETERS(...)
Designates the persons responsible for the message file.

OWNER = *NONE
No specifications are made regarding the persons responsible.

OWNER = *PARAMETERS(...)

NAME = *NONE / <c-string 1..20>
Designates the person responsible for the message file.

TEAM = *NONE / <c-string 1..15>
Designates the team responsible for the message file.

COMMENTS = *NONE / <c-string 1..60>
Comment on the selected messages, comprising up to 60 characters.

8.3.2.2 ADD-MSG - Add message unit

Function

The //ADD-MSG statement adds a new message unit to the message file that is currently open.

In addition to the message code, which represents the address of the message unit within the message file, the user must define the message attributes, language identifiers and texts.

Message attributes include the access methods, output destination, routing code, weight code and message warranty. A message text must be written for each defined language identifier; meaning and response texts, however, can be entered as required. If the message text contains inserts, names and default texts can be assigned to these inserts in the INSERT-ATTRIBUTES operand. The message attributes and insert attributes are defined only once and are common to all the texts and languages of a message unit.

It is not possible to add texts in a new language to an existing message unit. If it were, the //ADD-MSG statement would cause the existing message unit to be overwritten. In batch mode and in interactive procedures, spinoff is triggered if the message is not overwritable (see the OVERWRITE operand of the //ADD-MSG and //MODIFY-OPTION statements).

Differences compared with the //ADD-MSG statement in menu mode

The //ADD-MSG statement that can be entered in the command area of the screen mask differs from the //ADD-MSG statement in command procedures in that

- in menu mode, the operand value *PANEL-REQUEST can be assigned to any operand
- additional values can be specified in the MSG-ID operand
- the OVERWRITE operand is not available in menu mode.

For further details, see section “Special features of statements in menu mode”.

Format

```
ADD-MSG

 MSG-ID = <name 7..7>

 ,ACCESS-METHODS = *ISAM / list-poss(4): *ISAM / *DLAM / *LOCAL-DLAM / *MINIMIP / *BAMR

 ,DESTINATIONS = *USER-TASK / *ALL(...) / list-poss(2): *USER-TASK / *CONSOLE(...)

     *ALL(...)

         |    ROUTING-CODE = <alphanum-name 1..1> / *MAIN-CONSOLE / *CONSLOG

     CONSOLE(...)

         |    ROUTING-CODE = <alphanum-name 1..1> / *MAIN-CONSOLE / *CONSLOG

 ,WEIGHT = *NONE / <integer 0..99>

 ,WARRANTY = *NO / *YES

 ,MSG-TEXT-OUTPUT = *UPPER-CASE / *LOWER-CASE

 ,LANGUAGES = list-poss(8): <name 1..1>(...)

     <name 1..1>(...)

         |    MSG-TEXT = <c-string 1..220 with-low>

         |    ,MEANING = *NONE / list-poss(256): <c-string 1..74 with-low>

         |    ,RESPONSE = *NONE / list-poss(256): <c-string 1..74 with-low>

 ,INSERT-ATTRIBUTES = *NONE / list-poss(30): <integer 0..29>(...)

     <integer 0..29>(...)

         |    NAME = *NONE / <structured-name 1..20>

         |    ,DEFAULT-VALUE = *NONE / <c-string 1..54 with-low> / *EMPTY-STRING

         |    ,AUTOMATIC-HELP = *NO / *YES(...)

         |         *YES(...)

         |              |    PREFIX = *BY-INSERT-VALUE / <name 3...7>

 ,OVERWRITE = *STD / *YES / *NO
```

Operands

MSG-ID = <name 7..7>
Identifies the message unit to be added. The full message code, consisting of the threecharacter message class and the four-digit message number, must be entered.

ACCESS-METHODS = *ISAM / list-poss(4): *ISAM / *DLAM / *LOCAL-DLAM / *MINIMIP / *BAMR
Specifies the methods that MIP uses to access the messages. Up to three methods can be specified; a combination of *ISAM and *DLAM, and *ISAM and *LOCAL-DLAM, however, is not permitted.

ACCESS-METHODS = *ISAM
MIP searches for messages via the ISAM key (= message code).

ACCESS-METHODS = *DLAM
The DLAM access method is used for particularly frequent messages. If a message file containing a DLAM message is activated, the DLAM message is loaded into main memory. MIP can output the DLAM message directly without accessing the message file.

ACCESS-METHODS = *LOCAL-DLAM / *MINIMIP / *BAMR
These access methods are reserved for internal use with the manufacturer.

DESTINATIONS = *USER-TASK / *ALL(...) / list-poss(2): *USER-TASK / *CONSOLE
This operand allows the creator of the message to document its possible output destinations. The output destination proper is determined in the DEST (destination code) operand of the MSG7X macro.

DESTINATIONS = *USER-TASK
The message output destination is SYSOUT, SYSLST or a user-specific memory area.

DESTINATIONS = *ALL(...)
All output destinations are possible.

> **ROUTING-CODE = <alphanum-name 1..1> / *MAIN-CONSOLE / *CONSLOG**
> The one-character routing code is evaluated as a destination specification for console outputs. The meanings of the routing codes are given in the “Introduction to System Administration” [5 (Related publications)].
>
> **ROUTING-CODE = <alphanum-name 1..1>**
> Any letter, digit or one of the special characters #, $ and @ can be specified as the routing code. The meanings of the preassigned routing codes are given in the “Introduction to System Administration” [5 (Related publications)].
>
> **ROUTING-CODE = *MAIN-CONSOLE**
> The message’s destination is the special routing code `*`, which is always allocated to the main console at least.
>
> **ROUTING-CODE = *CONSLOG**
> Messages that do not require a response are only logged in the CONSLOG file. ROUTING-CODE = *CONSLOG has the same effect as ROUTING-CODE = @.

DESTINATION = *CONSOLE(...)
The message output destination is a console. Further information is provided in the "Introduction to System Administration” [5 (Related publications)].

> **ROUTING-CODE = <alphanum-name 1..1> / *MAIN-CONSOLE / *CONSLOG**
> The one-character routing code is evaluated as a destination specification for console outputs. The meanings of the routing codes are given in the “Introduction to System Administration” [5 (Related publications)].
>
> **ROUTING-CODE = <alphanum-name 1..1>**
> Any letter, digit or one of the special characters #, $ and @ can be specified as the routing code. The meanings of the preassigned routing codes are given in the “Introduction to System Administration” [5 (Related publications)].

ROUTING-CODE = *MAIN-CONSOLE
The message's destination is the special routing code *, which is always allocated to the main console at least.

ROUTING-CODE = *CONSLOG
Messages that do not require a response are only logged in the CONSLOG file. ROUTING-CODE = *CONSLOG has the same effect as ROUTING-CODE = @.

WEIGHT = *NONE / <integer 0..99>
The weight code specifies the priority of a message.
A weight must be specified only for messages whose output destination is a console. Otherwise, *NONE is the default value. For further information on weight codes, see the "Introduction to System Administration" [5 (Related publications)].

WARRANTY = *NO / *YES
The message attribute "Warranty" is evaluated by MIP. The warranty declaration indicates that specific parts of the message will not be modified in future BS2000 versions.

The following message components are guaranteed:

- message code
- numbering and meaning of inserts

The message text is **not** guaranteed.
MIP creates S variables for warranty messages.
Further information is provided in the "Introduction to System Administration" [5 (Related publications)].

MSG-TEXT-OUTPUT = *UPPER-CASE / *LOWER-CASE
Specifies the output format of the message text. The default texts for inserts and the texts that are specified via the MSG7X macro are likewise adapted to this format. This does not apply to the meaning text and response text.

The possible output destinations are:

- SYSOUT
- a SYSLST file
- a user-specific memory area
- a console

The texts are output via the MSG7X macro or the /HELP-MSG-INFORMATION command.

MSG-TEXT-OUTPUT = * UPPER-CASE
The message text and the insert texts (default or current) are always converted into uppercase letters.

MSG-TEXT-OUTPUT = *LOWER-CASE
The message text and the insert texts (default or current) are output exactly as they are entered in the operand LANGUAGES = ...(MSG-TEXT='...').

LANGUAGES = list-poss(8): <name 1..1>(...)
Identifies the language in which the message, meaning and response texts are entered and output. The identifier consists of one letter; E stands for English and D for German. There are no conventions for other languages. The texts can be entered in up to eight different languages.

MSG-TEXT = <c-string 1..220 with-low>
A message text **must** be written for each message, which may be up to 220 characters long. This includes the strings (&00) through (&29) but not including the default texts that replace these strings on message output.

Separator “^”
The message text is entered as a continuous line of text. To structure the message text, the separator “^” can be used at any position in the message text as often as required. The text that follows the separator is output on the next line. Text output by the MSG7X macro or by the /HELP-MSG-INFORMATION command can be checked with the //SHOW statement.

Example

```
MSG-TEXT ='THIS MESSAGE IS OUTPUT TO ''(&04)''^ IN THE FOLLOWING

LANGUAGES; ''(&00)'', ''(&01)'' AND ''(&03)'''
```

The following text is output:

```
% ABC1234 THIS MESSAGE IS OUTPUT TO 'SYSOUT'

  IN THE FOLLOWING LANGUAGES; 'E', 'D' AND 'F'
```

For further information on the format of the message text, see the “Introduction to System Administration” [5 (Related publications)].

MEANING = *NONE / list-poss(256): <c-string 1..74 with-low>

A meaning text **may** be written in the corresponding language for each message. This text can contain up to 256 lines of 74 characters each. Further information is provided in the "Introduction to System Administration” [5 (Related publications)].

RESPONSE = *NONE / list-poss(256): <c-string 1..74 with-low>
A response text **may** be written in the corresponding language for each message. This text can contain up to 256 lines of 74 characters each. Further information is provided in the "Introduction to System Administration” [5 (Related publications)].

INSERT-ATTRIBUTES = *NONE / list-poss(30): insert-number <integer 0..29>(...)
Describes the inserts defined in the message text.

INSERT-ATTRIBUTES = *NONE
No attributes are assigned to the inserts.

INSERT-ATTRIBUTES = list-poss(30): insert-number <integer 0..29>(...)
Specifies the number of the insert for which a default text is defined. Further information is provided in the "Introduction to System Administration” [5 (Related publications)].

NAME = *NONE / <structured-name 1..20>
Specifies the insert name. Insert names are evaluated by MIP. MIP uses them to create S variables. The value specified in the DEFAULT-VALUE operand or the current value (MSG7X) becomes the contents of these S variables. Further information is provided in the "Introduction to System Administration” [5 (Related publications)].
The letters of the names entered here are converted to uppercase.

Example

The inserts with numbers 0 and 4 are to be given the names “LANGUAGE” and “DESTINATION”:

```
INSERT-ATTRIBUTES=(0(NAME=LANGUAGE),4(NAME=DESTINATION))
```

DEFAULT-VALUE = *NONE / <c-string 1..54 with-low> / *EMPTY-STRING
Defines the default text to be inserted in the message text in place of the insert if no current text is defined in the MSG7X macro. DEFAULT-VALUE = *NONE means that there is no default text for the insert. DEFAULT-VALUE = *EMPTY-STRING specifies an empty string as the insert text.
DEFAULT-VALUE = *NULL (= *EMPTY-STRING) continues to be supported in batch jobs and procedures for compatibility reasons. Output format of the default text depends on the MSG-TEXT-OUTPUT operand.

Example

The inserts with the numbers 4, 0 and 1 are to be assigned the texts “F”, “E” and “D”:

```
INSERT-ATTRIBUTES=(4(DEF='F'),0(DEF='E'),1(DEF='D'))
```

AUTOMATIC-HELP = *NO / *YES(...)
If a message code or part of a message code is output using an insert (e.g. in the case of DVS errors only the message number is output), the automatic help function causes MIP to output the associated message text in addition to this message code.

AUTOMATIC-HELP = *NO
Only the message code is output.

AUTOMATIC-HELP = *YES(...)
The message code and the message text are output.

PREFIX = *BY-INSERT-VALUE / <name 3...7>
The complete message text is output for an insert included in the error message or for an explicitly specified message number.

PREFIX = *BY-INSERT-VALUE
Possible outputs:

- message code with seven characters
- DMS error code with “0...”at the beginning. The DMS error class is prefixed by /HELP-MSG-INFORMATION
- four-character message code that is converted to a seven-character message code.

MSGMAKER does not check whether the default insert value contains a valid message code or a valid message number. If there is no valid message code, MIP issues the message “MESSAGE UNDEFINED”

The following rules are applicable to the length of the insert values and standard insert values

Length < 4:	value is left-padded with “0” up to four characters
Length > 4 and < 7:	value is truncated at the last 4 characters
Length > 7:	value is truncated at the last 7 characters.

PREFIX = <name 3...7>
For output in the command /HELP-MSG-INFORMATION the message code is formed from the prefix and insert value or default insert text. The first three characters indicate the message class and must be letters.

If the prefix corresponds to a full message code, the automatic help function is applied to it directly, irrespective of insert value.

The insert value is either padded with “0” or truncated to the left so that the result corresponds to a message code with seven characters.

Example

Prefix	Insert value	/HELP-MSG-INFORMATION
ABC	12	ABC0012
ABCX	123	ABCX123
ABCX	1234	ABCX234
ABC01	XYZ1234	ABC0134
ABC01	<none>	ABC01 (error)

If an insert number is defined repeatedly in a message text, the automatic help function is applied once only to this insert number.

In messages that require a response, MIP outputs the message a second time after executing the automatic help function. Automatic help is not activated if the message is sent to a user buffer.

OVERWRITE = *STD / *YES / *NO
Specifies whether the current message unit can overwrite an existing message unit with an identical message code.

OVERWRITE = *STD
The default value *STD refers to the last OVERWRITE specification made in the //MODIFY-OPTION statement (see "MODIFY-OPTION - Overwrite message unit").

OVERWRITE = *YES
The current message unit overwrites the existing message unit in its entirety.

OVERWRITE = *NO
The current message unit does not overwrite the existing message unit.

Example

MIP reports the DMS error code:

```
% DMS05F8 DMS error code '0533'. Command processing aborted.
  IN SYSTEM MODE: /HELP-MSG DMS(0533)
```

Using the automatic help function the appropriate message text is output

```
DMS0533 REQUESTED FILE NOT CATALOGED IN PUBSET '(&00)'. COMMAND TERMINATED
```

8.3.2.3 COPY - Copy message unit

Function

The //COPY statement copies message units, with or without the associated documentation lines, from one message file to another or to a different location within the same message file. The copying procedure does not change the message unit in the source file.

Each message unit is identified by means of its message code. It is possible to copy more than one message unit at a time by specifying a message class or message range in the MSG-ID operand. If the value *DOCUMENTATION is specified in the INFORMATION operand, the documentation lines defined in the specified message range are copied.

If the source file for the copying procedure is identical to the target file, new message codes must be defined in the TO-MSG-ID operand.
It is possible to copy message units between any two message files, irrespective of the file that is currently open. The FROM-FILE and TO-FILE operands permit access to these files; the current message file remains open and, if it is neither the source file nor the target file, is not changed.

If languages are specified explicitly in the INFORMATION=*MESSAGES(...) operand, only the texts (message, meaning and response) in these languages are copied to the target message unit. If the target unit does not exist, it is created with the message attributes (access methods, output destination, ...) of the source message unit.

If the target message unit already contains a text with the same message code and language identifier, the text of the target message unit is overwritten with the text of the source message unit.
If this occurs in **interactive mode**, an error message is issued asking the user whether or not the existing message unit is to be overwritten.
In **batch jobs** and **procedures**, the specified message file components are not copied and processing is continued, unless OVERWRITE=*YES was specified in the //COPY or //MODIFY-OPTION statement.

If the only difference between the texts is the language identifier, the text is appended to the contents of the target message unit.

> If the entire contents of a message file are to be copied to another file, the //MERGE-MSG-FILES statement should be used rather than the //COPY statement (see the //MERGE-MSG-FILES statement on page "MERGE-MSG-FILES Merge message files").

Differences compared with the //COPY statement in menu mode

The //COPY statement that can be entered in the command area of the screen mask differs from the //COPY statement in command procedures in that

- in menu mode, the operand value *PANEL-REQUEST can be assigned to any operand
- an additional value can be specified for the MSG-ID operand
- the OVERWRITE operand is not available in menu mode.

For further details, see section “Special features of statements in menu mode”.

Format

```
COPY

MSG-ID = *ALL / *CLASS(...) / *INTERVAL(...) / list-poss(2000): <name 7..7>

        *CLASS(...)

                |       MSG-CLASS = <name 3..3>

        *INTERVAL(...)

                |       FROM = <name 7..7>

                |       ,TO = <name 7..7>

,INFORMATION = *MESSAGES (...) / *ALL / list-poss(4): *DOCUMENTATION / *COMPONENT-ID /

                        *CORRECTION-INFO / *MESSAGES(...)

        *MESSAGES(...)

                |       LANGUAGES = *ALL / list-poss(8): <name 1..1>

,FROM-FILE = *CURRENT / <filename 1..54>

,TO-MSG-ID = *SAME / *CLASS(...) / <alphanum-name 4..7 with-wild>

        *CLASS(...)

                |       MSG-CLASS = <name 3..3>

,TO-FILE = *CURRENT / <filename 1..54> (...)

        <filename 1..54>(...)

                |       ,FILE-FORMAT = *NEW / *OLD                                1)

,OVERWRITE = *STD / *YES / *NO
```

1) The FILE-FORMAT operand is obsolete. It can still be specified for reasons of compatibility.

Operands

MSG-ID = *ALL / *CLASS(...) / *INTERVAL(...) / list-poss(2000): <name 7..7>
Specifies the message code of one or more message units of the opened message file that are to be copied.
If INFORMATION = *DOCUMENTATION is specified, the documentation lines defined for the message codes are also copied.

MSG-ID = *ALL
All the message components defined in the INFORMATION operand are copied. In this case, the TO-MSG-ID operand must be assigned the value *SAME.

MSG-ID = *CLASS(...)
All components of the messages in the specified message classes are copied.

MSG-CLASS = <name 3..3>
Specifies the 3-letter message class.

MSG-ID = *INTERVAL(...)
All components of the messages within the specified message range are copied.
If the message range contains more than one message class, the value *SAME must be assigned to the TO-MSG-ID operand.

FROM = <name 7..7>
Specifies the first message code of the message range.

TO = <name 7..7>
The full seven-character message code must be specified.

MSG-ID = list-poss(2000): <name 7..7>
Specifies the full seven-character message code. If a list of message codes is specified, the TO-MSG-ID operand must be assigned the value *SAME.

INFORMATION = *MESSAGES(...) / *ALL / list-poss(3): *MESSAGES(...) / *DOCUMENTATION / *CORRECTION-INFO / *COMPONENT-ID
Specifies the message file components to be copied.

INFORMATION = *MESSAGES(...)
Specifies message units to be copied. The message units are selected via the MSG-ID operand.

LANGUAGES = *ALL / list-poss(8): <name 1..1>
Specifies the language identifiers. Message, meaning and response texts in these languages are copied.

LANGUAGES = *ALL
The entire message unit is copied, including all message attributes, inserts, insert attributes and texts.

LANGUAGES = list-poss(8): <name 1..1>
Message texts and/or meaning and response texts in the selected language(s) are copied. The message attributes and insert attributes of the source message unit are only copied if the target message unit does not already exist. Otherwise the following the message `MSME108` is output.

INFORMATION = *ALL
All message file components (message units and documentation lines) are copied. If MSG-ID = *ALL is specified, all the contents of the message file are copied.

INFORMATION = *DOCUMENTATION
All documentation lines whose message codes are within the selected range (MSG-ID operand) are copied. If other defined message ranges overlap the specified message range, their documentation lines are also copied.

Example

MSG-ID = *INTERVAL (TST1500-TST1599).
The documentation lines of the message ranges (TST1500-TST1510), (TST1000-TST1999), (TST1000-TST1500), TST1510, ... are copied.

INFORMATION = *CORRECTION-INFO / *COMPONENT-ID
These operands are reserved for internal use with the manufacturer.

FROM-FILE = *CURRENT / <filename 1..54>
Name of the message file from which the specified components (MSG-ID operand) are to be copied.
The operand value *CURRENT refers to the current message file, i.e. the message file last opened using the //OPEN-MSG-FILE statement.

TO-MSG-ID = *SAME / *CLASS(...) / <alphanum-name 4..7 with-wild>
Specifies the new message codes of the copied message components in the target area.

TO-MSG-ID = *SAME
When copied, the message components retain their original message code.
The value *SAME must not be specified if message components are being copied to another location within the same file. The value *SAME must be specified if the MSG-ID operand is assigned a list of message codes, the value *ALL or message codes from more than one message class.

TO-MSG-ID = *CLASS(...)
Specifies a new message class for the copied message components. The message numbers remain unchanged.

MSG-CLASS = <name 3..3>
Specifies the new three-letter message class.

TO-MSG-ID = <alphanum-name 4..7 with-wild>
The new message code is specified either in full or as a partially qualified name containing wildcards.
The partial name determines the first part of the new message code; the remaining characters, symbolized by *, are transferred unchanged from the original message code.

Example

Old message code: MSG-ID = ABC1234;
TO-MSG-ID = TST0* changes the first four characters of the new message code to TST0 but retains the remaining three characters from the old message code. Result: TST0234.

TO-FILE = *CURRENT / <filename 1..54>(...)
Specifies the message file to which the message components are to be copied.

T**O-FILE = *CURRENT** The target file is the last message file opened with the //OPEN-MSG-FILE statement. As a target file it must be opened in UPDATE mode.

TO-FILE = <filename 1..54>
Explicit specification of the message file into which the message components are copied. If a target file does not exist, it is created and the file type (customer or standard), product name and product version are defined as for the source file.

OVERWRITE = *STD / *YES / *NO
Specifies whether the message components of the source file may overwrite an existing area of the target file.
The default value *STD corresponds to the value last specified in the OVERWRITE operand of the //MODIFY-OPTION statement.

Example

```
//COPY MSG-ID=TST000,INFORMATION=MESSAGE(LANGUAGE=E),-
  FROM-FILE=SYSMES.TSTFILE,TO-MSG-ID=*CLASS(TTO)
```

8.3.2.4 DELETE-DOCUMENTATION - Delete documentation lines

Function

The //DELETE-DOCUMENTATION statement is used to delete documentation lines for message units. The documented messages are selected using the message code.

Differences compared with the //DELETE-DOCUMENTATION statement in menu mode

The //DELETE-DOCUMENTATION statement which can be entered in the command area of the screen mask differs from the //DELETE-DOCUMENTATION statement in command procedures in that in menu mode, the operand value *PANEL-REQUEST can be assigned to any operand.

For further details, see section “Special features of statements in menu mode”.

Format

```
DELETE-DOCUMENTATION

MSG-ID = *ALL / *CLASS(...) / *INTERVAL(...) / <name 7..7>

    *CLASS(...)

        |    MSG-CLASS = <name 3..3>

    *INTERVAL(...)

        |    FROM = <name 7..7>

        |    ,TO = <name 7..7>

        |    ,DELETE-SUBSETS = *NO / *YES
```

Operands

MSG-ID = *ALL / *CLASS(...) / *INTERVAL(...) / <name 7..7>
Name of one or more message units whose documentation is to be deleted.

MSG-ID = *ALL
All documentation lines of the currently open message file will be deleted.

MSG-ID = *CLASS(...)

MSG-CLASS = <name 3..3>

The documentation lines which were written for the messages of the message class specified here will be deleted.

MSG-ID = *INTERVAL(...)
The documentation lines which were written for the messages of the range indicated here will be deleted. The limit values of the range must belong to the same message class.

FROM = <name 7..7>
Designates the lower limit value of the message range (full message code).

TO = <name 7..7>
Designates the upper limit value of the message range (full message code).

DELETE-SUBSETS = *NO / *YES
Enables precise determination of the documentation lines to be deleted which are defined in this message range.

DELETE-SUBSETS = *NO
Only those documentation lines will be deleted which were defined precisely for the message range designated by FROM, TO. The message range is documented in an allocation.

Example

```
//ADD-DOCUMENTATION MSG-ID=*INTERVAL(FROM=AAA0001,TO=AAA0020),
    OWNER=*PARAMETERS(NAME='Smith', TEAM='TST0815'),
    COMMENTS='documentation for messages AAA0001 to AAA0020'
//DELETE-DOCUMENTATION MSG-ID=*INTERVAL(FROM=AAA0001,TO=AAA0020,
    DELETE-SUBSETS=NO)
```

DELETE-SUBSETS = *YES
All documentation lines are deleted that lie within the message range defined by FROM and TO. The message range is documented via several allocations.

Example

```
//ADD-DOCUMENTATION MSG-ID=*INTERVAL(FROM=AAA0001,TO=AAA0008),
    OWNER=*PARAMETERS(NAME='Smith', TEAM='TST0815'),
    COMMENTS='documentation for messages AAA0001 to AAA0008'
//ADD-DOCUMENTATION MSG-ID=AAA0010, OWNER=*PARAMETERS(NAME='Smith',
    TEAM='TST0815'), COMMENTS='documentation for message AAA0010'
//ADD-DOCUMENTATION MSG-ID=*INTERVAL(FROM=AAA0015,TO=AAA0019),
    OWNER=*PARAMETERS(NAME='Smith', TEAM='TST0815'),
    COMMENTS='documentation for messages AAA0015 to AAA0019'
//DELETE-DOCUMENTATION MSG-ID=*INTERVAL(FROM=AAA0001,TO=AAA0020,
    DELETE-SUBSETS=*YES)
```

MSG-ID = <name 7..7>
The documentation for the full message code designated here will be deleted.

```
//ADD-DOCUMENTATION MSG-ID=*INTERVAL(FROM=AAA0001,TO=AAA0020),
    OWNER=*PARAMETERS(NAME='Smith', TEAM='TST0815'),
    COMMENTS='documentation for messages AAA0001 to AAA0020'
//DELETE-DOCUMENTATION MSG-ID=*INTERVAL(FROM=AAA0001,TO=AAA0020,
    DELETE-SUBSETS=NO)
```

8.3.2.5 DELETE-MSG - Delete message unit

Function

The //DELETE-MSG statement deletes one or more message units from the opened message file.
It is also possible to delete language-dependent message components such as message, meaning and response texts. The user should note, however, that the entire message unit is deleted automatically once all its texts have been deleted.

Differences compared with the //DELETE-MSG statement in menu mode

The //DELETE-MSG statement that can be entered in the command area of the screen mask differs from the //DELETE statement in command procedures in that

- in menu mode, the operand value *PANEL-REQUEST can be assigned to any operand.
- an additional value can be specified for the MSG-ID operand.

For further details, see section “Special features of statements in menu mode”.

Format

```
DELETE-MSG

MSG-ID = *ALL / *CLASS(...) / *INTERVAL(...) / list-poss(2000): <name 7..7>

        *CLASS(...)

                |       MSG-CLASS = <name 3..3>

        *INTERVAL(...)

                |       FROM = <name 7..7>

                |       ,TO = <name 7..7>

,LANGUAGES = *ALL / list-poss(8): <name 1..1>
```

Operands

MSG-ID = *ALL / *CLASS(...) / *INTERVAL(...) / list-poss(2000): <name 7..7>
Designates one or more message units which are to be modified.

MSG-ID = *ALL
All message units of the current message file are deleted.

MSG-ID = *CLASS(...)
Message units whose message code begins with the specified message class are deleted.

MSG-CLASS = <name 3..3>
Specifies the three-letter message class.

MSG-ID = *INTERVAL(...)
All message units whose message code is within the defined message range are deleted. The specified range limits must belong to the same message class.

FROM = <name 7..7>
Specifies the first message code of the message range.

TO = <name 7..7>
The full seven-character message code must be specified.

MSG-ID = list-poss(2000): <name 7..7>
The message unit to be deleted is specified by means of a full message code. A list of message codes may be specified.

LANGUAGES = *ALL / list-poss(8): <name 1..1>
Specifies the identifiers for the defined languages. Message, meaning and response texts in the specified language are deleted.

LANGUAGES = *ALL
The entire message unit is deleted, including all message attributes, insert attributes and texts.

LANGUAGES = list-poss(8): <name 8..8>
Up to eight language identifiers may be specified; the texts in these languages are deleted.

Example

```
//DELETE-MSG MSG-ID=*CLASS(TST),LANGUAGE=E
```

8.3.2.6 END - Terminate MSGMAKER

Function

The //END statement terminates MSGMAKER.

Differences compared with the //END statement in menu mode

The //END statement can be used in the command area of the screen mask in exactly the same way as it is used in command procedures.

When using the //END statement to terminate MSGMAKER in menu mode, however, the user should note that the current screen function will be aborted.

Format

END

8.3.2.7 MERGE-MSG-FILES - Merge message files

Function

The //MERGE-MSG-FILES statement allows the total contents of two or more message files to be merged into a new message file which should be empty or not cataloged. The total contents means that all messages, including the documentation lines, are included in the merge run. The LANGUAGES operand can be used to restrict the merge run to messages in specific languages.

Special points relating to a merge run

If the message files contain messages with the same message codes, MSGMAKER recognizes this. The following message is displayed:

```
% MSMQI00 IDENTICAL OBJECTS IN FILES 1='filename 1' AND 2='filename 2'.
  PRIORITY? REPLY (1;2;T=TERMINATE).
```

In this query, the user decides which message file should be given priority in terms of the message code. The same query is output each time two files are found with the same code; indeed, the query is also output when messages from different files are stored with the same message code but in different languages.
The identical messages in the non-prioritized file are not included in the target file.
In batch jobs and procedures, the message file named first in the FILE-NAMES operand is given priority if message codes are found to coincide. A message is output to inform the user when this happens.

The same check procedure is applied to identical documentation lines and decided on the basis of the specified priority.

> If the total contents of two or more message files are to be merged to make a new message file or if a message file is to be copied to another file, the //MERGE-MSG-FILES statement should always be used instead of the COPY function. Furthermore, the merged message file requires less memory space than a file created with COPY.
> No screen is available at present which corresponds to the //MERGE-MSG-FILES statement. The statement can be entered in the command line of the mask and started with DUE or F2.

Format

MERGE-MSG-FILES
FILE-NAMES = list-poss(2000): *CURRENT / <filename 1..80 with-wild> /   <partial-filename 2..79 with-wild> ,LANGUAGES = *ALL / list-poss(8): <name 1..1> ,TO-FILE = *CURRENT / <filename 1..54>

Operands

FILE-NAMES = list-poss(2000): *CURRENT / <filename 1..80 with-wild> /<partial-filename 2..79 with-wild>
Designates one or more message files which are to be merged. If only one name is specified, this amounts to the same as copying the file to the file specified by TO-FILE.

*CURRENT denotes the last file opened using the //OPEN-MSG-FILE statement is specified.

If a file is named more than once, MSGMAKER ignores all mentions of it except the first one.

LANGUAGES = *ALL / list-poss(8): <name 1..1>
Designates the languages in which the messages are stored.

LANGUAGES = *ALL
Die Meldungsdatei soll vollständig, d. h. alle Meldungen in allen Sprachen, für den Mischvorgang verwendet werden.

LANGUAGES = list-poss(8): <name 1..1>
Only the messages from the message file in the specified languages are to be used in the merge run.

TO-FILE = *CURRENT / <filename 1..54>
Output file in which the contents of the input files are merged. *CURRENT designates the last file opened in MSGMAKER.

The output file must be empty or not yet cataloged. The following rules apply to the file attributes:

- The data type of the output file is based on the data type of the input file. If there is at least one input file of the standard type, the output file assumes this file type. Input files of the customer type also result in an output file of the customer type.If the output file is assigned the customer type with the //OPEN-MSG-FILE statement and there is at least one input file of the standard type, MSGMAKER displays the warning `MSMWJ01`:
- If there are several input files, the file attributes product name and product version of the output file are not defined.
  If there is only one input file, its file attributes are transferred to the output file. In other words, the output file is the copy of the input file.

Example

```
//OPEN-MSG-FILE FILE-NAME=SYSMES.EKP.112, MODE=CREATE(TYPE=STANDARD,
      PRODUCT=BS2000(V200))
//MERGE-MSG-FILES FILE-NAMES=(SYSMES.INT*.200,SYSMES.COMP*.///,
      SYSMES.SPECIAL.123), TO-FILE=*CURRENT
```

8.3.2.8 MODIFY-DOCUMENTATION - Modify and delete documentation lines

Function

The //MODIFY-DOCUMENTATION statement is used to process documented messages in the following ways:

- Modify documentation lines
- Delete documentation options (only the variable options are deleted)

The documented messages are selected using the message code.

Differences compared with the //MODIFY-DOCUMENTATION statement in menu mode

The //MODIFY-DOCUMENTATION statement which can be entered in the command area of the screen mask differs from the //MODIFY-DOCUMENTATION statement in command procedures in that in menu mode, the operand value *PANEL-REQUEST can be assigned to any operand.

For further details, see section “Special features of statements in menu mode”.

Format

MODIFY-DOCUMENTATION

```
MSG-ID = *INTERVAL(...) / <name 7..7>

    *INTERVAL(...)

        |    FROM = <name 7..7>

        |    ,TO = <name 7..7>

,OWNER = *UNCHANGED / *NONE / *PARAMETERS(...)

    *PARAMETERS(...)

        |    NAME = *UNCHANGED / *NONE / <c-string 1..20>

        |    ,TEAM = *UNCHANGED / *NONE / <c-string 1..15>

,COMMENTS = *UNCHANGED / *NONE / <c-string 1..60>
```

Operands

MSG-ID = *INTERVAL(...) / <name 7..7>
Designates one or more message units which are to be modified.

MSG-ID = <name 7..7>
The full message code, consisting of the three-character message class and the four-digit message number, must be entered.

MSG-ID = *INTERVAL(...)
Designates a message range. The limit values of the range must belong to the same message class.

FROM = <name 7..7>
Designates the lower limit value of the message range (full message code).

TO = <name 7..7>
Designates the upper limit value of the message range (full message code).

OWNER = *UNCHANGED / *NONE / *PARAMETERS(...)
Designates the persons responsible for the message file.

OWNER = *UNCHANGED
The specifications regarding the persons responsible will not be changed.

OWNER = *NONE
No specifications are made regarding the persons responsible. Existing specifications are deleted.

OWNER = *PARAMETERS(...)

NAME = *UNCHANGED / *NONE / <c-string 1..20>
Designates the person responsible for the message file.

NAME = *UNCHANGED
The specifications regarding this person will not be changed.

NAME = *NONE
The specifications regarding this person will be deleted.

NAME = <c-string 1..20>
The specifications regarding this person will be modified.

TEAM = *UNCHANGED / *NONE / <c-string 1..15>
Designates the team responsible for the message file.

TEAM = *UNCHANGED
The specifications regarding the team will not be modified.

TEAM = *NONE
The specifications regarding the team will be deleted.

TEAM = <c-string 1..15>
The specifications regarding the team will be modified.

COMMENTS = *UNCHANGED / *NONE / <c-string 1..60>
Designates the comment for the selected messages.

COMMENTS = *UNCHANGED
The comment will not be changed.

COMMENTS = *NONE
The comment will be deleted.

COMMENTS = <c-string 1..60>
The comment will be modified.

Example

The message ranges (AAA0001-AAA0010) and (AAA0021-AAA0030) are documented as follows:

```
//ADD-DOCUMENTATION -
   MSG-ID=*INTERVAL(FROM=AAA0001,TO=AAA0010),-
   OWNER=*PARAMETERS(NAME='John Miller',TEAM='ABC 0001'),-
   COMMENTS='messages for test product'
//ADD-DOCUMENTATION -
   MSG-ID=*INTERVAL(FROM=AAA0021,TO=AAA0030),-
   OWNER=*PARAMETERS(TEAM='ABC')
```

Output to SYSOUT or SYSLST:

```
################################################################################
##################  M e s s a g e  c l a s s :  A A A  ##################
################################################################################

-----------------------  D O C U M E N T A T I O N  -----------------------

AAA0001-AAA0010    Owner name: John Miller            Team: ABC 0001
                   Messages for test product
AAA0021-AAA0030    Owner name:                        Team: ABC
```

The documentation is to be modified:

- AAA0001-AAA0010: Name to be modified to Harry Miller; Comment to be modified to “device errors”
- AAA0021-AAA0030: Specification regarding team to be deleted

```
//MODIFY-DOCUMENTATION -
   MSG-ID=*INTERVAL(FROM=AAA0001,TO=AAA0010),-
   OWNER=*PARAMETERS(NAME='Harry Miller',TEAM=*UNCHANGED),-
   COMMENTS='device errors'
//MODIFY-DOCUMENTATION -
   MSG-ID=*INTERVAL(FROM=AAA0021,TO=AAA0030),-
   OWNER=*PARAMETERS(TEAM=*NONE)
```

Output to SYSOUT or SYSLST:

```
################################################################################
##################  M e s s a g e  c l a s s :  A A A  ##################
################################################################################

-----------------------  D O C U M E N T A T I O N  -----------------------

AAA0001-AAA0010    Owner name: Harry Miller           Team: ABC 0001
                   Device errors
AAA0021-AAA0030    Owner name:                        Team:
```

8.3.2.9 MODIFY-MSG - Modify message unit

Function

The //MODIFY-MSG statement modifies a message unit of the message file that is currently open. One or more message units can be accessed by specifying a message code or a message range. In addition to the message attributes, it is possible to modify message, meaning and response texts, inserts and insert attributes. This statement allows the user not only to modify existing operand values but also to assign new values to the operands or to delete operand values.

Differences compared with the //MODIFY-MSG statement in menu mode

The //MODIFY-MSG statement that can be entered in the command area of the screen mask differs from the //MODIFY-MSG statement in command procedures in that

- in menu mode, the operand value *PANEL-REQUEST can be assigned to any operand.
- an additional value can be specified for the MSG-ID operand
- an additional value can be specified for the LANGUAGES operand

For further details, see section “Special features of statements in menu mode”.

Format

```
MODIFY-MSG
```

```
MSG-ID = *CLASS(...) / *INTERVAL(...) / list-poss(2000): <name 7..7>
    *CLASS(...)
        | MSG-CLASS = <name 3..3>
    *INTERVAL(...)
        | FROM = <name 7..7>
        | ,TO = <name 7..7>
,ACCESS-METHODS = *UNCHANGED / list-poss(4): *ISAM / *DLAM / *LOCAL-DLAM / *MINIMIP / *BAMR
,DESTINATIONS = *UNCHANGED (...) / *ALL(...) / list-poss(2): *USER-TASK / *CONSOLE(...)
    *UNCHANGED(...)
        | ROUTING-CODE = *UNCHANGED / <alphanum-name 1..1> / *MAIN-CONSOLE / *CONSLOG
    *ALL(...)
        | ROUTING-CODE = *UNCHANGED / <alphanum-name 1..1> / *MAIN-CONSOLE / *CONSLOG
    CONSOLE(...)
        | ROUTING-CODE = *UNCHANGED / <alphanum-name 1..1> / *MAIN-CONSOLE / *CONSLOG
,WEIGHT = *UNCHANGED / *NONE / <integer 0..99>
,WARRANTY = *UNCHANGED / *NO / *YES
,MSG-TEXT-OUTPUT = *UNCHANGED / *UPPER-CASE / *LOWER-CASE
```

```
,LANGUAGES = *UNCHANCHED / list-poss(8): <name 1..1>(...)
    <name>(...)
        |   MSG-TEXT = *UNCHANGED / <c-string 1..220 with-low>
        |   ,MEANING = *UNCHANGED / list-poss(2000): *ADD(...) / *INSERT(...) /
        |                       *REPLACE(...) / *REMOVE(...)
        |       *ADD(...)
        |           |   TEXT = list-poss(2000): <c-string 1..74 with-low>
        |       *INSERT(...)
        |           |   LINE-NUMBER = <integer 1..256>
        |           |   ,TEXT = list-poss(2000): <c-string 1..74 with-low>
        |       *REPLACE(...)
        |           |   LINE-NUMBER = <integer 1..256>
        |           |   ,TEXT = <c-string 1..74 with-low>
        |       *REMOVE(...)
        |           |   LINE-NUMBERS = *ALL / list-poss(2000): <integer 1..256>
        |   ,RESPONSE = *UNCHANGED / list-poss(2000): *ADD(...) / *INSERT(...) /
        |                           *REPLACE(...) / *REMOVE(...)
        |       *ADD(...)
        |           |   TEXT = list-poss(2000): <c-string 1..74 with-low>
        |       *INSERT(...)
        |           |   LINE-NUMBER = <integer 1..256>
        |           |   ,TEXT = list-poss(2000): <c-string 1..74 with-low>
        |       *REPLACE(...)
        |           |   LINE-NUMBER = <integer 1..256>
        |           |   ,TEXT = <c-string 1..74 with-low>
        |       *REMOVE(...)
        |           |   LINE-NUMBERS = *ALL / list-poss(2000): <integer 1..256>
```

```
,INSERT-ATTRIBUTES = *UNCHANGED / *NONE / list-poss(30): <integer 0..29>(...)

    <integer 0..29>(...)

        |   NAME = *UNCHANGED / *NONE / <structured-name 1..20>

        |   ,DEFAULT-VALUE = *UNCHANGED / *NONE / <c-string 1..54 with-low> / *EMPTY-STRING

        |   ,AUTOMATIC-HELP = *UNCHANGED / *NO / *YES(...)

        |           *YES(...)

        |                   |       PREFIX = *BY-INSERT-VALUE / <name 3...7>
```

Operands

MSG-ID = *CLASS(...) / *INTERVAL(...) / list-poss(2000): <name 7..7>
Specifies the message codes of one or more message units of the open message file that are to be modified.

MSG-ID = *CLASS(...)
All message units whose message code begins with the specified message class can be modified.

MSG-CLASS = <name 3..3>
Specifies the 3-letter message class.

MSG-ID = *INTERVAL(...)
All message units within this message range are modified.

FROM = <name 7..7>
Specifies the first message code of the message range.

TO = <name 7..7>
Specifies the last message code of the message range.

MSG-ID = list-poss(2000): <name 7..7>
Specifies the full message code of a message unit that is to be modified. A list of message codes may be specified.

ACCESS-METHODS = *UNCHANGED / list-poss(4): *ISAM / *DLAM / *LOCAL-DLAM / *MINIMIP / *BAMR
A new MIP message access method can be specified. Any declarations that were specified previously are deleted when the access method is modified.

ACCESS-METHODS = *UNCHANGED
The access methods remain unchanged.

ACCESS-METHODS = *ISAM
MIP searches for messages via the ISAM key.

ACCESS-METHODS = *DLAM<
The DLAM access method is used for particularly frequent messages. If a message file containing a DLAM message is activated, the DLAM message is loaded into main memory. MIP can output the DLAM message directly without accessing the message file.

ACCESS-METHODS = *LOCAL-DLAM / *MINIMIP / *BAMR
These access methods are reserved for internal use with the manufacturer.

DESTINATIONS = *UNCHANGED(...) / *ALL(...) / list-poss(2): *USER-TASK / *CONSOLE(...)
This operand allows the message creator to document new, possible output destinations for the message. This specification is used for documentation purposes only. The output destination proper is determined in the DEST (destination code) operand of the MSG7X macro.

DESTINATIONS = *UNCHANGED(...)
The message output destination remains unchanged.

ROUTING-CODE = *UNCHANGED / <alphanum-name 1..1> / *MAIN-CONSOLE / *CONSLOG
The one-character routing code is evaluated as a destination specification for console outputs. Further information is provided in the "Introduction to System Administration” [5 (Related publications)].

ROUTING-CODE = *UNCHANGED
The routing code remains unchanged.

ROUTING-CODE = <alphanum-name 1..1>
Any letter, digit or one of the special characters #, $ and @ can be specified as the routing code. The asterisk * must not be used. Further information is provided in the "Introduction to System Administration” [5 (Related publications)].

ROUTING-CODE = *MAIN-CONSOLE
The message’s destination is the special routing code *, which is always allocated to the main console at least.

ROUTING-CODE = *CONSLOG
Messages that do not require a response are only logged in the CONSLOG file. ROUTING-CODE = *CONSLOG has the same effect as ROUTING-CODE = @.

DESTINATIONS = *ALL(...)
Messages may be output to any destination.

ROUTING-CODE = *UNCHANGED / <alphanum-name 1..1> / *MAIN-CONSOLE / *CONSLOG
The one-character routing code is evaluated as a destination specification for console outputs. Further information is provided in the "Introduction to System Administration” [5 (Related publications)].

ROUTING-CODE = *UNCHANGED<

The routing code remains unchanged.

ROUTING-CODE = <alphanum-name 1..1>
Any letter, digit or one of the special characters #, $ and @ can be specified as the routing code. The asterisk * must not be used. Further information is provided in the "Introduction to System Administration” [5 (Related publications)].

ROUTING-CODE = *MAIN-CONSOLE
The message’s destination is the special routing code *, which is always allocated to the main console at least.

ROUTING-CODE = *CONSLOG
Messages that do not require a response are only logged in the CONSLOG file. ROUTING-CODE = *CONSLOG has the same effect as ROUTING-CODE = @.

DESTINATIONS = *USER-TASK
The new message output destination is SYSOUT, SYSLST or a user-specific memory area.

DESTINATIONS = *CONSOLE(...)
The message output destination is a console. If *CONSOLE is specified as the new operand value, a routing code **must** be defined. If *CONSOLE is replaced by other output destinations, the associated routing code is suppressed automatically.

ROUTING-CODE = *UNCHANGED / <alphanum-name 1..1> / *MAIN-CONSOLE / *CONSLOG
The one-character routing code is evaluated as a destination specification for console outputs. Further information is provided in the "Introduction to System Administration” [5 (Related publications)].

ROUTING-CODE = *UNCHANGED
The routing code remains unchanged.

ROUTING-CODE = <alphanum-name 1..1>
Any letter, digit or one of the special characters #, $ and @ can be specified as the routing code. The asterisk * must not be used. Further information is provided in the "Introduction to System Administration” [5 (Related publications)].

ROUTING-CODE = *MAIN-CONSOLE
The message’s destination is the special routing code *, which is always allocated to the main console at least.

ROUTING-CODE = *CONSLOG
Messages that do not require a response are only logged in the CONSLOG file. ROUTING-CODE = *CONSLOG has the same effect as ROUTING-CODE = @.

WEIGHT = *UNCHANGED / *NONE / <integer 0..99><
The weight code specifies the priority of a message. A weight must be specified for messages whose output destination is a console (DESTINATION = *CONSOLE(...)/*ALL was specified). Further information is provided in the "Introduction to System Administration” [5 (Related publications)].

WEIGHT = *UNCHANGED
The weight code remains unchanged.

WEIGHT = *NONE
The message is not assigned a weight.

WEIGHT = <integer 0..99>
The message is assigned a value between 0 and 99. 99 represents the highest priority for a message.

WARRANTY = *UNCHANGED / *NO / *YES
The message attribute “Warranty” is evaluated by MIP.

The following message components are guaranteed:

- message code
- numbering and meaning of inserts

The message text is **not** guaranteed.
MIP creates S variables for warranty messages. Further information is provided in the "Introduction to System Administration” [5 (Related publications)].

MSG-TEXT-OUTPUT = *UNCHANGED / *UPPER-CASE / *LOWER-CASE
Specifies the format of the message text when output to SYSOUT, a SYSLST file, a console or a user-specific memory area via the MSG7X macro or the /HELP-MSG-INFORMATION command. The default texts for inserts and the texts that are specified via the MSG7X macro are likewise adapted to this format.

MSG-TEXT-OUTPUT = *UNCHANGED
The format of the message text remains unchanged.

MSG-TEXT-OUTPUT = *UPPER-CASE
The message text entered is output in uppercase letters.

MSG-TEXT-OUTPUT = *LOWER-CASE
The message text is output exactly as it is entered in the operand LANGUAGES = ...(MSG-TEXT = '...').

LANGUAGES = *UNCHANGED / list-poss(8): <name 1..1>(...)
Specifies language identifiers for languages that are already defined. The user can modify message, meaning and response texts in these languages.

LANGUAGES = *UNCHANGED
No language identifiers are specified; the message, meaning and response texts are not to be modified.

LANGUAGES = list-poss(8): <name 1..1>(...)
The message, meaning and response texts in the languages represented by these identifiers are to be modified.

MSG-TEXT = *UNCHANGED / <c-string 1..220 with-low>
Specifies the message text that was written in the specified language.

MSG-TEXT = *UNCHANGED
The message text remains unchanged.

MSG-TEXT = <c-string 1..220 with-low>
Defines a new message text for the specified language.

MEANING = *UNCHANGED / list-poss: *ADD(...) / *INSERT(...) / *REPLACE(...) / *REMOVE(...)
Specifies the meaning text written in the specified language that will be displayed via the /HELP-MSG-INFORMATION command. Further information is provided in the "Introduction to System Administration” [5 (Related publications)].
Each line of the meaning text (up to 256 lines) can be modified or deleted. In addition, new lines can be appended or inserted. The lines of the meaning text are numbered internally from 1 through to a maximum of 256 and can be addressed individually via these numbers.

MEANING = *UNCHANGED
The meaning text remains unchanged.

MEANING = *ADD(...)
One or more new lines can be appended to the meaning text.

TEXT = list-poss: <c-string 1..74 with-low>
Meaning text consisting of one or more lines that is to be appended to the existing text.

MEANING = *INSERT(...)
One or more new lines can be inserted into the meaning text. Once all changes have been made, all the subsequent lines are renumbered.

LINE-NUMBER = <integer 1..256>
The new text is inserted between the line number specified here and the previous line number.

TEXT = list-poss: <c-string 1..74 with-low>
Specifies the new meaning text.

MEANING = *REPLACE(...)
The meaning text in the specified line is replaced by the new text entered.

LINE-NUMBER = <integer 1..256>
Specifies the number of the line containing text to be replaced.

TEXT = <c-string 1..74 with-low>
Specifies the text that is to replace the meaning text in the specified line.

MEANING = *REMOVE(...)
Lines can be deleted from the meaning text. Once all changes have been made, all subsequent lines are displaced and renumbered.

LINE-NUMBERS = *ALL / list-poss: <integer 1..256>
Specifies the line numbers.

Examples for the MEANING operand

Meaning text:
Line 1: `'texttext 1'`
Line 2: `'texttext 2'`
Line 3: `'texttext 3'`

Example 1

This example inserts a new line into the meaning text given above, replaces one line of text with another and appends a line to the meaning text.

```
MEANING=(*INSERT(2,'texttext 1 ext'),*REPLACE(2,'texttext 2 new'),
         *ADD('texttext 4')
```

Meaning text with renumbered lines:
Line 1: `'texttext 1'`
Line 2: `'texttext 1 ext'`
Line 3: `'texttext 2 new'`
Line 4: `'texttext 3'`
Line 5: `'texttext 4'`

Example 2

This example deletes the first line of the original meaning text and replaces the second line with a new line.

```
MEANING=(*REMOVE(1),*REPLACE(2,'texttext' 2 new) )
```

Meaning text with renumbered lines:
Line 1: `'texttext 2 new'`
Line 2: `'texttext 3'`

RESPONSE = *UNCHANGED / list-poss: *ADD(...) / *INSERT(...) / *REPLACE(...) / *REMOVE(...)
Specifies the response text written in the specified language that will be displayed via the /HELP-MSG-INFORMATION command. Further information is provided in the "Introduction to System Administration" [5 (Related publications)].
Each line of the response text (up to 256 lines) can be modified or deleted. In addition, new lines can be appended or inserted. The lines of the response text are numbered internally from 1 through to a maximum of 256 and can be addressed individually via these numbers. For examples, see the MEANING operand.

RESPONSE = *UNCHANGED
The response text remains unchanged.

RESPONSE = *ADD(...)
One or more new lines can be appended to the response text.

TEXT = list-poss: <c-string 1..74 with-low>
Response text consisting of one or more lines that is to be appended to the existing text.

RESPONSE = *INSERT(...)
One or more new lines can be inserted into the response text. Once all changes have been made, all subsequent lines are renumbered.

LINE-NUMBER = <integer 1..256>
The new text is inserted between the line number specified here and the previous line number.

TEXT = list-poss: <c-string 1..74 with-low>
Defines the new response text.

RESPONSE = *REPLACE(...)
The response text in the specified line is replaced by the new text entered.

LINE-NUMBER = <integer 1..256>
Specifies the number of the line containing text to be replaced.

TEXT = <c-string 1..74 with-low>
Specifies the text that is to replace the response text in the specified line.

RESPONSE = *REMOVE(...)
Lines can be deleted from the response text. Once all changes have been made, all subsequent lines are displaced and renumbered.

LINE-NUMBERS = *ALL / list-poss: <integer 1..256>
Specifies the line numbers.

INSERT-ATTRIBUTES = *UNCHANGED / *NONE / list-poss(30): <integer 0..29>(...)
The attributes of the specified inserts are not added, modified or deleted.

INSERT-ATTRIBUTES = *UNCHANGED
The attributes of the specified inserts are not modified.

INSERT-ATTRIBUTES = *NONE
All attributes of the specified inserts are deleted.

INSERT-ATTRIBUTES = <integer 0..29>(...)
The attributes of the specified inserts (numbered 00 through 29) can be added, modified or deleted; see also the //ADD-MSG statement.

NAME = *UNCHANGED / *NONE / <structured-name 1..20>
The insert name is to be modified. Insert names are evaluated by MIP.

DEFAULT-VALUE = *UNCHANGED / *NONE / <c-string 1..54 with-low> / *EMPTY-STRING
Modification of the default text that is inserted in the message text in place of the insert if no current text is defined in the MSG7X macro.

DEFAULT-VALUE = *UNCHANGED
The default text remains unchanged.

DEFAULT-VALUE = *NONE
The default text is deleted.

Example

```
INSERT-ATTRIBUTES=(0(NAME=*IDENTIFIER),4(NAME=*NONE),
2(DEFAULT-VALUE=*NONE),1(AUTO-HELP=*YES(CMD)))
```

DEFAULT-VALUE = <c-string 1..54 with-low>
Defines a new insert text. Note the length restrictions for message texts. Further information is provided in the "Introduction to System Administration” [5 (Related publications)].

DEFAULT-VALUE = *EMPTY-STRING
Defines an empty string. For compatibility reasons the DEFAULT-VALUE = *NULL operand value continues to be supported in batch jobs and procedures.

Example

```
INSERT-ATTRIBUTES=(0(DEFAULT-VALUE = *EMPTY-STRING),
1(DEFAULT-VALUE ='$TSOS'),4(DEFAULT-VALUE=*NONE))
```

AUTOMATIC-HELP = *UNCHANGED / *NO / *YES(...)
If a message code or part of a message code is output using an insert (e.g. in the case of DMS errors only the message number is output), the automatic help function causes MIP to output the associated message text in addition to this message code, see "ADD-MSG - Add message unit".

AUTOMATIC-HELP = *NO
Only the message code is output.

AUTOMATIC-HELP = *YES(...)
The message code and the message text are output

PREFIX = *BY-INSERT-VALUE / <name 3...7>
The complete message text is output for an insert included in the error message or for an explicitly specified message number.

8.3.2.10 MODIFY-OPTION - Overwrite message unit

Function

The //MODIFY-OPTION statement is available only in batch jobs and procedures.

When MSGMAKER is started, the //MODIFY-OPTION statement is used to define whether or not (parts of) message units may be overwritten. This setting remains valid for all subsequent //ADD-DOCUMENTATION, //ADD-MSG, //COPY and //MOVE statements until the next time a //MODIFY-OPTION statement is entered.

Format

MODIFY-OPTION
OVERWRITE = *UNCHANGED / *YES / *NO

Operands

OVERWRITE = *UNCHANGED / *YES / *NO
Specifies whether or not (parts of) message units or documentation lines may be overwritten. The OVERWRITE operand has the value *NO when the routine is started.

8.3.2.11 MOVE - Copy and delete message unit

Function

The //MOVE statement allows the user to copy message units, with or without the associated documentation lines, from one message file to another or to a different location within the same file. Unlike the COPY function, the MOVE function also deletes the source area.

Each message unit is identified by means of its message code. It is possible to move more than one message unit at a time by specifying a message class or a message range in the MSG-ID operand. If the value *DOCUMENTATION is specified in the INFORMATION operand, the documentation lines defined in the specified message range are also moved.

If the source file for the MOVE procedure is identical to the target file, new message codes must be defined in the TO-MSG-ID operand. It is possible to move message units between any two message files, irrespective of the file that is currently open. The FROM-FILE and TO-FILE operands permit access to these files; the current message file remains open and, if it is neither the source file nor the target file, is not changed.

If languages are specified explicitly in the operand INFORMATION=*MESSAGES(...), only the texts (message, meaning and response) in these languages are moved to the target message unit. If the target unit does not exist, it is created with the message attributes (access methods, output destination, ...) of the source message unit. If the target message unit already contains a text with the same message code and language identifier, the text of the target message unit is overwritten with the text of the source message unit.
If this occurs in **interactive mode**, an error message is issued asking the user whether or not the existing message unit is to be overwritten.
In **batch jobs** and **procedures**, the specified message file components are not moved and processing is continued, unless OVERWRITE=*YES was specified in the //MOVE or //MODIFY-OPTION statement.

If the only difference between the texts is the language, the text is appended to the contents of the target message unit.

Once all the language-dependent components have been deleted from a message unit by the //MOVE statement, the remaining message attributes and thus the message unit itself are deleted automatically.

Differences compared with the //MOVE statement in menu mode

The //MOVE statement that can be entered in the command area of the screen mask differs from the //MOVE statement in command procedures in that

- in menu mode, the operand value *PANEL-REQUEST can be assigned to any operand
- an additional value can be specified for the MSG-ID operand
- the OVERWRITE operand is not available in menu mode.

For further details, see section “Special features of statements in menu mode”.

Format

MOVE

```
MSG-ID = *ALL / *CLASS(...) / *INTERVAL(...) / list-poss(2000): <name 7..7>

      *CLASS(...)

            |     MSG-CLASS = <name 3..3>

      *INTERVAL(...)

            |     FROM = <name 7..7>

            |     ,TO = <name 7..7>

,INFORMATION = *MESSAGES(...) / *ALL / list-poss(4): *DOCUMENTATION / *COMPONENT-ID /

                        *CORRECTION-INFO / *MESSAGES(...)

      *MESSAGES(...)

            |     LANGUAGES = *ALL / list-poss(8): <name 1..1>

,FROM-FILE = *CURRENT / <filename 1..54>

,TO-MSG-ID = *SAME / *CLASS(...) / <alphanum-name 4..7 with-wild>

      *CLASS(...)

            |     MSG-CLASS = <name 3..3>

,TO-FILE = *CURRENT / <filename 1..54>

,OVERWRITE = *STD / *YES / *NO
```

Operands

MSG-ID = *ALL / *CLASS(...) / *INTERVAL(...) / list-poss(2000): <name 7..7>
Specifies the message codes of one or more message units of the open message file that are to be copied and then deleted from the source file.
If INFORMATION=*DOCUMENTATION is specified, the documentation lines defined for these message codes are also moved.

MSG-ID = *ALL
All message components defined in the INFORMATION operand are moved. This operand value may be specified only if the TO-MSG-ID operand is assigned the value *SAME.

MSG-ID = *CLASS(...)
All message components of the specified message class are moved.

MSG-CLASS = <name 3..3>
Specifies the three-letter message class.

MSG-ID = *INTERVAL(...)
All message components within this message range are moved. If the message range contains more than one message class, the value *SAME must be assigned to the TO-MSG-ID operand.

FROM = <name 7..7>
Specifies the first message code of the message range.

TO = <name 7..7>
The full seven-character message code must be specified.

MSG-ID = list-poss(2000): <name 7..7>
Specifies the full message code of a message component that is to be moved. If a list of message codes is specified, the TO-MSG-ID operand must be assigned the value *SAME. Several message codes may be specified as a list.

INFORMATION = *MESSAGES(...) / *ALL / list-poss(2000): *MESSAGES(...) /*DOCUMENTATION / *COMPONENT-ID / *CORRECTION-INFO
Specifies the message components to be moved.

INFORMATION = *MESSAGES(...)
Specifies message units that are to be moved. The message units are selected via the MSG-ID operand.

LANGUAGES = *ALL / list-poss(8): <name 1..1>
Specifies the languages of the message, meaning and response texts to be moved.

LANGUAGES = *ALL
The entire message unit is moved, including all message attributes, inserts, insert attributes and texts.

LANGUAGES = list-poss(8): <name 1..1>
Message texts and/or meaning and response texts in the selected language(s) are moved. The message attributes and insert attributes of the source message unit are moved to the target message unit only if the latter does not already exist. Otherwise the following the message `MSME108` is output.

INFORMATION = *ALL
All message file components (message units and documentation lines) are moved. If MSG-ID = *ALL is specified, all the contents of the message file are moved.

INFORMATION = *DOCUMENTATION
All documentation lines whose message codes are within the selected range (MSG-ID operand) are transferred. If other defined message ranges overlap the specified message range, their documentation lines are likewise moved.

Example

MSG-ID = *INTERVAL (TST1500-TST1599).
The documentation lines of the message ranges (TST1500-TST1510), (TST1000-TST1999), (TST1000-TST1500), TST1510, ... are moved.

MESSAGES = *CORRECTION-INFO / *COMPONENT-ID
These operands are reserved for internal use with the manufacturer.

FROM-FILE = *CURRENT / <filename 1..54 without-gen-vers>
Name of the message file from which the specified message components (MSG-ID operand) are to be transferred.

The operand value *CURRENT refers to the current message file, i.e. the message file last opened using the //OPEN-MSG-FILE statement. The file must have been opened in UPDATE mode.

TO-MSG-ID = *SAME / *CLASS(...) / <alphanum-name 4..7 with-wild>
Specifies the new message code for the message components once they have been moved to the target area.

TO-MSG-ID = *SAME
Once moved, the message components retain their original message code. The value *SAME must not be specified if message components are being moved to a different location within the same message file. The value *SAME must be specified if the MSG-ID operand is assigned a list of message codes, the value *ALL or message codes from several message classes.

TO-MSG-ID = *CLASS(...)
Specifies a new message class for the message components to be moved. The message numbers remain unchanged.

MSG-CLASS = <name 3..3>
Specifies the new three-letter message class.

TO-MSG-ID = <alphanum-name 4..7 with-wild>
The new message code is specified either in full or as a partially qualified name containing wildcards. The partial name determines the first part of the new message code; the remaining characters, symbolized by *, are transferred unchanged from the original message code.

Example

See the TO-MSG-ID operand of the //COPY statement and the following examples.

TO-FILE = *CURRENT / <filename 1..54 without-gen-vers>
Specifies the message file to which the message components are to be moved.

TO-FILE = *CURRENT
The assignment TO-FILE = *CURRENT defines the message file last opened with the //OPEN-MSG-FILE statement as the target file. The target file must have been opened in UPDATE mode.

TO-FILE = <filename 1..54 without-gen-vers>
Explicit specification of the message file into which the message components are to be moved.

OVERWRITE = *STD / *YES / *NO
Specifies whether the message component of the source file may overwrite an existing area of the target file. The default value *STD corresponds to the value last specified in the OVERWRITE operand of the //MODIFY-OPTION statement.

Example

```
//MOVE MSG-ID=*INTERVAL(TST000,TST0009),-
INFORMATION=(MESSAGE,DOCUMENTATION), TO-MSG-ID=TTT*, OVERWRITE=*NO
```

Examples for the MSG-ID and TO-MSG-ID operands

Example A:

The statement has the following effect:

```
//MOVE MSG-ID=*INTERVAL(CCCA000,CCCI999), TO-MSG-ID=DDD0*
```

CCCA001, CCCA002, ...	->	DDD0001, DDD0002, ...
CCCBAAA, CCCBAAB, ...	->	DDD1AAA, DDD1AAB, ...
.		.
.		
CCCE100, CCCE101, ...	->	DDD4100, DDD4101, ...
.		.
.		
CCCI998, CCCI999	->	DDD8998, DDD8999

The message class or partially qualified message code defined in the TO-MSG-ID operand is transferred to the new message code. The remaining part of the original message code appears in place of the wildcard character *.

Example B:

The statement has the following effect:

```
//MOVE MSG-ID = *INTERVAL(CCCA000,CCCI999), TO-MSG-ID=CCC0*
```

The four message codes <CCCA010, CCCA800, CCCD033, CCCH000> to be renamed as <CCC0010, CCC0800, CCC3033, CCC7000> and **not** as <CCC0010, CCC0800, CCC1033, CCC2000>.

If necessary, the part of the message number that belongs to the defined part of the new message code (in “CCC0*”, this is “0”), is incremented automatically by the predefined amount. The message class is not incremented.

Example C:

The following combinations are possible:

MSG-ID	TO-MSG-ID	Function	Insufficient range specification
*CLASS(CCC)	*CLASS(DDD)/ DDD*	New message class Message numbers are retained	DDD0*
(CCCH403, CCCH503, CCCH603)	*SAME	*SAME	All entries except *SAME
*INT(CCC0000-CCC4999)	*CLASS(DDD)/ DDD* CCC5* CCC50*	DDD0000, ... ,DDD4999 DDD0000, ... ,DDD4999 CCC5000, ... ,CCC9999 CCC5000, ... ,CCC9999	CCC6* / CCC51*
*INT(CCC0000, CCC0599)	*CLASS(DDD)/ DDD* DDD5* DDD55* DDD555*	DDD0000, ... ,DDD0599 DDD0000, ... ,DDD0599 DDD5000, ... ,DDD5599 DDD5500, ... ,DDD5599 DDD5600, ... ,DDD5599 DDD6A00, ... ,DDD6A99 DDD5550, ... ,DDD5599 DDD56A0, ... ,DDD5999 DDD6AA0, ... ,DDD6BE9	DDD95*

The message range of the target file must be at least as large as that of the source file.

8.3.2.12 OPEN-MSG-FILE - Open message file

Function

The //OPEN-MSG-FILE statement opens a message file for processing.

The message file remains open until another file is opened or until MSGMAKER is terminated.

The //OPEN-MSG-FILE statement does not always have to be issued before a file can be processed. In the COPY, MOVE and SHOW functions, a message file can be accessed directly by means of the FROM-FILE operand and in MERGE-MSG-FILES functions by means of the FILE-NAMES and TO-FILE operands.

Differences to menu mode

This statement cannot be entered in menu mode since only one message file can be open at any one time.

Format

```
OPEN-MSG-FILE

FILE-NAME = <filename 1..54>

,MODE = *UPDATE(...) / *CREATE(...) / *READ

    UPDATE(...)

        |   TYPE = *UNCHANGED / *CUSTOMER / *STANDARD

        |   ,PRODUCT = *UNCHANGED (...) / <structured-name 1..15>(...) / *NONE

        |       *UNCHANGED(...)

        |           |   VERSION = *UNCHANGED / <composed-name 3..8> / <c-string 1..8> / *NONE

        |       <structured-name 1..15>(...)

        |           |   VERSION = *UNCHANGED / <composed-name 3..8> / <c-string 1..8> / *NONE

    CREATE(...)

        |   TYPE = *CUSTOMER / *STANDARD

        |   ,PRODUCT = *NONE / <structured-name 1..15>(...)

        |       <structured-name 1..15>(...)

        |           |   VERSION = *NONE / <composed-name 3..8> / <c-string 1..8>
```

Operands

FILE-NAME = <filename 1..54>
Specifies the name of the message file that is to be opened for processing. This file remains the current file until another file is opened.

MODE = *UPDATE(...) / *CREATE(...) / *READ
Specifies the mode in which the message file is to be opened.

MODE = *UPDATE(...)
The message file is already cataloged and is to be updated.

TYPE = *UNCHANGED / *CUSTOMER / *STANDARD
Specifies the type of an existing message file.

TYPE = *UNCHANGED
The message file type remains unchanged.

> It is not possible to convert a standard message file into a customer message file by specifying MODE=*UPDATE(TYPE=*CUSTOMER).A customer message file, however, can be converted into a standard message file without restrictions.

TYPE = *CUSTOMER
The message file is assigned to a customer product.

TYPE = *STANDARD
The message file is associated with a software product developed by the manufacturer.

PRODUCT = *UNCHANGED(...) / <structured-name 1..15>(...) / *NONE
Name and version of the software product with which the current message file is associated. The letters entered are always converted to uppercase.

PRODUCT = *UNCHANGED(...)
The name of the software product remains unchanged.

VERSION = *UNCHANGED / <composed-name 3..8> / <c-string 1..8> / *NONESpecifies the product version. Letters entered are always converted to uppercase.

PRODUCT = <structured-name 1..15>(...)
New name of the product with which the current message file is associated.

VERSION = *UNCHANGED / <composed-name 3..8> / <c-string 1..8> / *NONE
Specifies the product version. Letters entered are always converted to uppercase.

PRODUCT = *NONE
No name or version is assigned to the product.

MODE = *CREATE(...)
A new message file is cataloged and becomes the current work file.

TYPE = *CUSTOMER / *STANDARD
Specifies whether the message file is associated with a customer product or a BS2000 product.

TYPE = *CUSTOMER
The message file is associated with a customer product.

TYPE = *STANDARD
The message file is associated with a software product developed by the manufacturer.

PRODUCT = *NONE / <structured-name 1..15>(...)
Specifies the name and version of the product with which the current message file is associated.

PRODUCT = *NONE
No name or version is assigned to the product.

PRODUCT = <structured-name 1..15>(...)
Name of the product with which the message file is associated.

VERSION = *NONE / <composed-name 3..8> / <c-string 1..8>
Specifies the product version. The letters entered are always converted to uppercase.

MODE = *READ
The message file is opened for read access only and must not be modified.

Example

Generate message file:

```
//OPEN-MSG-FILE FILE-NAME=SYSMES.TSTFILE,MODE=CREATE(TYPE=STANDARD,-
   PRODUCT=TSTPROD(VERSION=V01.0A10))
```

Update message file:

```
//OPEN-MSG-FILE SYSMES.TSTFILE,(PRODUCT=*UNCHANGED('V1.0A10'))
```

8.3.2.13 SHOW - Display message file contents

Function

The //SHOW statement outputs the contents of a message file to the screen or to a printer. The message units are sorted by message class and transferred with or without documentation lines to SYSOUT or SYSLST. For more information on the output format of the message file, see "SHOW-OUTPUT mask - Output message units and additional information".

Differences compared with the SHOW statement in menu mode

The //SHOW statement that can be entered in the command area of the screen mask differs from the //SHOW statement in command procedures in that

- in menu mode, the operand value *PANEL-REQUEST can be assigned to any operand
- an additional value can be specified for the MSG-ID operand

For further details, see section “Special features of statements in menu mode”.

Format

```
SHOW

MSG-ID = *ALL / list-poss(2000): *CLASS(...) / *INTERVAL(...) / <name 7..7>

    *CLASS(...)

        |  MSG-CLASS = <name 3..3>

    *INTERVAL(...)

        |  FROM = <name 7..7>

        |  ,TO = <name 7..7>

,INFORMATION = *MESSAGES (...) / *ALL / list-poss(4): *MESSAGES(...) / *DOCUMENTATION /

                    *COMPONENT-ID / *CORRECTION-INFO

    MESSAGES(...)

        |  LANGUAGES = *ALL / list-poss(8): <name 1..1>

        |  ,ELEMENTS = *ALL / list-poss(2000): *ATTRIBUTES / *MSG-TEXT /

        |                 *MEANING-RESPONSE / *INSERT-ATTRIBUTES

,FROM-FILE = *CURRENT / <filename 1..54>

,OUTPUT = *SYSOUT / *ALL / list-poss(2): *SYSOUT / *SYSLST
```

Operands

MSG-ID = *ALL / list-poss(2000): *CLASS(...) / *INTERVAL(...) / <name 7..7>

Specifies the message codes of one or more message units of the open message file that are to be output.
If INFORMATION=*DOCUMENTATION is specified, the documentation lines defined for these message codes are also output.

MSG-ID = *ALL
All message components defined in the INFORMATION operand are output.

MSG-ID = *CLASS(...)
All message units whose message code begins with the specified message class are displayed, with or without documentation lines (depending on the value specified in the INFORMATION operand).

MSG-CLASS = <name 3..3>
Specifies the three-letter message class.

MSG-ID = *INTERVAL(...)
All message units whose message code is within the specified message range are displayed, with or without documentation lines (depending on the value specified in the INFORMATION operand).

FROM = <name 7..7>
Specifies the first message code of the message range.

TO = <name 7..7>
The full seven-character message code must be specified.

MSG-ID = <name 7..7>
Specifies the full message code of a message unit that is to be displayed with or without documentation lines (depending on the value specified in the INFORMATION operand).

INFORMATION = *MESSAGES(...) / *ALL / list-poss: *MESSAGES(...) / *DOCUMENTATION / *COMPONENT-ID / *CORRECTION-INFO
Specifies the message file components to be displayed.

INFORMATION = *MESSAGES(...)
One or more message units are to be displayed.

LANGUAGES = *ALL / list-poss(8): <name 1..1>
Specifies the identifiers for the languages in which the message, meaning and response texts were defined.

LANGUAGES = *ALL
All language-dependent elements of a message unit are displayed in alphabetical order.

LANGUAGES = list-poss(8): <name 1..1>

Up to eight language identifiers may be specified. The elements of the message unit that were defined in these languages are displayed in the order in which the language identifiers are specified here.

ELEMENTS = *ALL / list-poss: *ATTRIBUTES / *MSG-TEXT / *MEANING-RESPONSE / *INSERT-ATTRIBUTES
Specifies elements of a message unit. The operand values *ATTRIBUTES and *INSERT-ATTRIBUTES do not depend on the definition of a language.

ELEMENTS = *ALL
All defined elements of a message unit are output, including the message attributes, insert attributes and message, meaning and response texts.

ELEMENTS = *ATTRIBUTES
All defined message attributes of the message unit are output.

ELEMENTS = *MSG-TEXT
The message text is output.

ELEMENTS = *MEANING-RESPONSE
The meaning and response texts are output.

ELEMENTS = *INSERT-ATTRIBUTES
All defined insert attributes are output.

INFORMATION = *ALL
All message file components defined in the message range (MSG-ID operand) are output. If the MSG-ID operand is assigned the value *ALL, all the contents of the message file are displayed.

INFORMATION = *DOCUMENTATION
Documentation lines in the specified message range (MSG-ID operand) are output. Documentation lines in message ranges that only partially overlap the specified message range are also displayed.

Example

If MSG-ID = *INTERVAL(TST1500-TST1599) is specified, all the documentation lines of the defined message ranges (TST1500-TST1510), (TST1000-TST1999), (TST1000-TST1500) and TST1510 are displayed.

INFORMATION = *CORRECTION-INFO / *COMPONENT-ID
These operands are reserved for internal use with the manufacturer.

FROM-FILE = *CURRENT / <filename 1..54 without-gen-vers>

Name of the message file whose message components are to be displayed.
The operand value *CURRENT refers to the message file that is currently open.

OUTPUT = *SYSOUT / *ALL / list-poss: *SYSOUT / *SYSLST
Specifies the output destination for the message file components defined above.

OUTPUT = *SYSOUT
The message file components are output to SYSOUT.

OUTPUT = *ALL
The message file components are output to SYSOUT and SYSLST.

OUTPUT = *SYSLST
The message file components are output to SYSLST. When MSGMAKER is terminated, the SYSLST file can be printed out using the BS2000 command /PRINT-DOCUMENT. Lines per page = 60.

Example

```
//SHOW MSG-ID=*INTERVAL (TSTAAA,TTT9999),
    INFORMATION=MESSAGES(LANGUAGES=(E,F),ELEMENTS=(ATTIBUTES,MSG-TEXT)),
    FROM-FILE=SYSMES.TSTFILE.OUTPUT=*SYSLST
```

8.3.3 Special features of statements in menu mode

In menu mode, the user enters statements in the three-line command area of the mask. The user can enter the operands consecutively, without continuation characters in these lines. The input is confirm by pressing the DUE key. Once all the operands have been entered correctly, the function is executed in the background; the fields of the current mask remain unchanged.

If a statement is entered with incorrect syntax, the user has the opportunity to correct the input in an SDF guided dialog.
Similarly, MSGMAKER reacts to missing operands by branching either to the SDF guided dialog or to the corresponding masks to allow the user to enter the required values. Once the function has been executed successfully, the calling mask is redisplayed

It is possible to apply the prompt function to the statement names entered instead of entering operands directly in the command area. In other words, pressing function key F2 after making an entry in the command area calls the mask that corresponds to the statement. It allows the user to specify the missing operands in the mask. The user then presses DUE to confirm the input and execute the statement. For further information on the prompt function, see "General mask format".
Calling the mask and executing the statement does not change the fields of the original mask.

Statements that can be entered in the command area of the screen mask differ from the statements that can be used in batch jobs and procedures with regard to certain operands or operand values.

*Notes on the operand value *PANEL-REQUEST*

It is possible to assign the ***PANEL-REQUEST** operand value to all operands of the statements entered in the command area of the screen mask.
*PANEL-REQUEST switches to the mask whose "panel-id" is, in most cases, the same as the statement name. The current value of the operand is displayed in the mask and can be modified using further statements or screen functions.
*PANEL-REQUEST is always the default value if no other default value (underscored value) is specified for the operands described in the section “Statements”.

Example

The following statement calls the MODIFY-MSG mask:

```
//MODIFY-MSG MSG-ID = TST0001, LAN=*PANEL-REQUEST
```

This mask displays all the message attributes, language identifiers and texts associated with the message code TST0001 that can be modified in the next operation.

The rest of this section describes only those statements whose operands can be assigned different or additional operand values in menu mode.

- If an additional operand value is available, this is indicated as follows: e.g. `... / <alphanum-name 1..7 with-wild> / ...` All operands which are not listed (indicated by ...) are available both in menu mode and in batch jobs and procedures.
- If an operand value in menu mode differs fundamentally from the operand value available in batch jobs and procedures, this is indicated clearly in the operand description.

Restrictions

- The //MODIFY-OPTION and //OPEN-MSG-FILE statements are available only in command procedures.
- Entering a question mark in the command area of a mask and then hitting DUE causes SDF to produce a selection from all available statements.

  If a standard message file of the manufacturer is open, it includes the following statements, which are reserved for internal use:

  - //ADD-COMPONENT-ID
  - //DELETE-COMPONENT-ID
  - //MODIFY-COMPONENT-ID
  - //ADD-CORRECTION-INFORMATION
  - //DELETE-CORRECTION-INFORMATION
  - //MODIFY-CORRECTION-INFORMATION
  - //GENERATE-CSECT

8.3.3.1 ADD-DOCUMENTATION

The //ADD-DOCUMENTATION statement is the same in menu mode and in command procedures except for the operand value *PANEL-REQUEST.

8.3.3.2 ADD-MSG

If the user attempts to add an existing message unit using the //ADD-MSG statement in menu mode, MSGMAKER switches automatically to the MODIFY-MSG mask and issues a warning.

Format (operands in menu mode)

```
ADD-MSG

MSG-ID = *CLASS(...) / *INTERVAL(...) / <alphanum-name 1..7 with-wild> / list-poss(2000): <name 7..7>

        *CLASS(...)

            |       MSG-CLASS = <name 3..3>

        *INTERVAL(...)

            |       FROM = <alphanum-name 1..7 with-wild>

            |       ,TO = <name 7..7>
```

The OVERWRITE operand no longer exists

Operands

MSG-ID = *CLASS(...) / *INTERVAL(...) / <alphanum-name 1..7 with-wild> /list-poss(2000): <name 7..7>
Specifies the message code of the message unit.
Unlike the //ADD-MSG statement in batch jobs and procedures, ADD-MSG in menu mode allows the user to specify a message class, a message range or a partially qualified message code. It is also possible to specify a list of message codes.

MSG-ID = *CLASS(...)
Defines a message class.

MSG-CLASS = <name 3..3>
Specifies the three-letter message class.

MSG-ID = *INTERVAL(...)
Defines a message range.

FROM = <alphanum-name 1..7 with-wild>
Specifies the first message code of the message range. The seven-character message code may be fully or partially qualified, the first three characters, however, must always be letters (message class). The asterisk (*) can stand for between one and seven characters, and the character # can stand for between one and four digits.

Example

Valid specifications for <alphanum-name 1..7 with-wild> are:
, A, AB*, ABC*, ABC0*, ABC00*, ABC000*
ABC#, ABC0#, ABC00#, ABC000#

TO = <name 7..7>
The full seven-character message code must be specified. The seven-character message code can be fully or partially qualified.

MSG-ID = <alphanum-name 1..7 with-wild>
The seven-character message code may be fully or partially qualified, the first three characters, however, must always be letters (message class). The asterisk (*) can stand for between one and seven characters, and the character # can stand for between one and four digits.

Example

See the operand MSG-ID=*INTERVAL(FROM=<alphanum-name 1..7 with-wild>)

MSG-ID = list-poss(2000): <name 7..7>
A list of full message codes can be specified.

8.3.3.3 COPY

Format (operands in menu mode)

```
COPY

MSG-ID = ... / <alphanum-name 1..7 with-wild> / ...

    *INTERVAL(...)

        |    FROM = <alphanum-name 1..7 with-wild>
```

The OVERWRITE operand no longer exists

Operands

MSG-ID = <alphanum-name 1..7 with-wild>
In addition to the values that can be specified for the MSG-ID operand in batch jobs and procedures, the message code can be defined in fully or partially qualified form in menu mode.
The first three characters of the seven-character message code (message class) must always be letters. The asterisk (*) can stand for between one and seven characters.

Example

Valid specifications for <alphanum-name 1..7 with-wild> are:
, A, AB*, ABC*, ABC0*, ABC00*, ABC000*

MSG-ID = *INTERVAL(...)
Defines a message range.

FROM = <alphanum-name 1..7 with-wild>
Specifies the first message code of the message range. The seven-character message code may be fully or partially qualified, the first three characters, however, must always be letters (message class). The asterisk (*) can stand for between one and seven characters.
Difference compared with the operand in batch jobs and procedures; in batch jobs and procedures, the message code must be fully qualified (FROM = <name 7..7>).

Example

See operand MSG-ID=<alphanum-name 1..7 with-wild>

8.3.3.4 DELETE-DOCUMENTATION

The //DELETE-DOCUMENTATION statement is the same in menu mode and in command procedures except for the operand value *PANEL-REQUEST.

8.3.3.5 DELETE-MSG

> The MSG-ID operand **cannot** be assigned the value *ALL in menu mode.

Format (operands in menu mode)

DELETE-MSG
MSG-ID = ... / <alphanum-name 4..7 with-wild> / ...

Operands

MSG-ID = <alphanum-name 4..7 with-wild>
In addition to the values for the MSG-ID operand in batch jobs and procedures (with the exception of *ALL), the message code may be defined in fully or partially qualified form in menu mode. The asterisk (*) can stand for between one and four characters.

Example

Valid specifications for <alphanum-name 4..7 with-wild> are:
ABC*, ABC0*, ABC00*, ABC000*

8.3.3.6 END

The //END statement is the same in menu mode and in command procedures.

8.3.3.7 GO-TO - Branch to specified mask

The //GO-TO statement is available only in menu mode.

This statement allows the user to deviate from the path set out by MSGMAKER (mask overview) and call any mask directly.
The //GO-TO statement can be entered in the command area of any mask and is initiated by pressing DUE. GO-TO is executed immediately, aborting the function of the current screen mask. Any data entered in this mask is lost.

Exception: MENU Mask

If a message file is entered in the MENU mask and not confirmed with DUE, the message file is still opened first and then the routine branches to the mask specified by GO-TO.

Format (operands in menu mode)

GO-TO / GOTO
PANEL-ID = *MENU / *MSG-FILE-ATTRIBUTES / *ADD-MSG / *MODIFY-MSG / *DELETE-MSG / *COPY / *MOVE / *SHOW / *ADD-DOCUMENTATION / *MODIFY-DOCUMENTATION / *DELETE-DOCUMENTATION

Operands

PANEL-ID = *MENU / *MSG-FILE-ATTRIBUTES / *ADD-MSG / MODIFY-MSG / *DELETE-MSG / *COPY / *MOVE / *SHOW / *ADD-DOCUMENTATION /*MODIFY-DOCUMENTATION / *DELETE-DOCUMENTATION
Name of the mask to which the user wishes to branch. The name can be abbreviated in accordance with SDF conventions.

Restriction

The //GO-TO statement cannot be used to branch to the MSG-TEXT, MEANING/RESPONSE or INSERT-ATTRIBUTES mask.

Example

GO-TO **add-msg** corresponds to GO-TO **a-m**

8.3.3.8 MERGE-MSG-FILES

The //MERGE-MSG-FILES statement is the same in menu mode and in command procedures.

There is currently no mask in which the MERGE-MSG-FILES function can be executed. The operand value *PANEL-REQUEST is therefore not available in menu mode.

The //MERGE-MSG-FILES statement can be started in menu mode either with F2 or with DUE.

8.3.3.9 MODIFY-DOCUMENTATION

The //MODIFY-DOCUMENTATION statement is the same in menu mode and in command procedures except for the operand value *PANEL-REQUEST.

8.3.3.10 MODIFY-MSG

Format (operands in menu mode)

```
MODIFY-MSG

MSG-ID = ... / <alphanum-name 1..7 with-wild> / ...

    *INTERVAL(...)

        FROM = <alphanum-name 1..7 with-wild>

,LANGUAGES = ... / *ANY(...) / ...

    *ANY(...)

      |  MSG-TEXT = *UNCHANGED

      |  ,MEANING = *UNCHANGED

      |  ,RESPONSE = *UNCHANGED
```

Operands

MSG-ID = <alphanum-name 1..7 with-wild>
In addition to the values that can be specified for the MSG-ID operand in batch jobs and procedures, the message code can be defined in fully or partially qualified form in menu mode.

The first three characters of the seven-character message code (message class) must always be letters. The asterisk (*) can stand for between one and seven characters, and the character # can stand for between one and four digits.

Example

Valid specifications for <alphanum-name 1..7 with-wild> are:
, A, AB*, ABC*, ABC0*, ABC00*, ABC000*
ABC#, ABC0#, ABC00#, ABC000#

MSG-ID = *INTERVAL(...)
Defines a message range.

FROM = <alphanum-name 1..7 with-wild>
Specifies the first message code of the message range. The seven-character message code may be fully or partially qualified, the first three characters, however, must always be letters (message class). The asterisk (*) can stand for between one and seven characters, and the character # can stand for between one and four digits.
Difference compared with the operand in batch jobs and procedures; in batch jobs and procedures, the message code must be fully qualified (FROM = <name 7..7>).

Example

See operand MSG-ID=<alphanum-name 1..7 with-wild>

LANGUAGES = *ANY(...)

In addition to the values that can be specified for the LANGUAGES operand in batch jobs and procedures, the operand value *ANY can be specified here. Within the structure initiated by *ANY, the operands MSG-TEXT, MEANING and RESPONSE may be assigned only the value *UNCHANGED or *PANEL-REQUEST. If LANGUAGES = *ANY is specified, no texts may be entered.

MSG-TEXT = *UNCHANGED

The message text is to remain unchanged. The contents of the MSG-TEXT mask can be modified only if the value *PANEL-REQUEST is specified.

MEANING = *UNCHANGED

The meaning text is to remain unchanged. The contents of the MEANING/RESPONSE mask can be modified only if the value *PANEL-REQUEST is specified.

RESPONSE = *UNCHANGED

The response text is to remain unchanged. The contents of the MEANING/RESPONSE mask can be modified only if the value *PANEL-REQUEST is specified.

8.3.3.11 MOVE

Format (operands in menu mode)

MOVE
MSG-ID = ... / <alphanum-name 1..7 with-wild> / ... *INTERVAL(...) | FROM = <alphanum-name 1..7 with-wild>

The OVERWRITE operand no longer exists

Operands

MSG-ID = <alphanum-name 1..7 with-wild>
In addition to the values that can be specified for the MSG-ID operand in batch jobs and procedures, the message code can be defined in fully or partially qualified form in menu mode.
The first three characters of the seven-character message code (message class) must always be letters. The asterisk (*) can stand for between one and seven characters.

Example

Valid specifications for <alphanum-name 1..7 with-wild> are:
, A, AB*, ABC*, ABC0*, ABC00*, ABC000*

MSG-ID = *INTERVAL(...)
Defines a message range.

FROM = <alphanum-name 1..7 with-wild>
Specifies the first message code of the message range. The seven-character message code may be fully or partially qualified. The first three characters, however, must always be letters (message class). The asterisk (*) can stand for between one and seven characters.
Difference compared with the operand in batch jobs and procedures; in batch jobs and procedures, the message code must be fully qualified (FROM = <name 7..7>).

Example

See operand MSG-ID=<alphanum-name 1..7 with-wild>

8.3.3.12 SHOW

Format (operands in menu mode)

SHOW
MSG-ID = ... / <alphanum-name 1..7 with-wild> / ... *INTERVAL(...) | FROM = <alphanum-name 1..7 with-wild>

Operands

MSG-ID = <alphanum-name 1..7 with-wild>
In addition to the values that can be specified for the MSG-ID operand in batch jobs and procedures, the message code can be defined in fully or partially qualified form in menu mode.
The first three characters of the seven-character message code (message class) must always be letters. The asterisk (*) can stand for between one and seven characters.

Example

Valid specifications for <alphanum-name 1..7 with-wild> are: *, A*, AB*, ABC*, ABC0*, ABC00*, ABC000*

MSG-ID = *INTERVAL(...)
Defines a message range.

FROM = <alphanum-name 1..7 with-wild>
Specifies the first message code of the message range. The seven-character message code may be fully or partially qualified. The first three characters, however, must always be letters (message class). The asterisk (*) can stand for between one and seven characters.
Difference compared with the operand in batch jobs and procedures; in batch jobs and procedures, the message code must be fully qualified (FROM = <name 7..7>).

Example

See operand MSG-ID=<alphanum-name 1..7 with-wild>

8.3.4 Example

In the TEST.MSG procedure the MSGMAKER routine first creates a message fileand a message unit is entered. Messages from the message file SYSMES.MSG.010 are copied to the new message file and modified. The contents of the newly created and subsequently modified message file SYSMES.TSTFILE are output to a SYSOUT file and a SYSLST file. You will find an example of the SYSOUT log after the description of the individual procedure steps.

TEST.MSG procedure

```
/BEGIN-PROCEDURE
/ASSIGN-SYSLST TO=TEST.MSG.SYSLST
/ASSIGN-SYSOUT TO=TEST.MSG.SYSOUT
/ASSIGN-SYSDTA *SYSCMD
/REMARK ************************************************************************
/REMARK ** Part 1 : Create a new message file
/REMARK ************************************************************************
/START-MSGMAKER ------------------------------------------------------------ (1)
//REMARK
//REMARK ************************ OPEN-MSG-FILE ***************************
//REMARK
//OPEN-MSG-FILE FILE-NAME=SYSMES.TSTFILE,-
// MODE=CREATE(TYPE=C,PRODUCT=TSTPROD(VERSION=V01.0A10)) -- (2)
//REMARK
//REMARK ************************ ADD-MSG *********************************
//REMARK
//ADD-MSG MSG-ID=TST0000,- ------------------------------------------------- (3)
// ACCESS-METHODS=ISAM,-
// DESTINATIONS=CONSOLE(ROUTING-CODE=A),-
// WEIGHT=30,-
// LANGUAGES=(E(-
// MSG-TEXT='Text in english with inserts ''(&&00)'' and ''(&&01)''',-
// MEANING=('First meaning line',-
// 'second meaning line'),-
// RESPONSE='Response line'-
// ),-
// D(-
// MSG-TEXT='Text in Deutsch mit Inserts ''(&&01)'' und ''(&&00)''',-
// MEANING=('Erste Bedeutungszeile',-
// 'Zweite Bedeutungszeile'),-
// RESPONSE=('Erste Massnahmeszeile',-
// 'Zweite Massnahmeszeile')-
// )-
// ),-
// INSERT-ATTRIBUTES=(0(NAME=NAM0,DEFAULT-VALUE='default0'),-
// 1(NAME=NAM1,DEFAULT-VALUE='default1'))
//REMARK
//REMARK ************************ SHOW ************************************
//REMARK
//SHOW MSG-ID=*ALL,OUTPUT=*ALL --------------------------------------------- (4)
//END
/REMARK ************************************************************************
/REMARK ** Part 2 : Messages from another message file
/REMARK ************************************************************************
/REMARK
/START-MSGMAKER ------------------------------------------------------------ (5)
//REMARK
```

```
//REMARK ************************ SHOW (other file) ************************
//REMARK
//SHOW MSG-ID=(PEP0001,PEP0002,PEP0004),- ———————————————————————— (6)
// INFORMATION=(MESSAGES(LANGUAGES=E,ELEMENTS=(MSG-TEXT,MEAN-RESP)),-
// DOCUMENTATION),-
// FROM-FILE=SYSMES.MSG.010,-
// OUTPUT=*ALL
//REMARK
//REMARK ************************ OPEN-MSG-FILE ****************************
//REMARK
//OPEN-MSG-FILE SYSMES.TSTFILE,(PRODUCT=*UNCHANGED('V01.1A10')) ———— (7)
//REMARK
//REMARK ***** COPY (from the other file) *********************************
//REMARK
//COPY MSG-ID=(PEP0001,PEP0002,PEP0004),- ———————————————————————— (8)
// INFORMATION=*ALL,-
// FROM-FILE=SYSES.MSG.010
//REMARK
//REMARK ************************ ADD-MSG **********************************
//REMARK
//ADD-MSG MSG-ID=PEP0005,- ———————————————————————————————————— (9)
// LANGUAGES=(E(-
// MSG-TEXT='File ''(&&05)'' does not exist',-
// ),-
// D(-
// MSG-TEXT='Datei ''(&&05)'' nicht vorhanden',-
// )-
// )
//REMARK
//REMARK ************************ MODIFY-MSG ******************************
//REMARK
//REMARK +++++ Modify the message attributes +++++
//REMARK
//MODIFY-MSG MSG-ID=(PEP0001,PEP0002),- ———————————————————————— (10)
// ACCESS-METHODS=DLAM,-
// DESTINATIONS=(USER-TASK,-
// CONSOLE(ROUTING-CODE=*MAIN-CONSOLE)),-
// INSERT-ATTRIBUTES=1(DEFAULT-VALUE= 'E')
//REMARK
//REMARK +++++ Modify the texts +++++
//REMARK
//MODIFY-MSG - ———————————————————————————————————————————— (11)
// MSG-ID=PEP0001,-
// LANG=E(-
// MEANING=(-
// *REPLACE(-
// LINE-NUMBER=1,-
// TEXT='For more detailed information about the DMS error code-
// enter /HELP-MSG in'),-
// *ADD('system mode or see the BS2000 manual ''System-
// Messages''.')-
// )-
// )
//REMARK
//REMARK ************************ MOVE (rename a message) ******************
//REMARK
//MOVE MSG-ID=PEP0004,- ———————————————————————————————————— (12)
// INFORMATION=(MESSAGES(LANGUAGES=E)),-
// TO-MSG-ID=TST*,-
```

```
// OVERWRITE=NO
//REMARK
//REMARK ************************ SHOW (new message file) ******************
//REMARK
//SHOW MSG-ID=(*CLASS(PEP),- ------------------------------------------------ (13)
// TST0004),-
// INFORMATION=(MESSAGES,DOCUMENTATION),-
// OUTPUT=*ALL
//REMARK
//REMARK ************************ DELETE-MSG *******************************
//REMARK
//DELETE-MSG MSG-ID=*CLASS(TST),LANGUAGES=(D,E) ---------------------------- (14)
//END
/END-PROCEDURE
```

Explanation

1. MSGMAKER is called.
2. Message file SYSMES.TSTFILE is opened for the TSTPROD product with version V01.0A10.
3. A message with message code TST0000 is entered in the newly opened message. Apart from message attributes (access method, output target, weight code), the message, meaning and response texts are entered in English and German. The message texts contain inserts (&00) and (&01). If the inserts are to be enclosed in single quotes for the message output, double quotes must be input in the statement. Please note also that the ampersand character (&) is written twice.
4. The contents of the SYSMES.TSTFILE message file are output to SYSOUT and SYSLST.
5. MSGMAKER is called again.
6. Three messages from the message file SYSMES.MSG.010 are output to SYSOUT and SYSLST. The output includes the English message, meaning and response texts, as well as the documentation lines.
7. The message file SYSMES.TSTFILE is opened again, this time with a different version specification (V01.1A10).
8. Three messages, including documentation, are copied from the message file SYSMES.MSG.010 to the SYSMES.TSTFILE file.
9. The new message PEP0005 is entered in the SYSMES.TSTFILE message file.
10. The access method, the output destination and the default text for insert (&01) are modified for the two messages PEP0001 and PEP0002.
11. The English meaning text is modified for message PEP0001. The first line is replacedby a new text and a second line is added.
12. All the English texts of message PEP0004 are stored under message code TST0004.
13. The messages of message class MSM and message TST0004 are output to SYSOUT and SYSLST.
14. The English and German texts of message TST0004 are deleted.

Output to SYSOUT

ad (4):

```
################################################################################
###################   M e s s a g e   c l a s s :   T S T   ###################
################################################################################
-- TST0000 -----------------------------------------------------------------
Access     : ISAM
Destination: CONSOLE              Routing code: A                Weight: 30
Warranty   : NO                   Text format : UPPER CASE
Insert attributes:
(&00)  Name: NAM0
       Default value: 'default0'
(&01)  Name: NAM1
       Default value: 'default1'
D   Text in Deutsch mit Inserts '(&01)' und '(&00)'
  ? Erste Bedeutungszeile
    Zweite Bedeutungszeile
  ! Erste Massnahmezeile
    Zweite Massnahmezeile
E   Text in english with inserts '(&00)' and '(&01)'
  ? First meaning line
    second meaning line
  ! Response line
%  MSMN600 MSGMAKER TERMINATED NORMALLY
```

ad (6):

```
################################################################################
###################   M e s s a g e   c l a s s :   P E P   ###################
################################################################################
-- PEP0001 -----------------------------------------------------------------
E   MESSAGE WITH INSERT (&01)
  ? meaning for pep0001
  ! response for pep0001
-- PEP0002 -----------------------------------------------------------------
E   MESSAGE WITH INSERT (&01)
  ? meaning for pep0002
  ! response for pep0002
-- PEP0004 -----------------------------------------------------------------
E   MESSAGE WITH INSERT (&01)
  ? meaning for pep0004
  ! response for pep0004
=======================   D O C U M E N T A T I O N   ====================
PE0001-PEP0004     Owner name: MAIER                  Team: TEAM1
                   product = DMS-ERR
```

ad (13):

```
################################################################################
###################   M e s s a g e   c l a s s :   P E P   ###################
################################################################################
-- PEP0001 ---------------------------------------------------------------------
Access     : DLAM
Destination: USER-TASK, CONSOLE  Routing code: * (main console)   Weight: 99
Warranty   : NO                  Text format : UPPER CASE
Insert attributes:
(&01)  Default value: 'E'
E   MESSAGE WITH INSERT (&01)
  ? For more detailed information about the DMS error code enter /HELP-MSG
    in system mode or see the BS2000 manual 'System Messages'.
  ! response for pep0001
-- PEP0002 ---------------------------------------------------------------------
Access     : DLAM
Destination: USER-TASK, CONSOLE   Routing code: * (main console)  Weight: 99
Warranty   : NO                   Text format : UPPER CASE
Insert attributes:
(&01)  Default value: 'E'
E   MESSAGE WITH INSERT (&01)
  ? meaning for pep0002
  ! response for pep0002
-- PEP0005 ---------------------------------------------------------------------
Access     : ISAM
Destination: USER-TASK           Routing code:                    Weight: 99
Warranty   : NO                  Text format : UPPER CASE
D   Datei '(&05)' nicht vorhanden
E   File '(&05)' does not exist
```

```
========================   D O C U M E N T A T I O N   =====================
PEP0001-PEP0004    Owner name: MAIER                  Team: TEAM1
                  product = DMS-ERR
################################################################################
###################   M e s s a g e   c l a s s :   T S T   ###################
################################################################################
-- TST0004 ---------------------------------------------------------------------
Access     : ISAM
Destination: USER-TASK            Routing code:                    Weight: 99
Warranty   : NO                   Text format : UPPER CASE
E   MESSAGE WITH INSERT (&01)
  ? meaning for pep0004
  ! response for pep0004
```

Output to SYSLST

Output of the contents of the message file to SYSLST is almost the same as the output to SYSOUT.

Output of the messages and documentation lines is identical. Column 1 of the SYSLST log is, however, reserved for EBCDIC control characters to enable the output to be sent to a printer. The new page is headed by an information line containing the date, time, product name, product version and page number.

9 PAMCONV Conversion of file formats

Version: PAMCONV V21.0A

The PAMCONV utility routine is used for converting files from K format to NK format or vice versa.

Files encrypted with a crypto password are also supported (see "Functionality of PAMCONV").

In **K format**, the DMS management information is stored in a PAM key which is prefixed to the data block. This file format is referred to as key format, or K format for short.

In **NK format**, which does not use the PAM key, the DMS management information is either integrated in the data blocks or it is left out. This file format is known as nonkey format, or NK format for short.

The term NK file serves as a generic term for NK2 and NK4 files.
These file formats (K, NK2 and NK4) were created so as to be enable the data management system to make the best possible use of the existing disk formats.

The minimum transfer unit between the disk and main memory as well as the size of the smallest file (minimum allocation unit=min. AU) are depending on the file format.

Files on NK4 disks must always be in NK4 format. PAMCONV offers the possibility of converting K or NK2 files to NK4 format.

Disk formats

A disk format is defined by the criteria “with/without PAM key”, “minimum allocation unit (min. AU)” and “minimum transfer unit (min. TU)”.

The disk format within any one pubset is homogeneous (except SM pubsets).
For private disks the only formats supported are the K and NK2 disks with a minimum allocation unit of 6 Kbytes.

The following disk formats are supported:

Disk format	PAM key		min. AU			min. TU	
	with	without	6 KB	8 KB	64 KB	2 KB	4 KB
K disk	x		x			x	
NK2 disk		x	x			x	
NK2 disk		x		x		x	
NK2 disk		x			x	x	
NK4 disk		x		x			x
NK4 disk		x			x		x

The diagram below shows which file formats can be stored on the supported disk formats without the file formats first having to be converted using PAMCONV.

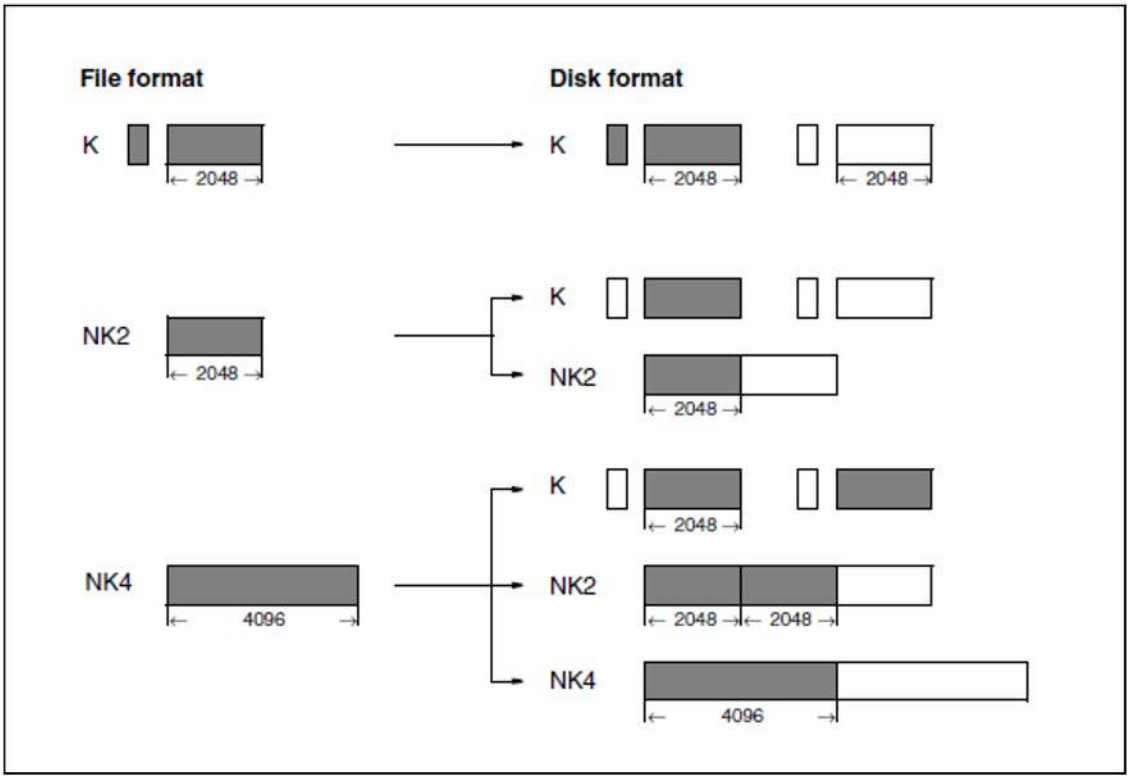
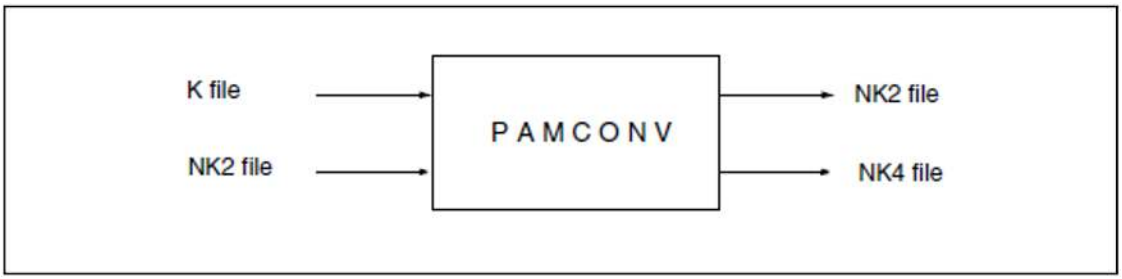


Figure 12: File formats that can be stored on certain disk formats without conversion

The primary purpose of PAMCONV is to convert K files into NK files so that the latter can be stored in NK pubsets.



Conversion options with PAMCONV

The conversion options provided by PAMCONV for the individual file structures are listed in the table below.

Conversion options with PAMCONV

File structure	File format1 <--> file format2
ISAM	K-ISAM <---> NK2-ISAM K-ISAM <---> NK4-ISAM NK2-ISAM <---> NK4-ISAM
SAM	K-SAM <---> NK2-SAM K-SAM <---> NK4-SAM NK2-SAM <---> NK4-SAM
PAM	K-PAM <---> NK-PAM K-PAM <---> NK4-PAM NK2-PAM <---> NK4-PAM
Load module	K module <---> NK2 module K module <---> NK4 module NK2 module <---> NK4 module

9.1 Starting the program run

The program PAMCONV is started with `/START-PAMCONV.`

START-PAMCONV	Alias: **PAMCONV**
VERSION = ***STD** / <product-version> **,MONJV** = ***NONE** / <filename 1..54 without-gen-vers> **,CPU-LIM**IT = ***JOB-RE**ST / <integer 1..32767 *seconds*>	

Command statements of the routine can be entered once the program is running.

PAMCONV is terminated using the END statement.

It can also be called as follows:

```
/START-EXECUTABLE-PROGRAM FROM-FILE=$PAMCONV
```

9.2 Functionality of PAMCONV

The PAMCONV utility routine provides the user with two basic functions for adapting files to the available disk format.

- File format conversion

  One of the functions of the PAMCONV routine is to convert files from K format to NK format (and vice versa). File format conversion is performed using statements, specifically the CONVERT-FILE statement.
- Reblocking

  PAMCONV can be used to convert the blocking factor of a file from an odd number to an even number. Only files with an even-numbered blocking factor can be stored on an NK4 disk.

Functionality of PAMCONV

The following tables provide an overview of the conversion options for ISAM, SAM, PAM and load module files.

In addition, the basic directions of conversion are shown in diagram form for each file structure. The block structure of the file is shown before and after conversion.

ISAM file

An ISAM file that is to be converted with PAMCONV may have one of the following three block structures.

- **PAMKEY**: the file is a K-ISAM file.
- **DATA2K**: the file is an NK2-ISAM file; the blocking factor n may be odd or even.
- **DATA4K**: the file is an NK4-ISAM file; the blocking factor n is even.

The table below summarizes all conversion options for an ISAM file:

Conversion options for an ISAM file

Source	Conversion options	Target
PAMKEY	------>	DATA2K
PAMKEY	------>	DATA4K
DATA2K	------>	PAMKEY
DATA2K	------>	DATA4K
DATA4K	------>	PAMKEY
DATA2K	<------>	DATA2K
DATA4K	<------>	DATA4K
DATA4K	------>	DATA2K

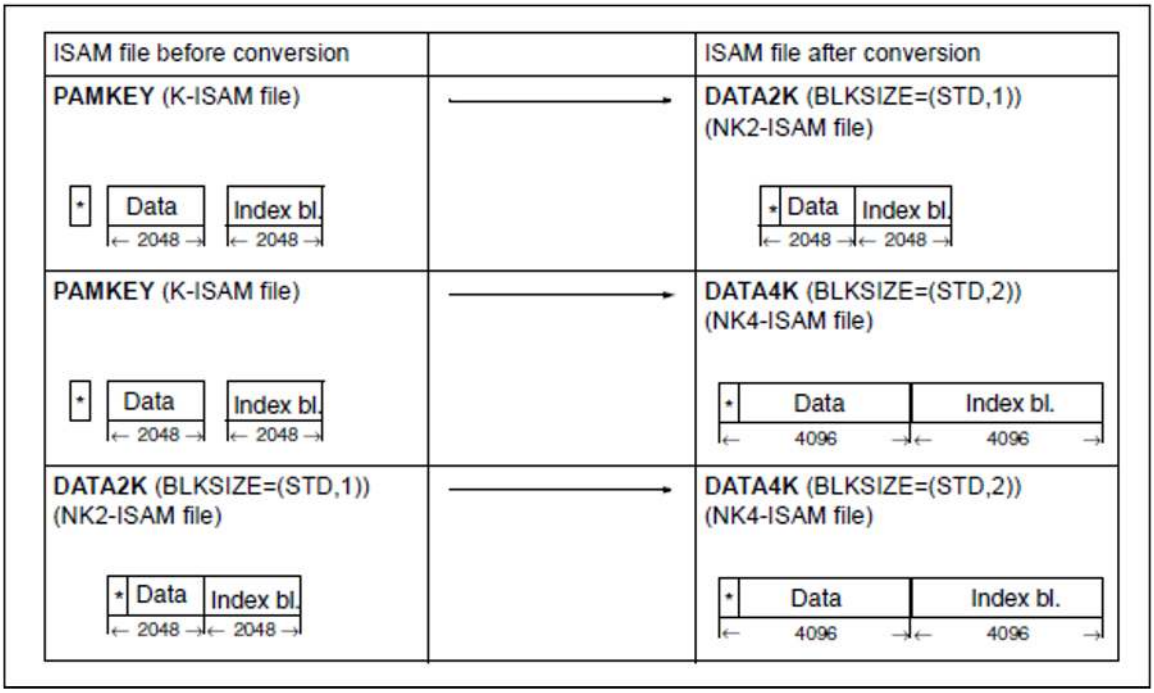
MLU macro libraries are ISAM files and must be converted as such.

What block structure an ISAM file has before and after file format conversion is illustrated for the following directions of conversion:

Source	Conversion options	Target
PAMKEY	------>	DATA2K
PAMKEY	------>	DATA4K
DATA2K	------>	DATA4K

Key:

*	16 bytes block management information
BLKSIZE	Logical block length
Index bl.	Index block



SAM file

A SAM file that is to be converted with PAMCONV may have either of the following two block structures:

- **PAMKEY**: the file is a K-SAM file.
- **DATA**: the file is an NK2-SAM file if the blocking factor n is odd, or an NK4-SAM file if n is even.

The table below summarizes all conversion options for a SAM file:

Conversion options for a SAM file

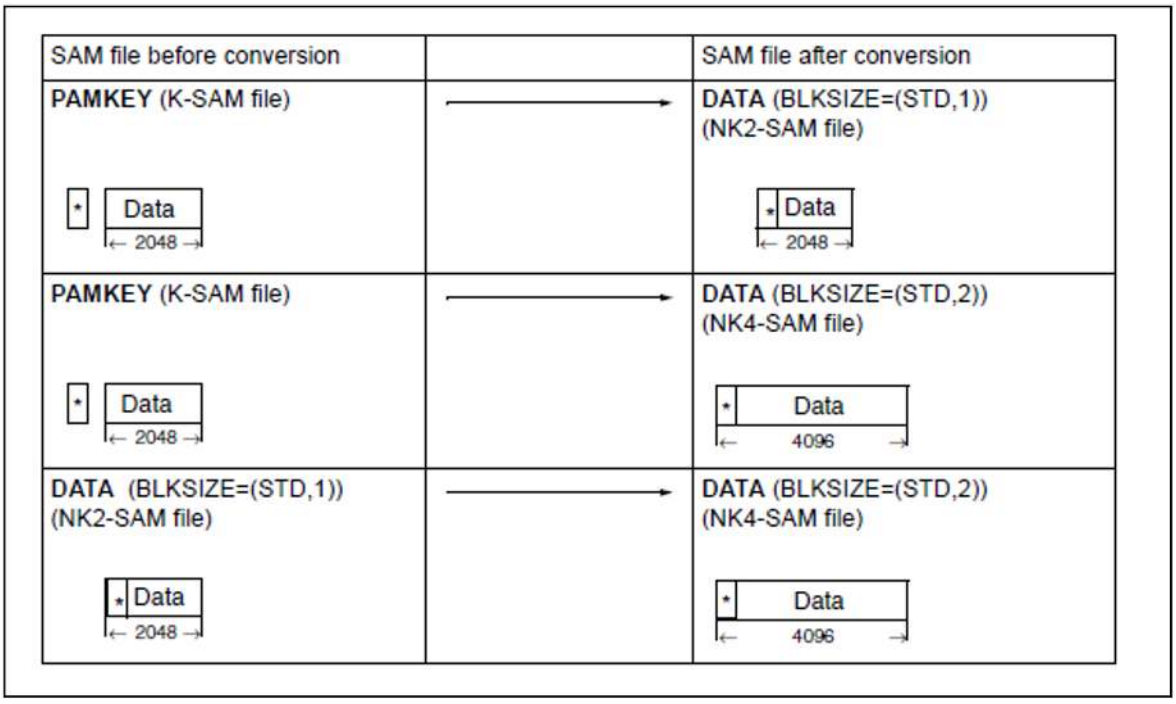
Source	Conversion options	Target
PAMKEY	------>	DATA
DATA	------>	PAMKEY
DATA	<------>	DATA

What block structure a SAM file has before and after file format conversion is illustrated for the following direction of conversion:

Source	Conversion options	Target
PAMKEY	------>	DATA

Key:

*	16 bytes block management information
BLKSIZE	logical block length; in the case of NK-SAM files the 16 bytes are deducted onlyonce per logical block length



PAM file

A PAM file that is to be converted with PAMCONV may have one of the following three block structures:

- **PAMKEY**: the file is a K-PAM file.
- **DATA**: the file is an NK2-PAM file if the blocking factor n is odd, or an NK4-PAM file if n is even.
- **NO**: the file is an NK2-PAM file if the blocking factor n is odd, or an NK4-PAM file if n is even. No block management information is stored.

The table below summarizes all conversion options for a PAM file:

Conversion options for a PAM file

Source	Conversion options	Target
PAMKEY	------>	NO
NO	------>	PAMKEY
NO	<------>	NO
DATA	<------>	DATA

PAM-DATA files cannot be converted into K format. If NONKEY-TO-KEY conversion is selected nevertheless, processing is rejected with message `PEA2212`.

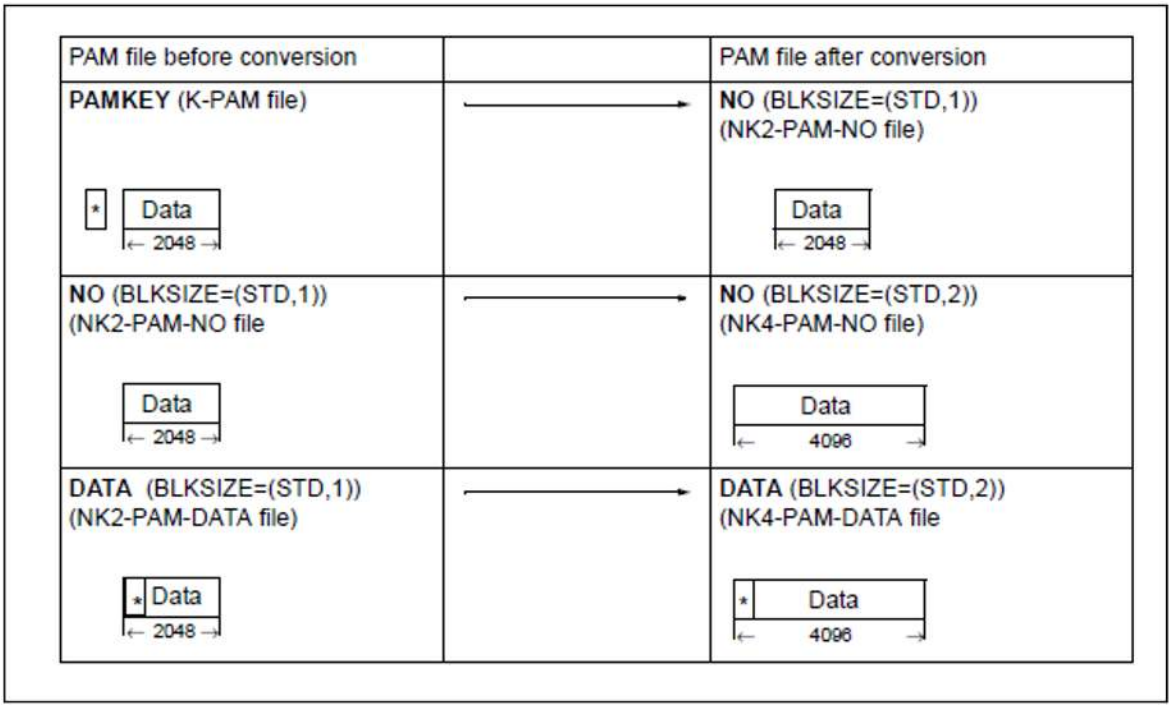
PLAM libraries are PAM files that do not use the PAM key. They can be converted.

What block structure a PAM file has before and after file format conversion is illustrated for the following directions of conversion:

Source	Conversion options	Target
PAMKEY	------>	NO
NO	------>	NO
DATA	------>	DATA

Key:

*	12 bytes block management information
BLKSIZE	Logical block length



Load module file

A load module is a specific type of PAM file. In the PAM file name (K-PAM or NK-PAM file), “PAM” is replaced by “load module”. Reference is therefore made to K, NK2 and NK4 load module files.

NK2 load module files have the file format with the block structure NO, i.e. no block control information is stored. The logical block length is (STD,1), i.e. 2048 bytes.
NK4 load module files have a logical block length of (STD,2), i.e. 4096 bytes. Any other block length specifications are rejected with an error message.

Conversion options for a load module file

Source	Conversion options	Target
PAMKEY	<------>	NO
NO	<------>	NO

Source files encrypted with a crypto password

Files encrypted with a crypto password are supported.

First a check is made to see whether the source file is encrypted. If it is encrypted,
PAMCONV also checks whether the crypto password involved is entered in the local task’s crypto password table. (The entry is made using `/ADD-CRYPTO-PASSWORD`.) If it is not, PAMCONV terminates conversion and issues an error message.

If the source file is encrypted and the password involved is entered in the crypt password table, the target file is assigned the same encryption.

The encryption is not adopted for the target file in the following cases:

- The target file is located on the home pubset under the TSOS user ID.
- The source file is a single file generation.
- The conversion is effected via an intermediate file (see “Conversion via an intermediate file” (Types of conversion)).

9.3 Conversion of file formats

- Types of conversion
- System environment requirements
- Specifying source and target files

9.3.1 Types of conversion

Standard conversion

It is assumed that the source and target files for a conversion are stored on public volumes.

This might be called the standard case. Enough space for both files must be available. If this is not the case, conversion must be effected via an intermediate file.

Conversion via an intermediate file

- General

  During file conversion, the target file requires about the same amount of disk space as the source file. A “self-contained” conversion without the need for additional space is not supported by PAMCONV. This means that enough disk space must be available for the target file. If this is not the case, conversion can be effected using an intermediate file on magnetic tape or private disk.

  When conversion takes place via an intermediate file, the FILE-DISPOSAL operand of the CONVERT-FILE statement (which specifies how the generated file is to be handled after conversion) is ignored.

  For this type of conversion, file convertibility is checked before an intermediate file is generated. This avoids a situation where the intermediate file would not be convertible into a target file.

- Two-step conversion using an intermediate file

  This type of conversion is performed explicitly by means of two CONVERT-FILE statements.

  The two statements may be issued in the same program run or in separate program runs.

  - Step 1
    Conversion of the source (disk) file on magnetic tape or private disk.
  - Step 2
    Conversion of the intermediate file from magnetic tape or private disk into the target file.

- One-step conversion using an intermediate file

  Two-step conversion as described above is combined into one operation here, which means that only one CONVERT-FILE statement is needed. This is achieved by entering DISK or TAPE for the DEVICE-FOR-TEMPFILE operand.

  After successful conversion from the source file to the intermediate file, the source file is deleted to obtain space for the target file.

  The intermediate file is assigned a name with the following format:

  `SYSTMP.<tsn>.PAMCONV.<ss>.<cpusec>`

  This intermediate file is erased after it has been successfully converted to the target file; otherwise further processing of the file is possible via this name.

  > **i** Internally, conversion is implemented in two steps as mentioned above. However, the user is not requested to make a new input until both conversion steps have been concluded.

Specification of the conversion options

The conversion variants are selected on the basis of the source and target file specifications and the entry in the DEVICE-FOR-TEMPFILE operand of the CONVERT-FILE statement:

- DEVICE-FOR-TEMPFILE=*NONE

  No intermediate file is stored on a private volume, unless this is specified via `/ADD-FILE-LINK`.

  The following options exist:

  - The source file is specified as a disk file *and* there is no catalog entry or `/ADD-FILE-LINK` for the target file, or the target file is specified as a disk file.

    Conversion from source file on magnetic disk to target file on magnetic disk. (Standard conversion)
  - The source file is specified as a disk file and the target file is specified as a tape file.

    Conversion from source file on magnetic disk to intermediate file on magnetic tape. (Two-step conversion via intermediate file on magnetic tape: Step 1.)
  - The source file is specified as a tape file *and* there is no catalog entry or `/ADD-FILE-LINK` for the target file, or the target file is specified as a disk file.

    Conversion from source file on magnetic tape (must be an intermediate file generated by PAMCONV) to target file on magnetic disk.
    (Two-step conversion via intermediate file on magnetic tape: Step 2.)

  Specifying both source file and target file as magnetic tape files is not permitted.

  > This implies that a file on magnetic tape must always be a PAMCONV intermediate file, otherwise conversion is rejected.

- DEVICE-FOR-TEMPFILE=*TAPE

  An intermediate file on magnetic tape is generated in each case. The source and/or target file must not simultaneously be specified as a tape file via `/ADD-FILE-LINK`.

  The following options exist:

  - The source file is specified as a disk file *and* there is no catalog entry or `/ADD-FILE-LINK` for the target file, or
  - the target file is specified as a disk file:

    Conversion from source file on magnetic disk to target file on magnetic disk. (Onestep conversion via intermediate file on magnetic tape.)

- DEVICE-FOR-TEMPFILE=*DISK

  An intermediate file on private disk is generated in each case. The source and/or target file must not simultaneously be specified as a tape file via `/ADD-FILE-LINK`. If the source and/or target file is defined by means of `/ADD-FILE-LINK` (for private disk), the volume specified in this command must not be identical to the private disk volume identified in the DEVICE-FOR-TEMPFILE operand.

  The following options exist:

  - The source file is specified as a disk file *and* there is no catalog entry or `/ADD-FILE-LINK` for the target file, or
  - the target file is specified as a disk file:

    Conversion from source file on magnetic disk to target file on magnetic disk. (Onestep conversion using an intermediate file on magnetic disk.)

Format of the intermediate file on magnetic tape

The source (disk) files may be SAM, ISAM or PAM files.

The ISAM access method is not defined for tapes. A standard format is therefore used for the intermediate file on tape. This is a SAM file containing the data records of the source file (ISAM: sorted by keys in ascending order). For general PAM files and load modules, a record consists of an 8-byte field with the user part of the PAM key and a 2048-byte field with the PAM block.

Additional file attributes are stored in a separate user header label (UHL).

Such an intermediate file is merely intended as a temporary file for conversion purposes. Magnetic tapes with standard labels must be used.

- Two-step conversion via intermediate file on tape

  If several files are to be converted at the same time, `/ADD-FILE-LINK` with SUPPORT=TAPE(FILE-SEQUENCE =...) must be issued; otherwise the intermediate file on tape will be overwritten.

- File attributes

  An intermediate file on magnetic tape has the following attributes:

```
FCBTYPE = SAM
BLKSIZE = (STD,16)
RECFORM = VARIABLE
LABEL   = STD
BLKCTRL = PAMKEY
```

  Otherwise the default values from the FCB macro apply.

  As can be seen from the following diagram, an intermediate file on tape is always a K-SAM file, regardless of the key format or FCB type of the source and target files.

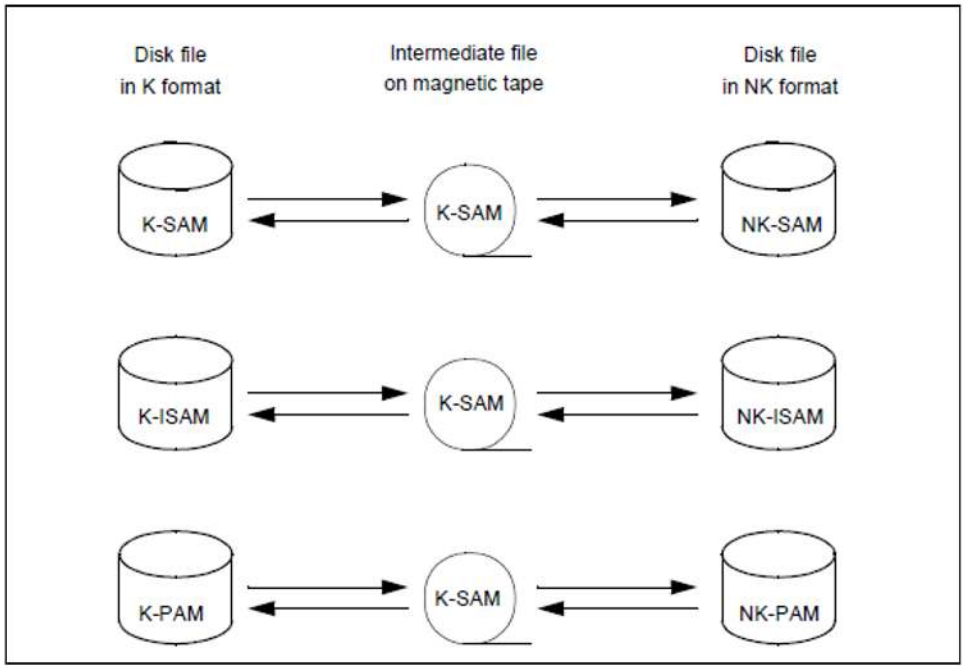


Figure 13: Intermediate file on magnetic tape

- User labels

  For the registration of the source file attributes, the user header label UHL1 with the following format is used.

Positions	Contents	Function
1..4	UHL1	user header label identification
5..12	PAMELA-I	ID for ISAM intermediate file
5..12	PAMELA-S	ID for SAM intermediate file
5..12	PAMELA-P	ID for PAM intermediate file
13..66	CL54	Name of source file
67	AL1(b)	BLKSIZE=(STD,b)
68	X'02'	RECFORM=VARIABLE
68	X'04'	RECFORM=FIXED
69..70	AL2(r)	RECSIZE=r
71..72	AL2(p)	KEYPOS=p
73	AL1(k)	KEYLEN=k
74	X'00'	VALPROP=MIN
74	X'01'	VALPROP=MAX
75	AL1(f1)	LOGLEN=f1
76	AL1(f2)	VALLEN=f2
77	X'80'	DUPEKY=YES
77	X'00'	DUPEKY=NO
78	XL1	Printer control character (IDCEDMS)
79	CL1	Library type ('P': PLAM)
80	XL1	File indicator (X'40': FGG)

Format of the intermediate file on private disk

In contrast to an intermediate file on tape, an intermediate file on private disk has no special format. It is always a copy of the source/target file, depending on the conversion direction. The intermediate file is always generated as an NK file. This makes it independent of the private disk’s key format.

> In the case of SAM files, the RECSIZE value of the copy (intermediate file) may differ from that of the NK source file. If the RECSIZE value of the NK source file is zero, the maximum possible value is used for the intermediate file.

The following diagram shows in which format the intermediate file is created on private disk and from which file the copy is made (depending on the conversion direction).

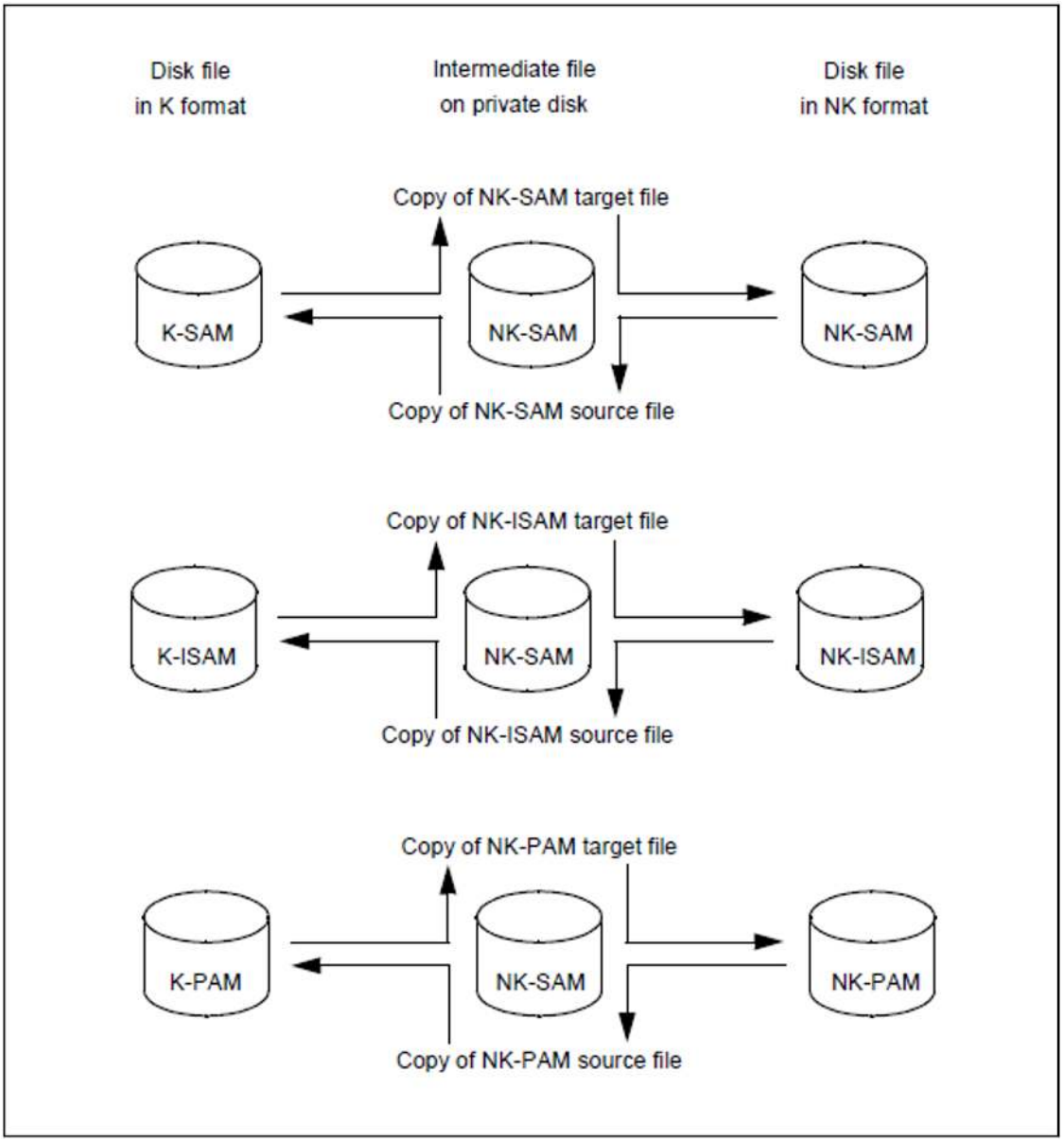


Figure 14: Intermediate file on private disk

9.3.2 System environment requirements

Tape peripherals

Magnetic tapes which are to store intermediate files must be equipped with standard labels.

Disk peripherals

If a system contains pure NK disk peripherals (without K disks), creation of K disk files is not possible. As a consequence, NONKEY-TO-KEY conversion is impossible in such systems.

If, generally speaking, both the source and the target file are to be situated within one pubset, care must be taken to provide enough space so that both files can be accommodated during conversion. The target file may need more space than the source file. If this is not possible, conversion should be effected via an intermediate file.

ACS (Alias Catalog Service)

Alias names may be used when specifying the source and target files for conversion. These names must be specified in precisely the form in which they are entered in the alias catalog. Otherwise it is impossible to establish the connection to the correct file name.

Diagnostic documentation

If problems occur in a PAMCONV run (unexpected messages, dump, ...), the following documents are required for diagnostic purposes:

- SYSLST log
  The log is designed to show the factors affecting the PAMCONV run (e.g. inputs made immediately before the error occurred).
- Input file in which the error occurred. The developers may then be able to find the cause of the problem which may not lead to an abort until considerably later.

9.3.3 Specifying source and target files

The source file and the corresponding target file can be specified in one of three ways:

Specifying fully qualified file names

Source and target files can be selected by specifying fully qualified file names. A single file generation can also be specified in this way.

In this case, the file attributes of the source file are transferred internally from the catalog entry.

The file attributes of the source file likewise apply to the target file.

The following file attributes are registered (see `/ADD-FILE-LINK`):

- access method
- record format
- record length
- data block length
- key length in ISAM files
- key position in ISAM files
- multiple keys in ISAM files
- length of logical markers in ISAM files
- length of value markers in ISAM files
- character set

Specifying selection criteria

The source file is specified in the form of a partial qualification with or without additional selection criteria (operand SELECT=*BY-ATTRIBUTES(...) of the CONVERT-FILE statement with CREATION-DATE, LAST-ACCESS-DATE, SIZE, FILE-STRUCTURE, BLKSIZE, BLKCTRL).

This information is used to determine all files to be converted (see `/SHOW-FILE-ATTRIBUTES` in the manual “Commands” [1 (Related publications)]).

The names of the target files may include wildcards (see `/SHOW-FILE-ATTRIBUTES`).

The remaining attributes of the target file are the same as for the source file.

> If LMR module libraries were included in the specification of selection criteria for PAM files, the module libraries could be destroyed.
> To avoid this, a check is performed prior to conversion of a PAM file. LMR module libraries are then recognized and not converted. LMR module libraries can be converted using the ISP-LMS (/EXEC $LMS).
>
> PAM files which are load modules are recognized as such and converted to the appropriate load module file format.

Specifying a reference via a link name

The reference is established via the link name to a previous `/ADD-FILE-LINK` specifying the file.

The link name is specified in the *LINK(LINK-NAME=...) operand of the CONVERT-FILE statement.

This method can also be used to select a single file generation.

The link name is mandatory if the target file is to have specific attributes which cannot be taken from the catalog entry of the source file. These include in particular:

- SPACE definitions
- index/data separation (for K-ISAM files only)
- PAD factor (normally the default value is used)
- RETPD specification
- WROUT=NO (WROUT=YES is standard)
- WRCHK=YES (WRCHK=NO is standard)

9.4 Reblocking

Only files with an even blocking factor can be stored on NK4 disks. PAMCONV therefore offers the reblocking function, which allows files with an uneven blocking factor to be stored on NK4 disks. It is implemented in the TO-FILE-BLKSIZE operand of the CONVERT-FILE, MODIFY-CONVERT-FILE-DEFAULTS and SHOW-CONVERT-FILE-DEFAULTS statements.

9.4.1 Explicit reblocking

The user initiates reblocking in the TO-FILE-BLKSIZE operand. This defines the logical block size of the target file. If the block size of the source file is uneven and the user explicitly specifies the block size of the target file, an error message is issued when the file is opened on an NK4 disk.

9.4.2 Implicit reblocking

If TO-FILE-BLKSIZE=*STD/*NK4 is specified, reblocking is carried out by PAMCONV.

- Increasing the blocking factor implicitly with TO-FILE-BLKSIZE=*STD

  The blocking factor of the target pubset applies; if necessary, PAMCONV increases the blocking factor implicitly. The block size remains unchanged for NK2 pubsets. In the case of NK4 pubsets, the blocking factor of the target file is increased if the blocking factor of the source file is uneven.
- Increasing the blocking factor implicitly with TO-FILE-BLKSIZE=*NK4

  If the blocking factor of the source file is uneven, the blocking factor of the target file is increased, regardless of the target pubset blocking factor.

PAMCONV does not decrease the blocking factor implicitly.

9.4.3 Reblocking PAM-DATA files without changing the file format

Another function of PAMCONV is reblocking NK-PAM-DATA files without changing the file format, i.e. both the source and the target file have the attribute BLKCTRL=DATA. Reblocking without conversion is supported by the specification DIRECTION=*TO-NONKEY and by the TO-FILE-BLKSIZE operand.

The **blocking factor** of a PAM-DATA file is increased only if the block size of the target file (TO-FILE-BLKSIZE) can be divided by the block size of the source file without leaving a remainder. Modulo(target-BLKSIZE / source-BLKSIZE)=0
If the remainder is not zero, reblocking is aborted with an error message.

When increasing the blocking factor of PAM-DATA files, only the block control field of the first block in each logical block is written to the target file. All other control fields in the same logical block are filled to 12 bytes with X'00'.

The **blocking factor** of PAM-DATA files is decreased only if the block size of the source file can be divided by the block size of the target file without leaving a remainder.
Modulo(source-BLKSIZE / target-BLKSIZE)=0
If the remainder is not zero, reblocking is aborted with an error message.

When the blocking factor is decreased, a check is made as to whether the control fields contain X'00'. If not, reblocking is aborted with an error message.
The blocking factor is decreased only if the blocking factor of the file has already been increased.

If the source PAM-DATA file has the block size BLKSIZE=(STD,1), PAMCONV performs implicit reblocking. Since this changes the file structure, a warning is issued.

Restrictions

- PAMCONV cannot convert NK2-PAM-DATA files with a BLKSIZE greater than (STD,8).
- The “reblocking” function for PAM-DATA files is executed for the TO-NONKEY conversion direction only. If NONKEY-TO-KEY is specified, conversion is aborted with message `PEA2212`, because PAM-DATA files cannot be converted to KEY.

9.4.4 Problems when decreasing the blocking factor

Decreasing the blocking factor can cause the record length (RECSIZE) of the source file to exceed the block size of the target file.

- Fixed record length (RECFORM=F)

  PAMCONV checks the record length of the source file and compares it with the record length of the target file, calculated from the block size of the target file. If the record length of the source file is greater than that of the target file, processing is aborted with a message.

- Variable or undefined record length (RECFORM=V/U)

  PAMCONV assumes that the record length of the target file is not exceeded and starts processing. If the record length of the target file is nevertheless exceeded, DMS informs PAMCONV and processing is aborted with a message.

Increasing the blocking factor does not cause any problems relating to the record length of the target file.

9.5 Controlling conversion and reblocking

The CONVERT-FILE statement is used to control the conversion and reblocking of files. Other PAMCONV statements are used to set or query the user-defined PAMCONV environment.

The following subsections describe special points relating to the conversion of ISAM and PAM file structures and to the conversion procedure.

9.5.1 Special points relating to conversion

Special points relating to the conversion of ISAM files

The TO-FILE-BLKCTRL operand of the CONVERT-FILE or MODIFY-CONVERT-FILE-DEFAULTS statement can be used to define the file format for ISAM files if the conversion direction TO-NONKEY is selected.
This particular case is subject to certain restrictions with regard to compatibility between the logical block size (TO-FILE-BLKSIZE) and the file format (TO-FILE-BLKCTRL) of an ISAM target file.

The assignment TO-FILE-BLKCTRL=*STD means that the block control information is set according to the target pubset. With NK2 pubsets, the file format is DATA2K; with NK4 pubsets, it is DATA4K.
If TO-FILE-BLKCTRL=*NK4 is specified, the block control information is assigned the value DATA4K.

TO-FILE-BLKSIZE defines the logical block size of the target file:

- TO-FILE-BLKSIZE=*STD:
  the logical block size is set according to the target pubset.
- TO-FILE-BLKSIZE=*NK4:
  the logical block size is always even.
- TO-FILE-BLKSIZE=<integer 1..16>:
  the target file is generated with a logical block size equal to the value specified here.

The following table shows the compatibility between the logical block size TO-FILE-BLKSIZE and the file format TO-FILE-BLKCTRL of an ISAM target file.

	TO-FILE-BLKSIZE																	
TO-FILE-BLKCTRL	STD	NK4	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
STD	X [1)]	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
NK4		X		X		X		X		X		X		X		X		X

[1)] In this table X means that both specifications are supported; otherwise, an error message is used.

Special points relating to the conversion of PAM files (not load modules)

In order to ascertain whether a PAM file with a PAM key actually uses this key, the user part of the PAM key is checked.

If the user part contains 8 X'00' bytes in each PAM block, the file is considered to be convertible.

Conversion to NK format consists in dropping the PAM key information (which is not in use); the BLKCTRL indicator field (see “Defining block control information during conversion” below) is assigned the value NO.

If the user part does not contain X'00', the file is assumed to use the PAM key. Such a file is classified as inconvertible, and conversion is aborted.

PAM files whose PAM key user part contains X'01' or X'80' in byte 1 but otherwise zero are an exception. With such files it is assumed that the PAM key is not used but that the user part was erroneously supplied with the above-mentioned values as a result of a DMS error. Such files are therefore classified as convertible to NONKEY format.
A message pointing out this exception is displayed during conversion.

If a K-PAM file contains gaps, these are replaced by “zero blocks” (2048 X'00' bytes) in the NK-PAM file during conversion.

Conversion of file generations

The file generations making up a file generation group can be converted by issuing a separate CONVERT-FILE statement for each file generation.

Fully automatic conversion of a file generation group with all its file generations in one pass is not possible; any such attempt is rejected with an error message.

Defining block control information during conversion

When converting from K to NK format, the block control information is set for the target file, depending on the relevant access method:

If the file name is specified by means of a link name and the BLKCTRL parameter is used, the BLKCTRL specification must be correct.

Converting an NK file into NK format

For compatibility reasons, the conversion of a file into NK format when the source file is already in NK format continues to be supported. But the function only copies the source file to the target file.

Transfer of file protection attributes after conversion

Following successful conversion, the file protection attributes of a source file can be transferred to the target file. To do this, the PROTECTION operand must be specified with the *SAME value in the CONVERT-FILE statement. If the user does not specify this value, the target file will be created without the file protection and file security features of the source file.

File protection attributes are transferred in accordance with the specification PROTECTION=*SAME in `/COPY-FILE` (see the “Commands” manual [1 (Related publications)]).
In order to be able to convert a protected file, before the PAMCONV run the user must specify all access authorizations (e.g. issuing of passwords) with the corresponding commands. The conversion algorithm itself is then executed in the usual way, without change. At the end of a successful conversion procedure, the protection attributes of the source file are transferred to the target file.

- Transfer of file protection attributes according to user ID

  The list below is based on direct conversion (without intermediate medium) from disk to disk.

- Conversion within any user ID

  The following protection attributes are transferred:

Protection attribute	Description
ACCESS	Standard access control; specifies whether write access (implicit read access) is permitted for the file, or read access only.
BACKUP-CLASS	specifies the frequency of automatic file saving with the ARCHIVE or HSMS backup system.
NUM-OF-BACKUP-VERS	specifies whether the file is part of the version backup and, if yes, also the maximum number of file versions to be saved in the version backup archive.
BASIC-ACL	Basic access control list; access control for the file is implemented via a BASIC-ACL entry. The read, write and execute access rights can be distributed among various user groups.
DESTROY-BY-DELETE	Files that are no longer required are overwritten with X'00'. This increases data protection.
ENCRYPTION	Password encryption
GUARDS	Access control via GUARDS; GUARDS is a functional unit of the SECOS software product.
LARGE	Extent of automatic file saving with the ARCHIVE or HSMS backup system.
MIGRATE	Files are migrated to another storage level if they have not been accessed for a certain length of time.
OPNBACK	specifies whether database files can also be saved with ARCHIVE even when they are open.
RETENTION-PERIOD	defines a protective deadline up until which only read access to the file is allowed, i.e. it must not be modified or deleted.
USER-ACCESS	controls access to the file via other user IDs.

  Passwords cannot be transferred as part of conversion within any user ID. This is only possible during conversion under the system administration ID (TSOS).

- Conversion under the system support ID (TSOS)

  Read (READ-PASSWORD), write (WRITE-PASSWORD) and execute passwords (EXEC-PASSWORD) are transferred, as are all protection attributes described above.

- Conversion of a source file from another user ID to the user's own ID

  The file saving attributes LARGE, BACKUP, NUM-OF-BACKUP-VERS, MIGRATE and OPNBACK and the file security attributes DESTROY-BY-DELETE, RETENTION-PERIOD, READ-PASSWORD, WRITE-PASSWORD, EXEC-PASSWORD and ENCRYPTION are transferred.
  BASIC-ACL or GUARDS entries in the source file are not transferred to the target file. These entries are assigned default values in the target file, as is the file protection attribute ACCESS. If an existing target file already has protection entries, they will be reset before the source file entries are transferred.

- Conversion of a source file from the user's own ID to another user ID

  The protection attributes of the source file are not transferred to the target file, even if the user is authorized to create the target file.

- Restrictions applying to the transfer of file protection attributes when converting via an intermediate medium

  The transfer of file protection attributes is only supported within "one-step conversion using an intermediate file". The file protection attributes are not transferred in the event of "two-step conversion using an intermediate file". If PROTECTION=*SAME is specified in the CONVERT-FILE statement, it is ignored.

  - One-step conversion using an intermediate medium

    The protection attributes of the source file are transferred because neither the PAMCONV run nor the current conversion statement CONVERT-FILE are interrupted. The following processing steps are executed:

    - conversion from a public disk to an intermediate medium
    - release of the source file's storage space
    - internal transfer from the intermediate medium to the public disk
    - setting of protection attributes
    - deletion of the source file

- Two-step conversion using an intermediate medium

  With this method of conversion both the PAMCONV run and the conversion statement may be interrupted. The file protection attributes cannot be transferred when the intermediate file is output on magnetic tape or disk, for the following reasons:

  - The second conversion step (from magnetic tape or private disk to the target file) could take place at an arbitrary later time, on an arbitrary system. This may result in incompatibilities, because the transfer of the NK file from the intermediate medium to a public data medium does not have to be executed with PAMCONV. Other transfer routines do not receive any information about the transfer of file protection attributes.
  - The user label of the magnetic tape on which the intermediate file is to be stored must be retained, for compatibility reasons; since further file characteristics cannot be included due to lack of space, it is impossible to transfer the protection attributes.

- Procedure with the target file after conversion

  The procedure with the target file after conversion is governed by the FILE-DISPOSAL operand in the CONVERT-FILE statement. The file protection attributes are transferred as shown below when the following operand values are specified:

FILE-DISPOSAL =	Meaning for transfer of file protection attributes
KEEP	Default setting. Source and target files are retained. The file protection attributes are transferred without problems.
RENAME	The source file is deleted after conversion. The protection attributes are transferred before deletion. If an error occurs during transfer, the source file is not deleted and the process is aborted with an error. The file protection attributes are therefore retained.
REPLACE	The source file is deleted after conversion and the target file is recataloged in accordance with the source file. The protection attributes are transferred before deletion. If an error occurs during transfer, the source file is not deleted and the process is aborted with an error. The file protection attributes are therefore retained.
INPLACE	After conversion, the source file is overwritten with the target file. The file protection attributes are transferred.

9.5.2 Further notes on conversion

To facilitate decision-making, PAMCONV offers the CLASSIFY-FILE statement, which checks files for their convertibility.

9.6 Statements

PAMCONV reads one statement at a time and executes it immediately. If link names are specified for a conversion, the associated TFT entries are not deleted following conversion, so that the link names can be used again.

9.6.1 Overview of PAMCONV statements

Statement	Function
CHANGE-TO-SYSTEM-MODE	Switch to system mode
CLASSIFY-FILE	Classify files according to their convertibility
CONVERT-FILE	Convert ISAM/SAM/PAM files or load module files
END	Terminate the PAMCONV program
MODIFY-CONVERT-FILE-DEFAULTS	Change the current default values for the CONVERT-FILE statement
MODIFY-LOGGING-OPTIONS	Change the current logging options
SHOW-CONVERT-FILE-DEFAULTS	List the current default values for the CONVERT-FILE statement
SHOW-LOGGING-OPTIONS	List the current logging options

Further permissible statements

In addition, the SDF standard statements such as SET-JOB-STEP can be used. They are of general importance in connection with SDF. They are described in the “SDF Dialog Interface” manual [20 (Related publications)].

9.6.2 Description of the statements

- CHANGE-TO-SYSTEM-MODE - Switch to system mode
- CLASSIFY-FILE - Classify files according to their convertibility
- CONVERT-FILE - Convert files
- END - Terminate PAMCONV
- MODIFY-CONVERT-FILE-DEFAULTS - Set default values for CONVERT-FILE statement
- MODIFY-LOGGING-OPTIONS - Set logging values
- SHOW-CONVERT-FILE-DEFAULTS - List current default values for CONVERT-FILE statement
- SHOW-LOGGING-OPTIONS - List specified logging options

9.6.2.1 CHANGE-TO-SYSTEM-MODE - Switch to system mode

The CHANGE-TO-SYSTEM-MODE statement causes a switchover to BS2000 system mode. BS2000 commands can subsequently be entered. Provided PAMCONV was not unloaded (e.g. by `/START-EXECUTABLE-PROGRAM` or `/LOAD-EXECUTABLE-PROGRAM`), the PAMCONV run can then be continued via `/RESUME-PROGRAM`.

Format

CHANGE-TO-SYSTEM-MODE

This statement has no operands.

Alternatively the SYSTEM or SYS statement can be entered to switch to system mode. These statements cannot be further abbreviated.

9.6.2.2 CLASSIFY-FILE - Classify files according to their convertibility

On each selected file, information as to its convertibility and any incompatibilities is requested.

> The default values set via the MODIFY-CONVERT-FILE-DEFAULTS statement are not taken into account here.The results of the CLASSIFY-FILE statement indicate the convertibility of a source file. For similar information on the target file, see the section “Reblocking”.

Format

CLASSIFY-FILE

```
DIRECTION = *TO-NONKEY / *NONKEY-TO-KEY
,FROM-FILE = *LINK(...) / *ALL / <partial-filename 2..53 with-wild(79)> / <filename 1..54>
    *LINK(...)
        |   LINK-NAME = <filename 1..8 without-gen>
,SELECT = *ALL / *BY-ATTRIBUTES(...)
    BY-ATTRIBUTES(...)
        |   CREATION-DATE = *ANY / <date> / *TODAY / *YESTERDAY / *INTERVAL(...)
        |       *INTERVAL(...)
        |           |   FROM = 0000-01-01 / <date> / *YESTERDAY
        |           |   ,TO = *TODAY / <date> / *YESTERDAY
        |   ,LAST-ACCESS-DATE = *ANY / <date> / *TODAY / *YESTERDAY / *INTERVAL(...)
        |       *INTERVAL(...)
        |           |   FROM = 0000-01-01 / <date> / *YESTERDAY
        |           |   ,TO = *TODAY / <date> / *YESTERDAY
        |   ,SIZE = *ANY / <integer 0..16777215> / *INTERVAL(...)
        |       *INTERVAL(...)
        |           |   FROM = 0 / <integer 0..16777215>
        |           |   ,TO = 16777215 / <integer 0..16777215>
        |   ,FILE-STRUCTURE = *ANY / list-poss(3): *SAM / *ISAM / *PAM
        |   ,BLKSIZE = *ANY / <integer 1..16>
        |   ,BLKCTRL = *ANY / *PAMKEY / *NO / *DATA / *DATA2K / *DATA4K
```

Operands

DIRECTION = *TO-NONKEY / *NONKEY-TO-KEY
Designates the user-defined direction of file conversion. This must be specified here because it influences the classification type.

DIRECTION = *TO-NONKEY
Files are to be converted to NK format.

DIRECTION = *NONKEY-TO-KEY
Files are to be converted from NK to K format.

FROM-FILE = <filename 1..54> / *LINK(...) / <partial-name 2..79 with-wild> / *ALL
Defines the files to be checked.

FROM-FILE = <filename 1..54>
Designates the fully qualified file name. Specification of a file generation is possible.

FROM-FILE = *LINK(...)
Identifies the files via a link name.

LINK-NAME = <filename 1..8 without-gen>
Specifies the link name.

FROM-FILE = <partial-filename 2..53 with-wild(79)>
Identifies the partially qualified file name with wildcard syntax.

FROM-FILE = *ALL
All files of the user ID are to be checked.

SELECT = *ALL / *BY-ATTRIBUTES(...)
Specifies whether the files to be classified are selected via specific selection criteria in addition to the partially qualified file name.

SELECT = *ALL
No additional selection criteria are specified for the source files.

SELECT = *BY-ATTRIBUTES(...)
Defines the selection criteria for the files to be classified.

CREATION-DATE = *ANY / *INTERVAL(...) / <date> / *TODAY / *YESTERDAY
Designates the creation date as a selection criterion.

CREATION-DATE = *ANY
The creation date is not used as a selection criterion. All files are taken into account for selection.

CREATION-DATE = *INTERVAL(...)
Files with a creation date within the specified interval are selected. The range limits are defined by the FROM and TO operands.

FROM = 0000-01-01 / *YESTERDAY / <date>
Files with a creation date equal to or later than the specified limit are selected.

FROM = 0000-01-01
The lower limit is the earliest possible date.

FROM = *YESTERDAY
The lower limit is yesterday's date. Files with a creation date yesterday's date are selected.

FROM = <date>
The lower limit is the specified date. Files with a creation date the specified value are selected.

TO = *TODAY / *YESTERDAY / <date>
Files with a creation date equal to or earlier than the specified limit are selected.

TO = *TODAY
The upper limit is the current date. Files with a creation date the current date are selected.

TO = *YESTERDAY
The upper limit is yesterday's date. Files with a creation date yesterday's date are selected.

TO = <date>
The upper limit is the specified date. Files with a creation date the specified value are selected.

LAST-ACCESS-DATE = *ANY / *INTERVAL(...) / <date> / *TODAY / *YESTERDAY
Designates the date of the last file access as a selection criterion.

For the meaning of ANY, INTERVAL(...), <date>, TODAY and YESTERDAY see the CREATION-DATE operand.

SIZE = *ANY / <integer 0..16777215> / *INTERVAL(...)
Designates the file size as a selection criterion.

SIZE = *ANY
The file size is not used as a selection criterion.

SIZE = <integer 0..16777215>
Files with a size equal to the specified value are selected.

SIZE = *INTERVAL(...)
Files with a size within the specified range are selected. The range limits are defined by the FROM and TO operands.

FROM = 0 / <integer 0..16777215>
Files with a size the specified limit are selected.

FROM = 0
The lower limit is the absolute minimum.

FROM = <integer 0..16777215>
The lower limit is the specified size.

TO = 16777215 / <integer 0..16777215>
Files with a size the specified limit are selected.

TO = 16777215
The upper limit is the absolute maximum.

TO = <integer 0..16777215>
The upper limit is the specified size.

FILE-STRUCTURE = *ANY / list-poss(3): *SAM / *ISAM / *PAM
Designates the access method as a selection criterion.

FILE-STRUCTURE = *ANY
The access method is not used as a selection criterion.

FILE-STRUCTURE = *SAM
Files with the SAM access method are selected.

FILE-STRUCTURE = *ISAM
Files with the ISAM access method are selected.

FILE-STRUCTURE = *PAM
Files with the PAM access method are selected.

BLKSIZE = *ANY / <integer 1..16>
Designates the block size as a selection criterion.

BLKSIZE = *ANY
The block size is not used as a selection criterion.

BLKSIZE = <integer 1..16>
Files with a block size equal to the specified value are selected.

BLKCTRL = *ANY / *PAMKEY / *NO / *DATA / *DATA2K / *DATA4K
Designates the block control attribute as a selection criterion.

BLKCTRL = *ANY
The block control attribute is not used as a selection criterion.

BLKCTRL = *PAMKEY
Files with the block control attribute PAMKEY are selected.

BLKCTRL = *NO
Files with the block control attribute NO are selected.

BLKCTRL = *DATA
Files with the block control attribute DATA are selected.

BLKCTRL = *DATA2K
Files with the block control attribute DATA2K are selected.

BLKCTRL = *DATA4K
Files with the block control attribute DATA4K are selected.

The output destination is determined by the OUTPUT operand of the MODIFY-LOGGING-OPTIONS statement.

The results of the check are output to SYSLST in the following form (line length up to 132 characters):

```
%//CLASSIFY-FILE   FROM-FILE=>from-file<
% CLASSIFY-FILE    >from-file<    DIRECTION = >direction<
%
% FILENAME                                          !  PAM-  !FCB- !CONVER-!
INCOMPATIBILITIES  !
%                                                  !  PAGES !TYPE !TIBLE
!                     !
%
-------------------------------------------------------------------------------------------------
```

The results of the check are output to SYSOUT in the following form (line length up to 80 characters):

```
% CLASSIFY-FILE   FROM-FILE=>from-file<
% CLASSIFY-FILE   >from-file<    DIRECTION = >direction<
%
% FILENAME                                                    ! CONVER- !
%                                                             ! TIBLE   !
% ------------------------------------------------------------------------
% :CATID:$USERID.>filename 1< ..............................  ! >yesno< !
% :CATID:$USERID.>filename 2< ..............................  ! >yesno< !
% :CATID:$USERID.>filename 3< ..............................  ! >yesno< !
%
%
% :CATID:$USERID.>filename n< ..............................  ! >yesno< !
% ------------------------------------------------------------------------
%            >n<       FILE(S) LISTED
```

Meanings of the output fields:

`>from-file<`	File names specified in the CLASSIFY-FILE statement.
`>direction<`	Conversion direction specified in the CLASSIFY-FILE statement
`>filename<`	Name of the file checked
`>size<`	Size of the file checked
`>fcb<`	FCB type of the file checked
`>n<`	Total number of files checked
`>yesno<`	File convertibility information. Possible values: YES File is convertible. NO File is not convertible. NK2 File is only convertible on NK2 pubsets with standard blksize (PLAM libraries). NK4 ... File is only convertible on NK2 pubsets (e.g. SAM files with BLKSIZE=RECSIZE and standard blksize).

>reason<	Reason for incompatibility. Possible values: NONE No incompatibilities, file is convertible. RECSIZE EXCEEDS MAXIMUM The record length exceeds the maximum value determined by BLKSIZE. The file can be converted by increasing the blocking factor. RECSIZE EXCEEDS MAX(NK2) The record length would exceed the maximum value determined by BLKSIZE on NK2 pubsets. It is possible to convert to NK4 pubsets. PLAM(NK4) NO CONVERT PLAM libraries with BLKSIZE > 2 are not converted with PAMCONV. FILE ALREADY IN KEY-FORMAT The file is already in K format. The file in K format cannot be converted to K format. FILESIZE INCREASES The formation of overflow blocks increases the size of the target file, the file is convertible. KEYPOS IN OVERFLOW-BLOCK The ISAM key would have to be stored in an overflow block. The file can be converted by increasing the blocking factor. LMR-LIBRARY File is an LMR library, i.e. it is not convertible. PAMKEY IS USED The file makes use of the user part of the PAM key, file is not convertible PAMKEY CONTAINS SPECIAL FLAG The file contains X'01' or X'80' only in byte 1 of the user part of the PAM key. The file is convertible.

9.6.2.3 CONVERT-FILE - Convert files

This statement serves to convert files from a format in which the PAM key is used for data representation into a format in which the PAM key is not used, or vice versa.

The CONVERT-FILE statement is also used for reblocking (see the section “Reblocking”).

The CONVERT-FILE statement offers three options:

- **Conversion:** changes the file format, i.e. converts a file from K format to NK format or vice versa.
- **Reblocking:** changes the logical block size. No change to the file format.
- **Conversion and reblocking**

The default values for the CONVERT-FILE statement are set using the MODIFY-CONVERT-FILE-DEFAULTS statement.

The default values indicated by underscoring take effect only if no other values have been specified.

Format

```
CONVERT-FILE

 DIRECTION = *TO-NONKEY / *NONKEY-TO-KEY

 ,FROM-FILE = <filename 1..54> / *LINK(...) / <partial-filename 2..53 with-wild(79)> / *ALL

     *LINK(...)

         |    LINK-NAME = <filename 1..8 without-gen>

 ,SELECT = *ALL / *BY-ATTRIBUTES(...)

     BY-ATTRIBUTES(...)

         |    CREATION-DATE = *ANY / <date> / *TODAY / *YESTERDAY / *INTERVAL(...)

         |         *INTERVAL(...)

         |              |      FROM = 0000-01-01 / <date> / *YESTERDAY

         |              |      ,TO = *TODAY / <date> / *YESTERDAY

         |    ,LAST-ACCESS-DATE = *ANY / <date> / *TODAY / *YESTERDAY / *INTERVAL(...)

         |         *INTERVAL(...)

         |              |      FROM = 0000-01-01 / <date> / *YESTERDAY

         |              |      ,TO = *TODAY / <date> / *YESTERDAY

         |    ,SIZE = *ANY / <integer 0..16777215> / *INTERVAL(...)

         |         *INTERVAL(...)

         |              |      FROM = 0 / <integer 0..16777215>

         |              |      ,TO = 16777215 / <integer 0..16777215>

         |    ,FILE-STRUCTURE = *ANY / list-poss(3): *SAM / *ISAM / *PAM

         |    ,BLKSIZE = *ANY / <integer 1..16>

         |    ,BLKCTRL = *ANY / *PAMKEY / *NO / *DATA / *DATA2K / *DATA4K
```

```
,TO-FILE = <filename 1..54> / *LINK(...) / <partial-filename 2..53 with-wild(79)>
      *LINK(...)
           |     LINK-NAME = <filename 1..8 without-gen>
,TO-FILE-BLKSIZE = *STD / *NK4 / <integer 1..16>
,TO-FILE-BLKCTRL = *STD / *NK4
,REPLACE-OLD-FILES = *NO / *YES / *DIALOG
,FILE-DISPOSAL = *KEEP / *REPLACE / *INPLACE / *RENAME
,PROTECTION = *STD / *SAME
,DEVICE-FOR-TEMPFILE = *NONE / *TAPE(...) / *DISK(...)
      *TAPE(...)
           |     VOLUME = list-poss(100): <alphanum-name 1..6>
           |     ,DEVICE-TYPE = <device>
      *DISK(...)
           |     VOLUME = list-poss(100): <alphanum-name 1..6>
           |     ,DEVICE-TYPE = <device>
```

Operands

DIRECTION = *TO-NONKEY / *NONKEY-TO-KEY
Designates the direction in which file conversion is to take place.

DIRECTION = *TO-NONKEY
This file is to be converted into NK format. The source file may be in K format or NK format.

DIRECTION = *NONKEY-TO-KEY
This file is to be converted from NK to K format.

FROM-FILE = <filename 1..54> / *LINK(...) / <partial-filename 2..53 with-wild(79)> / *ALL
Identifies the files to be converted.

FROM-FILE = <filename 1..54>
Designates the file to be converted. Specification of a file generation is possible.

FROM-FILE = *LINK(...)
The file to be converted was specified by a previous `/ADD-FILE-LINK`; the link name given there must be identical to the one specified here.

LINK-NAME = <filename 1..8 without-gen>
Specifies the link name.

FROM-FILE = <partial-filename 2..53 with-wild(79)>
Means that all files corresponding to the specified wildcard syntax and to any additional selection criteria are to be converted.

FROM-FILE = *ALL
Means that all files corresponding to the specified selection criteria are to be converted.

SELECT = *ALL / *BY-ATTRIBUTES(...)
Defines whether the files to be converted are to be selected via specific criteria in addition to the partially qualified file name.

SELECT = *ALL
No additional selection criteria are specified for the source files.

SELECT = *BY-ATTRIBUTES(...)
Defines the selection criteria for the files to be converted.

CREATION-DATE = *ANY / *INTERVAL(...) / <date> / *TODAY / *YESTERDAY
Designates the creation date as a selection criterion.

CREATION-DATE = *ANY
The creation date is not used as a selection criterion. All files are taken into account for selection.

CREATION-DATE = INTERVAL(...)
Files with a creation date within the specified interval are selected. The range limits are defined by the FROM and TO operands.

FROM = 0000-01-01 / *YESTERDAY / <date>
Files with a creation date equal to or later than the specified limit are selected.

FROM = 0000-01-01
The lower limit is the earliest possible date.

FROM = *YESTERDAY
The lower limit is yesterday’s date. Files with a creation date yesterday’s date are selected.

FROM = <date>
The lower limit is the specified date. Files with a creation date the specified value are selected.

TO = TODAY / *YESTERDAY / <date>
Files with a creation date equal to or earlier than the specified limit are selected.

TO = *TODAY
The upper limit is the current date. Files with a creation date the current date are selected.

TO = *YESTERDAY
The upper limit is yesterday’s date. Files with a creation date yesterday’s date are selected.

TO = <date>
The upper limit is the specified date. Files with a creation date the specified value are selected.

LAST-ACCESS-DATE = *ANY / *INTERVAL(...) / <date> / *TODAY / *YESTERDAY
Designates the date of the last file access as a selection criterion.

For the meaning of ANY, INTERVAL(...), <date>, TODAY and YESTERDAY see the CREATION-DATE operand.

SIZE = *ANY / <integer 0..16777215> / *INTERVAL(...)
Designates the file size as a selection criterion.

SIZE = *ANY
The file size is not used as a selection criterion.

SIZE = <integer 0..16777215>
Files with a size equal to the specified value are selected.

SIZE = *INTERVAL(...)
Files with a size within the specified range are selected. The range limits are defined by the FROM and TO operands.

FROM = 0 / <integer 0..16777215>
Files with a size the specified limit are selected.

FROM = 0
The lower limit is the absolute minimum.

FROM = <integer 0..16777215>
The lower limit is the specified size.

TO = 16777215 / <integer 0..16777215>
Files with a size the specified limit are selected.

TO = 16777215
The upper limit is the absolute maximum.

TO = <integer 0..16777215>
The upper limit is the specified size.

FILE-STRUCTURE = *ANY / list-poss(3): *SAM / *ISAM / *PAM
Designates the access method as a selection criterion.

FILE-STRUCTURE = *ANY
The access method is not used as a selection criterion.

FILE-STRUCTURE = *SAM
Files with the SAM access method are selected.

FILE-STRUCTURE = *ISAM
Files with the ISAM access method are selected.

FILE-STRUCTURE = *PAM
Files with the PAM access method are selected.

BLKSIZE = *ANY / <integer 1..16>
Designates the block size as a selection criterion.

BLKSIZE = *ANY
The block size is not used as a selection criterion.

BLKSIZE = <integer 1..16>
Files with a block size equal to the specified value are selected.

BLKCTRL = *ANY / *PAMKEY / *NO / *DATA / *DATA2K / *DATA4K
Designates the block control attribute as a selection criterion.

BLKCTRL = *ANY
The block control attribute is not used as a selection criterion.

BLKCTRL = *PAMKEY
Files with the block control attribute PAMKEY are selected.

BLKCTRL = *NO
Files with the block control attribute NO are selected.

BLKCTRL = *DATA
Files with the block control attribute DATA are selected.

BLKCTRL = *DATA2K
Files with the block control attribute DATA2K are selected.

BLKCTRL = *DATA4K
Files with the block control attribute DATA4K are selected.

TO-FILE = <filename 1..54> / *LINK(...) / <partial-filename 2..53 with-wild(79)>
Identifies the files to be created by conversion.

TO-FILE = <filename 1..54>
Designates the file to be created by conversion. Specification of a file generation is possible.

TO-FILE = *LINK(...)
The file to be created was specified by a previous `/ADD-FILE-LINK`; the link name given there must be identical to the one specified here.

LINK-NAME = <filename 1..8 without-gen>
Specifies the link name.

TO-FILE = <partial-filename 2..53 with-wild(79)>
Specifies the files to be created by conversion in the form of a partial qualification with wildcard syntax.

TO-FILE-BLKSIZE = *STD / *NK4 / <integer 1..16>
Specifies the logical block size of the target file.

TO-FILE-BLKSIZE = *STD
The logical block size of the target file is not defined by the user. PAMCONV uses the values defined for the target pubset and, if necessary, increases the blocking factor. The blocking factor is increased internally by a maximum of 1.

TO-FILE-BLKSIZE = *NK4
The logical block size of the target file is controlled in such a way that it is always even, i.e. the target file can be stored on an NK4 pubset. The blocking factor is increased internally by a maximum of 1.

TO-FILE-BLKSIZE = <integer 1..16>
The target file is generated with a block size equal to the specified value, provided that this specification is compatible with the other conditions governing reblocking (see the section “Reblocking”).

TO-FILE-BLKCTRL = *STD / *NK4
Specifies the block control information of the target file. This operand is relevant only for the conversion direction TO-NONKEY and for ISAM files.

TO-FILE-BLKCTRL = *STD
The block control information is set in accordance with the target pubset. With NK2 pubsets, the data format is DATA2K; with NK4 pubsets, it is DATA4K.

TO-FILE-BLKCTRL = *NK4
The block control information is assigned the value DATA4K.

REPLACE-OLD-FILES = *NO / *YES / *DIALOG
Indicates whether any files existing under this name are to be overwritten.

REPLACE-OLD-FILES = *NO
Existing files must not be overwritten; file conversion is aborted in this case.

REPLACE-OLD-FILES = *YES
Existing files are overwritten unless additional protection (password, ACCESS=*READ,..) has been provided.

REPLACE-OLD-FILES = *DIALOG
The system issues a query as to the desired procedure for existing files with the same name. Possible in interactive mode only.

FILE-DISPOSAL = *KEEP / *RENAME / *REPLACE / *INPLACE
Determines what happens to the source and target files after conversion.

FILE-DISPOSAL = *KEEP

The target files are to be created with the names specified for them in the conversion statement. The target files exist in addition to the source files.

FILE-DISPOSAL = *RENAME
The target files are to be created with the names specified for them in the conversion statement. After successful conversion, the source files are to be deleted.

FILE-DISPOSAL = *REPLACE
The target files are to be created with the names specified for them in the conversion statement. After successful conversion, the source files are to be deleted and the target files are to receive the names of the source files. In effect, the source file is replaced by the target file.

FILE-DISPOSAL = *INPLACE
The target files are to be created with the names specified for them in the conversion statement. After successful conversion, an attempt is to be made to overwrite the source file with the target file and to assign the source file name to the target file. This has the effect that the target file also physically takes up approximately the same space as the source file. In effect, the source file is replaced by the target file.

PROTECTION = *STD / *SAME
Specifies whether the file protection attributes of the source file are to be transferred to the target file.

PROTECTION = *STD
The file protection attributes are not transferred.

PROTECTION = *SAME
The file protection attributes are transferred to the target file. For more details see “Transfer of file protection attributes after conversion” (Special points relating to conversion).

DEVICE-FOR-TEMPFILE = *NONE / *TAPE(...) / *DISK(...)
Designates the storage medium which is to accommodate the intermediate (temporary) file created.

DEVICE-FOR-TEMPFILE = *NONE
No intermediate file is to be stored on a private volume.

DEVICE-FOR-TEMPFILE = *TAPE(...)
The intermediate file is to be stored on magnetic tape.

VOLUME = list-poss(100): <alphanum-name 1..6>
Designates the VSN(s) of the tape(s) to be used as storage medium, if more than one VSN are stated in a list.

DEVICE-TYPE = <device>
Designates the tape device type to be used.

DEVICE-FOR-TEMPFILE = *DISK(...)
The intermediate file is to be stored on private disk.

VOLUME = list-poss(100): <alphanum-name 1..6>
Designates the VSN(s) of the private disk(s) to be used as storage medium, if more than one VSN are stated in a list.

DEVICE-TYPE = <device>
Designates the disk device type to be used.

9.6.2.4 END - Terminate PAMCONV

The END statement terminates the PAMCONV program.

Format

END

This statement has no operands.

9.6.2.5 MODIFY-CONVERT-FILE-DEFAULTS - Set default values for CONVERT-FILE statement

This statement sets the default values for the CONVERT-FILE statement. The specified defaults then apply to the current program run until the next MODIFY-CONVERT-FILE-DEFAULTS statement is issued.

If the statement is entered without operands, the existing default values remain valid.

The currently applicable values may be queried using the SHOW-CONVERT-FILE-DEFAULTS statement.

> The default values set here apply **only** to the CONVERT-FILE statement. The CLASSIFY-FILE statement is not influenced by this statement.

Format

```
MODIFY-CONVERT-FILE-DEFAULTS

DIRECTION = *UNCHANGED / *TO-NONKEY / *NONKEY-TO-KEY
,SELECT = *UNCHANGED / *ALL / *BY-ATTRIBUTES(...)
    *BY-ATTRIBUTES(...)
        |  CREATION-DATE = *ANY / <date> / *TODAY / *YESTERDAY / *INTERVAL(...)
        |        *INTERVAL(...)
        |              |     FROM = 0000-01-01 / <date> / *YESTERDAY
        |              |     ,TO = *TODAY / <date> / *YESTERDAY
        |  ,LAST-ACCESS-DATE = *ANY / <date> / *TODAY / *YESTERDAY / *INTERVAL(...)
        |        *INTERVAL(...)
        |              |     FROM = 0000-01-01 / <date> / *YESTERDAY
        |              |     ,TO = *TODAY / <date> / *YESTERDAY
        |  ,SIZE = *ANY / <integer 0..16777215> / *INTERVAL(...)
        |        *INTERVAL(...)
        |              |     FROM = 0 / <integer 0..16777215>
        |              |     ,TO = 16777215 / <integer 0..16777215>
        |  ,FILE-STRUCTURE = *ANY / list-poss(3): *SAM / *ISAM / *PAM
        |  ,BLKSIZE = *ANY / <integer 1..16>
        |  ,BLKCTRL = *ANY / *PAMKEY / *NO / *DATA / *DATA2K / *DATA4K
,TO-FILE-BLKSIZE = *UNCHANGED / *STD / *NK4 / <integer 1..16>
,TO-FILE-BLKCTRL = *UNCHANGED / *STD / *NK4
```

```
,REPLACE-OLD-FILES = *UNCHANGED / *NO / *YES / *DIALOG

,FILE-DISPOSAL = *UNCHANGED / *KEEP / *REPLACE / *INPLACE / *RENAME

,PROTECTION = *UNCHANGED / *STD / *SAME

,DEVICE-FOR-TEMPFILE = *UNCHANGED / *NONE / *TAPE(...) / *DISK(...)

    *TAPE(...)

        |    VOLUME = list-poss(100): <alphanum-name 1..6>

        |    ,DEVICE-TYPE = <text 1..20>

    *DISK(...)

        |    VOLUME = list-poss(100): <alphanum-name 1..6>

        |    ,DEVICE-TYPE = <text 1..20>
```

Operands

DIRECTION = *UNCHANGED / *ONKEY / *NONKEY-TO-KEY
Designates the desired default value for the direction of file conversion.

DIRECTION = *UNCHANGED
The current default value for DIRECTION is not changed.

DIRECTION = *TO-NONKEY
This file is to be converted into NK format.

DIRECTION = *NONKEY-TO-KEY
This file is to be converted from NK to K format.

SELECT = *UNCHANGED / *ALL / *BY-ATTRIBUTES(...)
Specifies whether the files to be converted are selected via specific selection criteria in addition to the partially qualified file name.

SELECT = *UNCHANGED
The current default value for SELECT is not changed.

SELECT = *ALL
No selection criteria are set for the source files.

SELECT = *BY-ATTRIBUTES(...)
Defines the selection criteria for the files to be converted and which are to apply as default values for the CONVERT-FILE statement.

CREATION-DATE = *ANY / *INTERVAL(...) / <date> / *TODAY / *YESTERDAY
Designates the creation date as a selection criterion.

CREATION-DATE = *ANY
The creation date is not used as a selection criterion. All files are taken into account for selection.

CREATION-DATE = *INTERVAL(...)
Files with a creation date within the specified interval are selected. The range limits are defined by the FROM and TO operands.

FROM = 0000-01-01 / *YESTERDAY / <date>
Files with a creation date equal to or later than the specified limit are selected.

FROM = 0000-01-01
The lower limit is the earliest possible date.

FROM = *YESTERDAY
The lower limit is yesterday's date. Files with a creation date yesterday's date are selected.

FROM = <date>
The lower limit is the specified date. Files with a creation date the specified value are selected.

TO = *TODAY / *YESTERDAY / <date>
Files with a creation date equal to or earlier than the specified limit are selected.

TO = *TODAY
The upper limit is the current date. Files with a creation date the current date are selected.

TO = *YESTERDAY
The upper limit is yesterday's date. Files with a creation date yesterday's date are selected.

TO = <date>
The upper limit is the specified date. Files with a creation date the specified value are selected.

LAST-ACCESS-DATE = *ANY / *INTERVAL(...) / <date> / *TODAY / *YESTERDAY
Designates the date of the last file access as a selection criterion.

For the meaning of ANY, INTERVAL(...), <date>, TODAY and YESTERDAY see the CREATION-DATE operand.

SIZE = *ANY / <integer 0..16777215> / *INTERVAL(...)
Designates the file size as a selection criterion.

SIZE = *ANY
The file size is not used as a selection criterion.

SIZE = <integer 0..16777215>
Files with a size equal to the specified value are selected.

SIZE = *INTERVAL(...)
Files with a size within the specified range are selected. The range limits are defined by the FROM and TO operands.

FROM = 0 / <integer 0..16777215>
Files with a size the specified limit are selected.

FROM = 0
The lower limit is the absolute minimum.

FROM = <integer 0..16777215>
The lower limit is the specified size.

TO = 16777215 / <integer 0..16777215>
Files with a size the specified limit are selected.

TO = 16777215
The upper limit is the absolute maximum.

TO = <integer 0..16777215>
The upper limit is the specified size.

FILE-STRUCTURE = *ANY / list-poss(3): *SAM / *ISAM / *PAM
Designates the access method as a selection criterion.

FILE-STRUCTURE = *ANY
The access method is not used as a selection criterion.

FILE-STRUCTURE = *SAM
Files with the SAM access method are selected.

FILE-STRUCTURE = *ISAM
Files with the ISAM access method are selected.

FILE-STRUCTURE = *PAM
Files with the PAM access method are selected.

BLKSIZE = *ANY / <integer 1..16>
Designates the block size as a selection criterion.

BLKSIZE = *ANY
The block size is not used as a selection criterion.

BLKSIZE = <integer 1..16>
Files with a block size equal to the specified value are selected.

BLKCTRL = *ANY / *PAMKEY / *NO / *DATA / *DATA2K / *DATA4K
Designates the block control attribute as a selection criterion.

BLKCTRL = *ANY
The block control attribute is not used as a selection criterion.

BLKCTRL = *PAMKEY
Files with the block control attribute PAMKEY are selected.

BLKCTRL = *NO
Files with the block control attribute NO are selected.

BLKCTRL = *DATA
Files with the block control attribute DATA are selected.

BLKCTRL = *DATA2K
Files with the block control attribute DATA2K are selected.

BLKCTRL = *DATA4K
Files with the block control attribute DATA4K are selected.

TO-FILE-BLKSIZE = *UNCHANGED / *STD / *NK4 / <integer 1..16>
Specifies the logical block size of the target file.

TO-FILE-BLKSIZE = *UNCHANGED
The default value currently valid for TO-FILE-BLKSIZE is to remain unchanged.

TO-FILE-BLKSIZE = *STD
The logical block size of the target file is not defined by the user. PAMCONV uses the values defined for the target pubset and, if necessary, increases the blocking factor. The blocking factor is increased internally by a maximum of 1.

TO-FILE-BLKSIZE = *NK4
The logical block size of the target file is controlled in such a way that it is always even, i.e. the target file can be stored on an NK4 pubset. The blocking factor is increased internally by a maximum of 1.

TO-FILE-BLKSIZE = <integer 1..16>
The target file is generated with a block size equal to the specified value, provided that this specification is compatible with the other conditions governing reblocking (see the section “Reblocking”).

TO-FILE-BLKCTRL = *UNCHANGED / *STD / *NK4
Specifies the block control information of the target file. This operand is relevant only for the conversion direction TO-NONKEY and for ISAM files.

TO-FILE-BLKCTRL = *UNCHANGED
The default value currently valid for TO-FILE-BLKCTRL is to remain unchanged.

TO-FILE-BLKCTRL = *STD
The block control information is set in accordance with the target pubset. With NK2 pubsets, the data format is DATA2K; with NK4 pubsets, it is DATA4K.

TO-FILE-BLKCTRL = *NK4
The block control information is assigned the value DATA4K.

REPLACE-OLD-FILES = *UNCHANGED / *NO / *YES / *DIALOG
Indicates whether any files existing under this name are to be overwritten.

REPLACE-OLD-FILES = *UNCHANGED
The current default value for REPLACE-OLD-FILES is not changed.

REPLACE-OLD-FILES = *NO
Existing files must not be overwritten; file conversion is to be aborted in this case (default value).

REPLACE-OLD-FILES = *YES
Existing files are overwritten unless additional protection (password, ACCESS=*READ,..) has been provided.

REPLACE-OLD-FILES = *DIALOG
The system issues a query as to the desired procedure for existing files with the same name. Possible in interactive mode only.

FILE-DISPOSAL = *UNCHANGED / *KEEP / *RENAME / *REPLACE / *INPLACE
Determines what happens to the source and target file after conversion.

FILE-DISPOSAL = *UNCHANGED
The current default value for FILE-DISPOSAL is not changed.

FILE-DISPOSAL = *KEEP
The target files are to be created with the names specified for them in the conversion statement. The target files exist in addition to the source files.

FILE-DISPOSAL = *RENAME
The target files are to be created with the names specified for them in the conversion statement. After successful conversion, the source files are to be deleted.

FILE-DISPOSAL = *REPLACE
The target files are to be created with the names specified for them in the conversion statement. After successful conversion, the source files are to be deleted and the target files are to receive the names of the source files. In effect, the source file is replaced by the target file.

FILE-DISPOSAL = *INPLACE
The target files are to be created with the names specified for them in the conversion statement. After successful conversion, an attempt is to be made to overwrite the source file with the target file and to assign the source file name to the target file. This has the effect that the target file also physically takes up approximately the same space as the source file. In effect, the source file is replaced by the target file.

PROTECTION = *UNCHANGED / *STD / *SAME
Specifies whether the file protection attributes of the source file are to be transferred to the target file.

PROTECTION = *UNCHANGED
The currently valid default value for PROTECTION is to remain unchanged.

PROTECTION = *STD
The file protection attributes are not transferred.

PROTECTION = *SAME
The file protection attributes are transferred to the target file. For more details see “Transfer of file protection attributes after conversion” (Special points relating to conversion).

DEVICE-FOR-TEMPFILE = *UNCHANGED / *NONE / *TAPE(...) / *DISK(...)
Designates the storage medium which is to accommodate the intermediate (temporary) file created.

DEVICE-FOR-TEMPFILE = *UNCHANGED
The current default value for DEVICE-FOR-TEMPFILE is not changed.

DEVICE-FOR-TEMPFILE = *NONE
No intermediate file is to be stored on a private volume.

DEVICE-FOR-TEMPFILE = *TAPE(...)
The intermediate file is to be stored on magnetic tape.

> **VOLUME = list-poss(100): <alphanum-name 1..6>**
> Designates the VSN(s) of the tape(s) to be used as storage medium, if more than one VSN are stated in a list.
>
> **DEVICE-TYPE = <text 1..20>**
> Designates the device type to be used.

DEVICE-FOR-TEMPFILE = *DISK(...)
The intermediate file is to be stored on private disk.

VOLUME = list-poss(100): <alphanum-name 1..6>
Designates the VSN(s) of the private disk(s) to be used as storage medium, if more than one VSN are stated in a list.

DEVICE-TYPE = <text 1..20>
Designates the device type to be used.

The volume specified in the DEVICE-FOR-TEMPFILE operand is only reserved/ released during execution of the CONVERT-FILE statement.

9.6.2.6 MODIFY-LOGGING-OPTIONS - Set logging values

This statement defines logging values for PAMCONV which have global validity for all functions of the program. If the statement is entered without operands, the existing values remain valid. The currently applicable values can be queried using the SHOW-LOGGING-OPTIONS statement.

Format

MODIFY-LOGGING-OPTIONS
INFORMATION = *UNCHANGED / *MEDIUM / *MINIMUM / *MAXIMUM ,OUTPUT = *UNCHANGED / list-poss(2): *SYSOUT / *SYSLST

Operands

INFORMATION = *UNCHANGED / *MEDIUM / *MINIMUM / *MAXIMUM
Controls the scope of the log created by PAMCONV.

INFORMATION = *UNCHANGED
The current default value for INFORMATION is not changed.

INFORMATION = *MEDIUM
Statements are logged in the case of errors only. Positive acknowledgments (message class 5) are logged in addition to MINIMUM.

INFORMATION = *MINIMUM
Only error messages, completion messages and negative acknowledgments are logged (i.e. all message classes except class 5).

INFORMATION = *MAXIMUM
The log created consists of statements, positive and negative acknowledgments, error messages and completion messages.

OUTPUT = *UNCHANGED / list-poss(2): *SYSOUT / *SYSLST
Defines the output medium for the logs created by PAMCONV.

OUTPUT = *UNCHANGED
The current default value for OUTPUT is not changed.

OUTPUT = list-poss(2): *SYSOUT / *SYSLST
Either one output medium or a list of 2 output media (SYSOUT and SYSLST) may be entered.

If SYSOUT is specified, logs are written to the SYSOUT system file (in interactive mode this is by default the terminal). If SYSLST is specified, logs are written to the SYSLST system file. In the case of a list, the logs are output to both system files.

9.6.2.7 SHOW-CONVERT-FILE-DEFAULTS - List current default values for CONVERT-FILE statement

The SHOW-CONVERT-FILE-DEFAULTS statement lists the currently valid default values for the CONVERT-FILE statement.

Format

SHOW-CONVERT-FILE-DEFAULTS

This statement has no operands.

The output destination is determined by the OUTPUT operand of the MODIFY-LOGGING-OPTIONS statement. The specified values are listed in the following form:

```
%//SHOW-CONVERT-FILE-DEFAULTS
% CURRENT CONVERT-FILE DEFAULTS
%   DIRECTION                  : <wert>
%   SELECT                     : <wert>
%   TO-FILE-BLKSIZE            : <wert>
%   TO-FILE-BLKCTRL            : <wert>
%   REPLACE-OLD-FILES          : <wert>
%   FILE-DISPOSAL              : <wert>
%   PROTECTION                 : <wert>
%   DEVICE-FOR-TEMPFILE        : <wert>
```

<value> ... see MODIFY-CONVERT-FILE-DEFAULTS for possible values.

> The text segments `CURRENT CONVERT-FILE DEFAULTS`, ..., `DEVICE-FOR-TEMPFILE` above have been defined in the message file and are taken from there. If the text is to be changed, these segments can be redefined in the message file.

9.6.2.8 SHOW-LOGGING-OPTIONS - List specified logging options

A list of the currently valid values for logging is requested.

Format

SHOW-LOGGING-OPTIONS

This statement has no operands.

The output destination is determined by the OUTPUT operand of the MODIFY-LOGGING-OPTIONS statement.

The specified values are listed in the following form:

```
%//SHOW-LOGGING-OPTIONS
%CURRENT LOGGING OPTIONS
%  INFORMATION : <wert>
%  OUTPUT      : <wert>
```

<value> ... see MODIFY-LOGGING-OPTIONS for possible values.

> The text segments `CURRENT LOGGING OPTIONS`, `INFORMATION`, and `OUTPUT` above have been defined in the message file and are taken from there. If the text is to be changed, these segments can be redefined in the message file by system administration.

9.7 PAMCONV program execution

The program reads the control statements via SYSDTA. Messages are output via SYSOUT and/or SYSLST, depending on the logging options specified (see the MODIFY-LOGGING-OPTIONS and SHOW-LOGGING-OPTIONS statements).

Example

```
/START-EXECUTABLE-PROGRAM FROM-FILE=PAMCONV                                  1.
%  BLS0523 ELEMENT 'PAMCONV', VERSION '<version>', TYPE 'L' FROM LIBRARY
':1OSH:$TSOS.SYSLNK.PAMCONV.<version>' IN PROCESS
%  BLS0524 LLM 'PAMCONV', VERSION '<version>' OF '<date> <time>' LOADED
%  PEA7000 <time> PAMCONV VERSION <version> STARTED IN BS2000 <version>
%  PEA7001 PLEASE ENTER PAMCONV STATEMENTS
%//CONVERT-FILE FROM-FILE=DAT*,SELECT=BY-ATTRIBUTES(CREATION-DATE=
   INTERVAL(,YESTERDAY),FILE-STRUCTURE=SAM),TO-FILE=NK.DAT*                   2.
%  PEA5000 CONVERSION TO NON-KEY FORMAT COMPLETED. SOURCE FILE:
   ':N:$USER0001.DATEI1'; TARGET FILE: ':N:$USER0001.NK.DATEI1'
%  PEA5000 CONVERSION TO NON-KEY FORMAT COMPLETED. SOURCE FILE:
   ':N:$USER0001.DATEI2'; TARGET FILE: ':N:$USER0001.NK.DATEI2'
%  PEA5000 CONVERSION TO NON-KEY FORMAT COMPLETED. SOURCE FILE:
   ':N:$USER0001.DATEI3'; TARGET FILE: ':N:$USER0001.NK.DATEI3'
%  PEA5000 CONVERSION TO NON-KEY FORMAT COMPLETED. SOURCE FILE:
   ':N:$USER0001.DATEI4'; TARGET FILE: ':N:$USER0001.NK.DATEI4'               3.
%//CONVERT-FILE FROM-FILE=DAT*,TO-FILE=NK.DAT*                               4.
%  PEA5000 CONVERSION TO NON-KEY FORMAT COMPLETED. SOURCE FILE:
   ':N:$USER0001.DATEI5'; TARGET FILE: ':N:$USER0001.NK.DATEI5'               5.
%  PEA2103 'TO-FILE' ALREADY EXISTS. TARGET FILE: 'NK.DATEI1'
%  PEA2103 'TO-FILE' ALREADY EXISTS. TARGET FILE: 'NK.DATEI2'
%  PEA2103 'TO-FILE' ALREADY EXISTS. TARGET FILE: 'NK.DATEI3'
%  PEA2103 'TO-FILE' ALREADY EXISTS. TARGET FILE: 'NK.DATEI4'                 6.
%//MODIFY-CONVERT-FILE-DEFAULTS SELECT=BY-ATTRIBUTES(FILE-STRUCTURE=ISAM)     7.
%//SHOW-CONVERT-FILE-DEFAULTS                                                8.
% CURRENT   CONVERT-FILE   DEFAULTS
%   DIRECTION                  : TO-NONKEY
%   SELECT                     : BY-ATTRIBUTES( )
%      CREATION-DATE           : ANY
%      LAST-ACCESS-DATE        : ANY
%      SIZE                    : ANY
%      ACCESS-METHOD           : ISAM
%      BLKSIZE                 : ANY
%      BLKCTRL                 : ANY
%   TO-FILE-BLKSIZE            : STD
%   TO-FILE-BLKCTRL            : STD
%   REPLACE-OLD-FILES          : NO
%   FILE-DISPOSAL              : KEEP
%   PROTECTION                 : STD
%   DEVICE-FOR-TEMPFILE        : NONE                                          9.
%//CONVERT-FILE DIRECTION=NONKEY-TO-KEY,FROM-FILE=NK.DAT*,TO-FILE=K.DAT*     10.
%  PEA5001 CONVERSION FROM NON-KEY TO KEY FORMAT COMPLETED. SOURCE FILE:
   ':N:$USER0001.NK.DATEI5'; TARGET FILE: ':N:$USER0001.K.DATEI5'            11.
%//CLASSIFY-FILE FROM-FILE=*DAT*                                             12.
% CLASSIFY-FILE   *DAT*  DIRECTION = KEY-TO-NONKEY
%
% FILENAME                                                    ! CONVER- !
%                                                             ! TIBLE   !
% :N:$USER0001.DATEI1 ..................................      !   YES   !
% :N:$USER0001.DATEI2 ..................................      !   YES   !
```

```
% :N:$USER0001.DATEI3 ..................................         !   YES   !
% :N:$USER0001.DATEI4 ..................................         !   YES   !
% :N:$USER0001.DATEI5 ..................................         !   YES   !
% :N:$USER0001.K.DATEI5 ................................         !   YES   !
% :N:$USER0001.NK.DATEI1 ...............................         !   YES   !
% :N:$USER0001.NK.DATEI2 ...............................         !   YES   !
% :N:$USER0001.NK.DATEI3 ...............................         !   YES   !
% :N:$USER0001.NK.DATEI4 ...............................         !   YES   !
% :N:$USER0001.NK.DATEI5 ...............................         !   YES   !
%            11      FILE(S)LISTED                                      13.
%//END                                                                  14.
%  PEA7003 10:01:09/1.5323 PAMCONV TERMINATED ABNORMALLY                15.
```

1. The PAMCONV program is called.
2. The CONVERT-FILE statement is entered with partially qualified file names and selection criteria.
3. Acknowledgment of successful file conversion.
4. Input of the CONVERT-FILE statement with partially qualified file names without further selection criteria.
5. Acknowledgments of successful file conversions.
6. File exists already, conversion is rejected.
7. Definition of selection criteria for further CONVERT-FILE statements.
8. Input of the SHOW-CONVERT-FILE-DEFAULTS statement.
9. Output of the values requested via the SHOW-CONVERT-FILE-DEFAULTS statement. The values shown are the defaults for subsequent CONVERT-FILE statements.
10. Input of the CONVERT-FILE statement with partially qualified file names for NK to K conversion.
11. Acknowledgments of successful file conversions.
12. Input of the CLASSIFY-FILE statement with partially qualified file names.
13. Output of the values requested via the CLASSIFY-FILE statement. Classification of the input files by convertibility.
14. Input of the END statement.
15. The PAMCONV program is terminated abnormally, since an error has occurred in the PAMCONV run.

9.8 Error handling

Control statement errors in interactive mode

- Syntax errors: Incorrect control statements are rejected with appropriate error messages.
- Semantic errors: An error dialog is conducted with the user.

Control statement errors in batch mode

Syntax or/and semantic errors: As soon as an incorrect statement is detected, all statements up to the next STEP or END statement are skipped.

DMS error messages, error code

DMS error messages are output for the user (missing passwords, ACCESS=*READ, crypto password, etc.). The DMS error code reported is output with additional PAMCONV-specific information in certain cases.

Errors during conversion

If errors occur during processing of a CONVERT-FILE statement before the target file has been successfully generated and closed, the incomplete target file (or the intermediate file) is deleted and the source file is retained regardless of the explicitly or implicitly (default value) specified FILE-DISPOSAL operand of the MODIFY-CONVERT-FILE-DEFAULTS statement. Successful closure of the target file is acknowledged with a corresponding message. The following error situations are possible after successful closure of the target file (or intermediate file):

- The source file cannot be closed properly. Conversion has been successfully concluded, but the activities specified explicitly or implicitly (default value) in the FILE-DISPOSAL operand are not performed.
- Both the source and the target file have been successfully closed, and in accordance with the conversion specification of the FILE-DISPOSAL operand the source file is to be replaced with the target file (REPLACE or RENAME), but the source file cannot be deleted. The same procedure as above applies.
- The source file is to be replaced with the target file; the source file has been deleted, but the renaming operation in accordance with the conversion value REPLACE in the FILE-DISPOSAL operand cannot be performed. In this case the end result is the same as if RENAME had been specified in the FILE-DISPOSAL operand.

Subsequently the next statement is executed.

Inconsistent files

If the source file is an ISAM/SAM file or a load module, it is subjected to a strict check for formal consistency during conversion. Inconsistencies may be subdivided into two groups:

- The inconsistency is such that loss of data is to be expected during further conversion (e.g. in the case of defective index entries). Conversion is aborted with an appropriate message.
- There is a formal inconsistency, but loss of data can be excluded (e.g. reserved fields or free blocks do not contain binary zero). In this case a warning is output and conversion continues.

Subsequently the next statement is executed.

Incompatible files

Operation of NK formats is not fully compatible. Consequently, various incompatibilities result from conversion into NK format. See the section on PAM key elimination in the “Introductory Guide to DMS” [4 (Related publications)].

As soon as incompatibilities are sensed which prevent proper conversion, the conversion process is aborted with an appropriate message (hard incompatibility).

If incompatibilities are detected which reduce performance but do not essentially prevent conversion, a warning is issued and conversion continues (soft incompatibility).

Subsequently the next statement is executed.

9.9 PAMCONV messages

The messages of the PAMCONV utility routine have the message class PEA.

See also the section “Messages and their meaning” (Notational conventions).

10 PASSWORD Encryption of passwords

Version: PASSWORD V21.0A

This chapter describes the password encryption routine PASSWORD.

The PASSWORD routine encrypts passwords in a system without automatic password encryption, thus enabling files to be read that were created in a system with password encryption.

The PASSWORD program is a service routine for exporting files from one system to another (e.g. with the ARCHIVE routine or HSMS). PASSWORD is needed only if files that have been created and protected in a system with password encryption are to be processed in a system without password encryption. PASSWORD encrypts the specified passwords and enters them in the password table of the job.

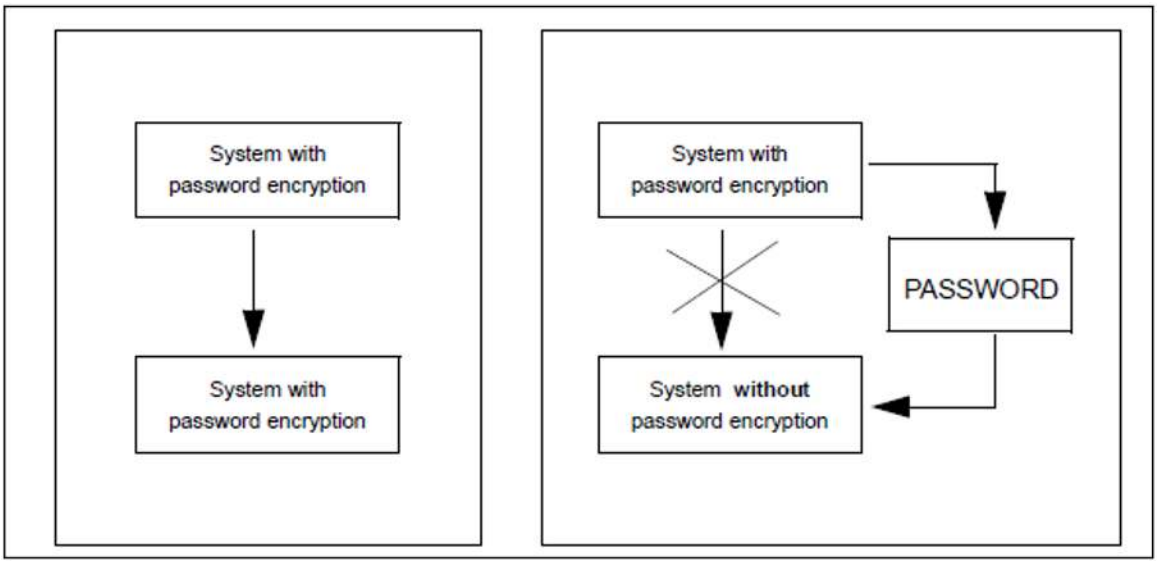


Figure 15: Password processing in systems with and without password encryption

10.1 Operation and execution

A file or LOGON password specified by a user is written into the catalog entry of the file or the user catalog, respectively. In systems operating with password encryption, the password entered by the user is encrypted according to an internal code and transferred in this form to the catalog entry or user catalog. The user must enter the password in its unencrypted form when calling the file or entering `/SET-LOGON-PARAMETERS`, since the encryption is performed internally by the system.
The purpose of this procedure is to prevent unauthorized persons from discovering passwords if parts of the catalog or user catalog happen to be included in the output when a dump is taken.

When files with encrypted passwords are imported into a system operating without password encryption, a user who specifies the unencrypted password will not be given access to the file because the password is contained in its encrypted form in the catalog. The function of PASSWORD is to convert the unencrypted passwords into encrypted passwords in order to allow the file to be accessed.

10.1.1 Passwords

PASSWORD processes file passwords and LOGON passwords.

File passwords protect files against

- unauthorized writing (write password, WRITE-PASSWORD operand of `/MODIFY-FILE-ATTRIBUTES`)
- unauthorized reading (read password, READ-PASSWORD operand of `/MODIFY-FILE-ATTRIBUTES`)
- unauthorized execution in the case of program and procedure files (execute password, EXEC-PASSWORD operand of `/MODIFY-FILE-ATTRIBUTES`).

The power of these file passwords is weighted with respect to access. Write passwords are the most powerful, read passwords the next most powerful, then execute passwords. A less powerful password does not have to be specified explicitly if a more powerful one has already been specified.

File passwords are defined for one or more files. A check is made for the presence of the correct passwords prior to execution of the system commands `/START-EXECUTABLE-PROGRAM`, `/LOAD-EXECUTABLE-PROGRAM`, `/MODIFY-FILE-ATTRIBUTES`, `/DELETE-FILE`, `/CREATE-FILE`, `/ADD-FILE-LINK`, `/COPY-FILE`, `/SHOW-FILE`, and also at file opening.

To avoid repeated specification of file passwords for system commands, the password can be written to the password table during task execution. This table is then searched for the appropriate password whenever a password-protected file is to be processed.

The **LOGON password** is a password that must be entered with `/SET-LOGON-PARAMETERS` in order to ensure that only authorized users may work under a particular user ID.

LOGON passwords contained in ENTER jobs are evaluated only if the ENTER job was called from a console. Otherwise all LOGON operands specified in the ENTER job, including the LOGON password, are ignored.

Format conventions for passwords

File passwords and *job variable* passwords must not exceed 4 bytes in length. They are represented in the following format:

C'x'	where x stands for 1 to 4 alphanumeric or special characters.
X'n'	where n stands for 1 to 8 hexadecimal digits.
d	where d stands for a decimal number of up to 8 digits (with or without a positive or negative sign) whose value is converted into a binary value.

LOGON passwords may be between 1 and 32 characters long. The passwords are represented in the following format:

C'x' where x stands for 1 to 32 alphanumeric or special characters

X'n' where n stands for 1 to 16 hexadecimal digits.

> PASSWORD still only process LOGON passwords that are between 1 and 8 characters long. Longer LOGON Passwords must be abbreviated to 8 characters beforehand. The software product SDF-P provides the predefined function HASH-STRING() function which can be used to carry out this conversion. (Example see "ENCRYPTJ - Encrypt specified LOGON password").For further details regarding “long passwords” see also `/MODIFY-USER-PROTECTION` in the “Commands” manual [1 (Related publications)]).

10.1.2 Program execution

The PASSWORD routine is started with `/START-PASSWORD`.

START- PASSWORD
VERSION = ***STD** / <product-version> **,MONJV** = ***NONE** / <filename 1..54 without-gen-vers> **,CPU-LIM**IT = ***JOB-RE**ST / <integer 1..32767 *seconds*>

Thereafter you can input PASSWORD statements. Each statement issued to PASSWORD is executed immediately. Any output is displayed on the terminal (SYSOUT).

Once a statement has been completely executed, a new PASSWORD statement may be entered or the routine may be terminated with the END statement.

10.2 Statements

- Overview of PASSWORD statements
- Description of the statements
  - CONVERT - Encrypt most powerful password of a file
  - ENCPASS - Encrypt file password and enter in PASSWORD table
  - ENCRYPTD - Encrypt specified file password
  - ENCRYPTJ - Encrypt specified LOGON password
  - END - Terminate PASSWORD
  - HELP - List operands
  - JVCONV - Encrypt password for job variable
  - MODE - Select encryption routine
  - PASSWORD - Encrypt specified file password

10.2.1 Overview of PASSWORD statements

Statements for the PASSWORD routine are entered via SYSDTA.

Statement	Brief description
CONVERT	The most powerful password for the specified file is entered in the password table in its encrypted form. `/MODIFY-FILE-ATTRIBUTES` containing the passwords is issued for the file.
ENCPASS	Encrypts the specified password and enters the result in the password table.
ENCPRYPTD	Encrypts the specified file password and writes the result to SYSOUT.
ENCRYPTJ	Encrypts the specified LOGON password and writes the result to SYSOUT.
END	Terminates the PASSWORD run.
HELP	Outputs a brief description of all statements or of the specified statement.
JVCONV	The most powerful password for the specified job variable is entered in the password table in its encrypted form. `/MODIFY-JV-ATTRIBUTES` containing the passwords is then issued for the job variable.
MODE / M	Determines the encryption algorithm.
PASSWORD	Encrypts the specified file password and enters both the password and its encrypted form in the password table.

10.2.2 Description of the statements

- CONVERT - Encrypt most powerful password of a file
- ENCPASS - Encrypt file password and enter in PASSWORD table
- ENCRYPTD - Encrypt specified file password
- ENCRYPTJ - Encrypt specified LOGON password
- END - Terminate PASSWORD
- HELP - List operands
- JVCONV - Encrypt password for job variable
- MODE - Select encryption routine
- PASSWORD - Encrypt specified file password

10.2.2.1 CONVERT - Encrypt most powerful password of a file

Up to three passwords of a given file can be specified with the CONVERT statement. The CONVERT statement encrypts the most powerful password with `/ADD-PASSWORD` and then issues `/MODIFY-FILE-ATTRIBUTES` containing all the specified passwords.

Once the command has been executed, subsequent access is not possible until the password has been entered in the password table (`/ADD-PASSWORD`).

Format

CONVERT
filename { ,WRPASS = fpassword / ,RDPASS = fpassword / ,EXPASS = fpassword }

Operands

filename
Fully qualified name of the file whose passwords are to be encrypted.

WRPASS = fpassword
Specifies the write password to be encrypted.

RDPASS = fpassword
Specifies the read password to be encrypted.

EXPASS = fpassword
Specifies the execute password to be encrypted.

> fpassword is a file password consisting of 1 to 4 bytes (see “Format conventions for passwords” (Passwords) for the format). At least one of the three passwords must be specified.

Examples

1. PASSWORD routine with specification of a password:

```
/start-password
ENTER COMMAND NOW :
*convert anne.3,rdpass=c'susi'
PASSWORD X'9A41632A'
CAT     ANNE.3,RDPASS=C'SUSI',STATE=U
*end
PASSWORD : NORMAL   END
/
```

2. The following excerpt shows which password is encrypted if more than one is specified:

```
/start-password
ENTER COMMAND NOW :
*convert pass.test,wrpass=c'susi',rdpass=c'anne',expass=c'otto'
PASSWORD X'9A41632A'
CAT     PASS.TEST,WRPASS=C'SUSI',RDPASS=C'ANNE',EXPASS=C'OTTO',STATE=U
*convert pass.test,expass=c'otto',rdpass=c'anne'
PASSWORD X'7A08FCC2'
CAT     PASS.TEST,EXPASS=C'OTTO',RDPASS=C'ANNE',STATE=U
*convert pass.test,expass=c'otto'
PASSWORD X'DE28062F'
CAT     PASS.TEST,EXPASS=C'OTTO',STATE=U
*end
PASSWORD : NORMAL   END
/
```

10.2.2.2 ENCPASS - Encrypt file password and enter in PASSWORD table

This statement encrypts the specified file password and enters the result in the password table of the task. The encrypted password is also output to SYSOUT.

Purpose of the statement

A protected file was taken from a system with password encryption and added to a system without password encryption (e.g. with ARCHIVE). The catalog entry of the file contains an encrypted password. The file can consequently only be accessed by someone who specifies the encrypted password.

ENCPASS enters the specified password in the password table in encrypted form, thus enabling the user to access the file protected by the password.

Format

ENCPASS / EP
fpassword

Operands

fpassword
File password consisting of 1 to 4 bytes (see “Format conventions for passwords” (Passwords)).

Example

```
/start-password
ENTER COMMAND NOW :
*encpass c'susi'
PASSWORD X'9A41632A'
*end
PASSWORD : NORMAL   END
/
```

10.2.2.3 ENCRYPTD - Encrypt specified file password

This statement encrypts the specified file password and outputs the result to SYSOUT.

Purpose of the statement

A protected file was taken from a system with password encryption and added to a system without password encryption (e.g. with ARCHIVE). The catalog entry of the file contains an encrypted password. The file can consequently only be accessed by someone who specifies the encrypted password.

The ENCRYPTD statement encrypts the original password. In contrast to the ENCPASS statement, the user only regains access to the file by entering the encrypted password, e.g. with ADD-PASSWORD.

Format

ENCRYPTD / ED
fpassword

Operands

fpassword
File password consisting of 1 to 4 bytes (see “Format conventions for passwords” (Passwords)).

Example

```
/start-password
ENTER COMMAND NOW :
*encryptd c'susi'
PASSWORD ENCRYPTED IS = X'9A41632A'
*end
PASSWORD : NORMAL   END
/
```

10.2.2.4 ENCRYPTJ - Encrypt specified LOGON password

This statement encrypts the specified LOGON password and outputs the result to SYSOUT.

Format

ENCRYPTJ / EJ
lpassword

Operands

lpassword

LOGON password consisting of 1 to 8 bytes (see “Format conventions for passwords” (Passwords) for the format).

> In BS2000, LOGON passwords of up to 32 characters can be declared. To be able to edit them with PASSWORD, they must be shortened to 8 characters at system level.
> The software product SDF-P provides the predefined function HASH-STRING() which enables this conversion.
>
> ```
> (TO-X-LIT(HASH-STRING('<long password>',8)))
> ```

Examples

1. Encryption of a password that is 4 characters long.

```
ENTER COMMAND NOW :
*encryptj c'susi'
PASSWORD ENCRYPTED IS = X'6B0537211705D615'
*end
PASSWORD : NORMAL   END
```

2. Encryption of a password that is 14 characters long.

```
/A=(TO-X-LIT(HASH-STRING('longpassword',8)))
/show-var a
A = X'523E146036CED784'
/start-password
ENTER COMMAND NOW :
*encryptj x'523E146036CED784'
PASSWORD ENCRYPTED IS = X'73FE57A922EA780D'
*end
PASSWORD : NORMAL   END
```

10.2.2.5 END - Terminate PASSWORD

The END statement terminates the PASSWORD run.

Format

END

This statement has no operands.

Normal termination of the routine produces the message:

```
PASSWORD : NORMAL END
```

10.2.2.6 HELP - List operands

This statement lists either all control statements for PASSWORD with their permitted operands, or all additional operands of a specified statement.

Format

HELP / H
statement

Operands

statement
Full name of a PASSWORD statement, i.e.: CONVERT, JVCONV, ENCRYPTD, ENCRYPTJ, ENCPASS, MODE, PASSWORD or END

Note, however, that the short form of each statement, as shown in the statement formats, is equally valid as input.

10.2.2.7 JVCONV - Encrypt password for job variable

The JVCONV statement is used to encrypt the password(s) for a specified job variable. At least one password must be specified in addition to the job variable name. The JVCONV statement encrypts the most powerful password and then issues `/MODIFY-JV-ATTRIBUTES` with all specified passwords.

Once the command has been executed, subsequent access is not possible until the password has been entered in the password table (`/ADD-PASSWORD`).

Format

JVCONV
jvname { ,WRPASS = jpassword / ,RDPASS = jpassword }

Operands

jvname
Job variable jpassword consisting of 1 to 4 bytes (see “Format conventions for passwords” (Passwords)). At least one of the two passwords must be specified.

WRPASS=jpassword
Specifies the write password to be encrypted.

RDPASS=jpassword
Specifies the read password to be encrypted.

Example

```
/start-password
ENTER COMMAND NOW :
*jvconv hugo,wrpass=c'susi'
PASSWORD X'9A41632A'
CATJV   HUGO,WRPASS=C'SUSI',STATE=U
*end
PASSWORD : NORMAL   END
```

10.2.2.8 MODE - Select encryption routine

The MODE statement selects the encryption routine to be used in BS2000. The choice is between the “old” and the “SCA85” encryption routine.

Format

MODE / M
OLD / SCA

Operands

OLD / O
This means that the predefined default routine is to be chosen.

SCA
This means that the SCA85 encryption routine is to be used.

10.2.2.9 PASSWORD - Encrypt specified file password

This statement encrypts the specified file password. Both the encrypted and the unencrypted password are entered in the password table of the job. Both forms of the password are also output to SYSOUT. This means that although the file password is contained in its encrypted form in the catalog, the user can specify the unencrypted password when accessing the file, without the access being rejected as unauthorized.

Format

PASSWORD
fpassword

Operands

fpassword
File password consisting of 1 to 4 bytes (see “Format conventions for passwords” (Passwords)).

Example

```
/start-password
ENTER COMMAND NOW :
*password c'susi'
PASSWORD X'9A41632A'
PASSWORD C'SUSI'
*end
PASSWORD : NORMAL   END
```

11 PVSREN Renaming pubsets and volume sets

Version: PVSREN V21.0A

PVSREN V21.0A runs on version BS2000 OS DX and higher and supports SF and SM pubsets.

The utility routine PVSREN (Public Volume Sets REName) offers the following functions:

- Rename pubsets and volume sets within a notational type
- Convert one notation to another

For details on the format of SF and SM pubsets see the “Introduction to System Administration” [5 (Related publications)].

11.1 Prerequisites for PVSREN

- Prerequisites for the execution of PVSREN
- Prerequisites for renaming
- Prerequisites for renaming mirror pubsets

11.1.1 Prerequisites for the execution of PVSREN

PVSREN renames pubsets or converts pubset notations. For simplicity, this is referred to as “renaming pubsets” below.

The following basic requirements apply to the PVSREN program execution:

- PVSREN is executable only under the TSOS user ID.
- The default catalog ID can only be modified with the system privilege USER-ADMINISTRATION.
- PVSREN uses the SVC 79, so the system parameter SVC79 must be set to 0 or 1.

DRV pubsets

PVSREN also supports the renaming of DRV pubsets. To do this, all disks must be operated in dual mode, since an inconsistent status could otherwise be produced. This requirement is checked by PVSREN.

11.1.2 Prerequisites for renaming

If one of the following conditions is not satisfied, the pubset is not renamed.

- The pubset must have the status INACCESSIBLE since unwanted accesses to the catalog would otherwise be possible during renaming.

  If a volume set is to be renamed, the associated SM pubset must first be exported. It is not possible to export a volume set.
- All disks required must be available in the system in which the renaming is carried out.

  When renaming an SM pubset, all disks of the control volume set and also the Volres of all volume sets that belong to the SM pubset are required.

  When renaming a “normal” volume set, all disks of the volume set and also all disks of the control volume set are required.

  When renaming a control volume set, all disks of the control volume set and also the Volres of all volume sets of all volume sets that belong to the SM pubset are required.
- PVSREN must be able to exclusively reserve all the disks required.

  When renaming an SM pubset, a volume set of this pubset must not be renamed simultaneously.

  Renaming a volume set is possible only if no other volume set, or the SM pubset to which the volume set belongs, is renamed at the same time.
- There must be no disks with the destination VSN.

  Example: Renaming “ABC” to “X”

  > The pubset ABC consists of the disks ABC.00 to ABC.05: In this case, the disks PUBX00 to PUBX05 must not exist.
- If an MRSCAT entry with the destination catalog ID exists, it must not have the status ACCESSIBLE.If this MRSCAT entry has the status INACCESSIBLE, renaming is carried out only after confirmation of message `PVR0205`.
- When renaming an SM pubset or a control volume set, the SM pubset must not contain a volume set that has the status DEFECT.
- It is not possible to rename a volume set that has the status DEFECT.
- The length of the file, file generation groups and job variable names must not exceed the maximum length. The CHECK-FILENAME-LENGTH checks the length. This check is necessary only if the length of the catalog ID is increased.

  The length check is not necessary for volume sets, since the volume set ID is not incorporated in the file name.

Furthermore, the following points should be observed prior to renaming so as to ensure correct execution of the function:

- If an MRSCAT entry with the destination catalog ID exists, renaming should only be carried out on this catalog ID when this MRSCAT entry is no longer required. PVSREN deletes this MRSCAT entry and creates a new one.
- All private files that are used after renaming should have a catalog entry in the catalog of this pubset, since only then is the length check carried out for these files.
- It should be ensured that no ENTER jobs are present in the job pool, since these could possibly cease to be executable after renaming.

- Before renaming is carried out, all print jobs for files located on the pubset to be renamed should be completed. The jobs would otherwise remain in the queue and the pubset required cannot be imported because it has been renamed .
- If a home pubset is renamed, the following points must be observed:
  - In many computer centers, enter jobs in which this catalog ID is defined are started automatically at startup (e. g. ENTER to load SPEEDCAT). This catalog ID is now no longer valid and must be replaced by the new one.

    If the source file of the subsystem catalog <subsystem catalog>.SRC created by IMON is used, the catalog ID which is no longer valid must also be replaced by the new one in this file, too.
  - A FAST startup is executed using a parameter file. Parts which reference the VSN must be changed.
- Conversion from PUB to POINT notation is always possible. Conversion from POINT to PUB notation is possible only if the number of volumes is greater than 32.

11.1.3 Prerequisites for renaming mirror pubsets

When renaming mirror pubsets, the following prerequisites must be satisfied:

- All required disks must be available on the system on which the renaming process is to be performed.
- PVSREN must be able to exclusively reserve all the disks required.

11.2 PVSREN operation

PVSREN renames pubsets or volumes sets or converts pubset notations.
For simplicity, this is referred to as “renaming pubsets” below.

When files, file generation groups and job variables are referred to, the general term “object” is used.

The RENAME-PUBSET-OR-VOLUME-SET is used for renaming. This statement supports SF pubsets, SM pubsets and volume sets. By means of the MODE operand it is possible to select two different modes in order to rename a pubset or volume set:

1. Complete renaming of a pubset or volume set.
   In this case, the catalog ID, user catalogs and SCI are modified.
2. Renaming of a pubset or volume set after separation by SHC-OSD. In this case, the user catalogs and SCI are modified.

Complete Renaming should not be carried out until all the requirements have been checked (see section “Prerequisites for the execution of PVSREN”). Please also observe the “Restrictions and reworking”.

Renaming an SF pubset

1. The lengths of the object names are checked, even if the CHECK-FILENAME-LENGTH statement has been processed previously. All objects whose names exceed the maximum permissible length are listed. Renaming is not performed until there are no more object names that are too long.
2. The VSN is modified in the volume list of the catalog entry for files and file generation groups.
3. The VSN is modified in the SVL of all disks that belong to the pubset.
4. The VSN of all disks that belong to the pubset is modified in the OLC (online catalog on the Pubres).
5. The old MRSCAT entry is deleted for the SF pubset and a new entry created with the new catalog ID.
6. The default catalog ID in the user catalog (TSOSJOIN or SYSSRPM) of the home pubset and of the renamed pubset is modified. This action is carried out in interactive mode only after confirmation, and by default in batch mode. The SF pubset is also imported.

   The default catalog ID is modified with `/MODIFY-USER-ATTRIBUTES`.
7. Change the affected entries in the IMON-SCI on the data pubset and (following confirmation) HOME pubset.
8. The user can decide whether the pubset is to become or remain available.

Renaming an SM pubset

1. If a disk with corresponding VSN (e.g. ABC 00 for catid=ABC) is present for the MRSCAT entry of an SM pubset, the renaming operation is aborted, since it is not clear whether the SM pubset or the SF pubset, which also exists, is to be renamed.
2. A length check is performed for the object names, even if the CHECK-FILENAME-LENGTH statement has been executed. All objects whose names exceed the maximum permissible length are listed. Renaming is not performed until there are no more object names that are too long.
3. The catalog ID of the SM pubset is modified in the pubset configuration file on the control volume set.
4. The catalog ID of the SM pubset is modified in the SVL of each Volres of all SM volume sets that belong to the SM pubset.
5. The MRSCAT entries of all volume sets that belong to the SM pubset are deleted. They are recreated when the SM pubset is imported.

6. The old MRSCAT entry is deleted for the SM pubset and a new entry created with the new catalog ID.
7. The default catalog ID in the user catalog (SYSSRPM) of the home pubset and of the renamed pubset is modified. This action is carried out in interactive mode only after confirmation, and by default in batch mode. In the course of the operation, the SM pubset is imported.

   The default catalog ID is modified with `/MODIFY-USER-ATTRIBUTES`.
8. Change the affected entries in the IMON-SCI on the data pubset and (following confirmation) HOME pubset.
9. The user can decide whether the pubset is to become or remain available.

Renaming a volume set

1. The catalog ID of the volume set is modified in the pubset configuration file on the control volume set.
2. The VSN is modified in the volume list of the catalog entry for files and file generation groups.
3. The VSN is modified in the SVL of all disks that belong to the volume set.
4. The VSN of all disks that belong to the volume set is modified in the OLC (online catalog on the Volres).
5. A length check of the object names is not necessary, since the catalog ID of the volume set is not incorporated in the file name.
6. The catalog ID of the volume set cannot be used as the default catalog ID; therefore, it is not necessary to modify the default catalog ID. The associated SM pubset is not imported after renaming. This action must be carried out by the user.

Renaming a control volume set

1. The catalog ID of the control volume set is modified in the pubset control volume set.
2. The VSN is modified in the volume list of the catalog entry for files and file generation groups.
3. The VSN is modified in the SVL of all disks that belong to the control volume set.
4. The VSN of all disks that belong to the control volume set is modified in the OLC (online catalog on the Pubres).
5. The catalog ID of the control volume set is modified in the SVL of the primary disk of all volume sets that belong to the SM pubset.
6. The MRSCAT entries of all volume sets that belong to the relevant SM pubset are deleted. They are recreated when the SM pubset is imported.
7. The MSRCAT entry is deleted for the relevant SM pubset and the a new one created with the new catalog ID of the control volume set.
8. A length check of the object names is not necessary, since the catalog ID of the control volume set is not incorporated in the file name.
9. The catalog ID of the control volume set cannot be used as the default catalog ID; therefore, it is not necessary to modify the default catalog ID. The associated SM pubset is not imported after renaming. This action must be carried out by the user.

Completing a mirror disk separation initiated by SHC-OSD

After separating and renaming mirror disks using SHC-OSD (START-CLONE-SESSION .. UNIT=*BY-PUBSET (PUBSET=A, NEW-PUBSET=B)), the references in the IMON-SCI and user catalog are not yet adjusted.
This still has to be done using //RENAME-PUBSET-OR-VOLUME-SET ... MODE=*COMPLETE-SHC-RENAME, i.e.

1. (after query): Changing the standard catalog ID in the user catalog (SYSSRPM) of the home pubset and the changed pubset.

2. Changing the affected entries in the IMON-SCI on the data pubset and (after a query) also on the home pubset.

11.3 Restrictions and reworking

Files on Net-Storage when renaming SF or SM pubsets

If there are files in a pubset which reside on a default Net-Storage volume, the associated metadata of the file in the local pubset and on the Net-Storage volume is adjusted to the new pubset name.

If there are files in a pubset which reside on a Net-Storage volume with a user-defined VSN, the associated metadata of the file on the Net-Storage volume is adjusted to the new pubset name.

In both cases the Net-Storage files can be accessed without restriction after they have been renamed.

The administration of Net-Storage is described in the “Introduction to System Administration” [5 (Related publications)].

Modifying the default catalog ID

The default catalog ID can only be modified if the ID possesses the system privilege USER-ADMINISTRATION.

If the User ID does not possess this privilege, message `PVR0200` is used to decide whether renaming should nevertheless take place. If so, the default catalog ID can be modified later using the MODIFY-JOINFILE as soon as the User ID has the correct privilege.

Maximum number of volumes

Renaming from point notation to PUB notation is possible only if the number of volumes is not greater than 32.

Renaming from point notation to point notation is not possible if the new catalog ID has four digits and the number of volumes is greater than 36, since only one digit is available for the sequence number.

File protection with GUARDS

With a guard, you can also define the program that controls file access (e.g. $TSOS.EDT). Here, the catalog ID is also stored in the GUARDS catalog.

When the pubset where the program resides is renamed, the access condition is no longer satisfied: the catalog ID in the GUARDS catalog and the catalog ID of the pubset where the program resides no longer match.
In this case, the access conditions are changed for all guards that contain this catalog ID. This affects not only those guards on the renamed pubset and the home pubset, but also the guards on all other pubsets.

PVSREN does not provide support for this problem.

Catalog ID in user files

PVSREN cannot modify the catalog ID in the user files.

If a user file contains the catalog ID of a pubset that is to be renamed (e.g. in procedures), this catalog ID must be changed by the owner of the file as required.

Catalog ID in IMON files

After the installation of software with the installation monitor (IMON), the storage location of the installation item is noted in the software configuration inventory (SCI). The catalog ID is also stored.

If a pubset on which the software was installed using IMON is renamed, all the SCIs in which this software is registered must be changed. The SCIs of the HOME and renamed pubsets are changed by PVSREN. The SCIs of other pubsets must be changed manually using `/MODIFY-IMON-SCI`. Here, the name of the pubset prior to renaming must be specified for OLD-NAME, and the name to be applied to the pubset after renaming must be specified for NEW-NAME.

Shared pubset

PVSREN can only modify the MRSCAT entries of the system on which renaming is carried out.

Since there is an MRSCAT entry for a shared pubset on all systems that use the pubset, the MRSCAT entries must be adapted by system administration on all remote systems.

- For pubsets, the old MRSCAT entry must be deleted and a new MRSCAT entry created with the new catalog ID. For SM pubsets, the name of the control volume set has to be specified (the name can be new as well).

  The MRSCAT entries for the volume sets of an SM pubset must be deleted together with the pubset entry (default for the REMOVE-MASTER-CATALOG-ENTRY command). They will be newly created automatically. See also the note at the bottom of the page.
- When renaming the control volume set, the MRSCAT entry of the control volume set must also be deleted. The entry for the new control volume set is newly created during the import of the SM pubset. See also the note at the bottom of the page.

  In the MRSCAT entry of the SM pubset to which the control volume set belongs, the catalog ID of the control volume set must be entered using /MODIFY-MASTER-CATALOG-ENTRY.
- When renaming a “normal” volume set, only the MRSCAT entry of this volume set need be deleted. A new one need not be created, since this is done when the SM pubset is imported. See also the following note.

> **i** A volume set cannot be entered into the MRSCAT automatically if an identically named SF pubset or an identically named volume set of another SM pubset have already been entered. In that case, the previous entry must be manually deleted from the MRSCAT first.

Storage classes

Catalogs for storage classes and volume set lists exist on all SM pubsets. Storage classes have no direct references to catalog IDs. Volume set lists, on the other hand, assign one or more volume sets to a storage class. The entries in the catalog of the volume set lists therefore contain one or more catalog IDs of the SM pubset's volume sets.

When a volume set is renamed, this renaming also takes place in the volume set catalog to prevent the allocation of storage space for files via storage classes (explicitly specified or default storage classes in the user catalog) on the pubset from being affected by the renaming.

When a pubset is generated from mirror pubsets, the storage classes of the source pubset can either be accepted or deleted. Message `PVR0207`, which you can reply to, is output for this purpose. When the storage classes are accepted, the volume set lists (as in the case of renaming) are adjusted to the catalog IDs of the target pubset's volume sets.

Startup parameter file

When renaming a home or paging pubset, it might be necessary to adapt the startup parameter file. In addition to the VSNs of the paging disks, it can contain full file names with catalog ID.

SDF parameter file

`/MODIFY-SDF-PARAMETERS` can be used to activate a syntax file and specify its name with the catalog ID.

`/SHOW-SDF-PARAMETERS` can be used to determine whether a syntax file resides on the renamed pubset. If so, it must be deactivated using `/MODIFY-SDF-PARAMETERS` and reactivated with the new file name.

MIP parameter file

`/MODIFY-MIP-PARAMETERS` can be used to activate a message file and specify its name with the catalog ID.

`/SHOW-MIP-PARAMETERS` can be used to determine whether a message file resides on the renamed pubset. If so, it must be deactivated using `/MODIFY-MIP-PARAMETERS` and reactivated with the new file name.

SLED parameter file

If the renamed pubset contains a SLEDFILE that is being referenced in the SLED parameter file, the file has to be adjusted.

Repeat jobs

Repeat jobs, for which the ENTER file resides on the renamed pubset, must be restarted.

Print jobs

Print jobs for files residing on the pubset to be converted cannot be performed following the conversion. They must be deleted and new ones created.

Migrated files on a pubset

If an SF or an SM pubset is renamed, the ARCHIVE directory must be processed using the DIRCONV tool. PVSREN does not output messages to indicate that a pubset contains migrated files.

Example

```
/START-DIRCONV
//RENAME-CATID DIRECTORY-NAME = <directory name>
               OLD-CATID = *BY-SPECIFICATION(CATID = <old catid>),
               NEW-CATID = <new catid>,
               NEW-DIRECTORY-NAME = <new directory name>
//END
```

Then the new directory must be replaced with the old directory.

When renaming an HSMS-CONTROLLED volume set (S1 level), the S1 volume set must be modified using the HSMS statement MODIFY-SM-PUBSET-PARAMETERS.

11.4 Starting and stopping PVSREN

The PVSREN program is started under the user ID TSOS using `/START-PVSREN`.

START-PVSREN	Alias: **PVSREN**
VERSION = ***STD** / <product-version>	
,MONJV = ***NONE** / <filename 1..54 without-gen-vers>	
,CPU-LIMIT = ***JOB-RE**ST / <integer 1..32767 *seconds*>	

Using the MODIFY-LOGGING-OPTIONS statement the scope and output destination can be defined (see "MODIFY-LOGGING-OPTIONS - Modify default logging option values").

Starting by means of `/START-EXECUTABLE-PROGRAM FROM-FILE=PVSREN` is still supported for reasons of compatibility.

The **END** statement causes PVSREN to be terminated.

Format

END

This statement has no operands.

11.5 PVSREN messages

The messages of the PVSREN utility routine have the message class PVR.

See also the section “Messages and their meaning” (Notational conventions).

Their output destination is defined using the MODIFY-LOGGING-OPTIONS statement. The default output destination is SYSOUT.

11.6 Statements of PVSREN

- Overview of PVSREN statements
- Description of the statements
  - CHECK-FILENAME-LENGTH - Check length of file and job variable names
  - MODIFY-JOINFILE - Modify default catalog ID in user catalog
  - MODIFY-LOGGING-OPTIONS - Modify default logging option values
  - RENAME-PUBSET-OR-VOLUME-SET - Convert pubset notation or rename SF or SM pubset or volume set
  - RESTART-RENAMING - Restart conversion or renaming
  - RESTORE-LABELS-OF-PUBSET - Restoring the names of a pubset from mirror pubsets in double-point notation
  - SHOW-LOGGING-OPTIONS - List current logging option values

11.6.1 Overview of PVSREN statements

Statement	Function
CHECK-FILENAME-LENGTH	Check the length of file, file generation group and job variable names in the catalog of a pubset
MODIFY-JOINFILE	Modify the default catalog ID in the user catalog of a pubset
MODIFY-LOGGING-OPTIONS	Modify the current default logging option values for all program functions
RENAME-PUBSET-OR-VOLUME-SET	Convert pubset notation or rename SF or SM pubset or volume set
RESTART-RENAMING	Restart an interrupted conversion or renaming process
RESTORE-LABELS-OF-PUBSET	Rename mirror pubsets mit VSNs in double point notation to their original VSN
SHOW-LOGGING-OPTIONS	List the current logging option values

Furthermore, PVSREN supports the execution of the SDF standard statements (see the “SDF Dialog Interface” manual [20 (Related publications)]).

11.6.2 Description of the statements

- CHECK-FILENAME-LENGTH - Check length of file and job variable names
- MODIFY-JOINFILE - Modify default catalog ID in user catalog
- MODIFY-LOGGING-OPTIONS - Modify default logging option values
- RENAME-PUBSET-OR-VOLUME-SET - Convert pubset notation or rename SF or SM pubset or volume set
- RESTART-RENAMING - Restart conversion or renaming
- RESTORE-LABELS-OF-PUBSET - Restoring the names of a pubset from mirror pubsets in double-point notation
- SHOW-LOGGING-OPTIONS - List current logging option values

11.6.2.1 CHECK-FILENAME-LENGTH - Check length of file and job variable names

The CHECK-FILENAME-LENGTH statement checks the length of all file, file generation group and job variable names in the catalog of the pubset. The names must not exceed a specified maximum length (see below).

This function is useful only when performing renaming where the length of the catalog ID is increased. The statement is not executed unless the specified pubset has been set to “inaccessible” by means of `/EXPORT-PUBSET`.

If required, the length should always be checked prior to renaming. If the program detects names whose lengths exceed the specified maximum during the check, these are logged and PVSREN moves to the next STEP statement. The file or job variable names affected must be modified to the maximum permissible length using `/MODIFY-FILE-ATTRIBUTES` or `/MODIFY-JV-ATTRIBUTES`.

The program also checks whether the system ID in the SVL of the SF Pubres is initialized. If not, an appropriate message is output.

Since SM pubsets and volume sets cannot be used as home pubsets, the system ID has no meaning and is not checked.

Since the volume set ID is not incorporated in the file name, the CHECK-FILENAME-LENGTH statement is rejected for volume sets.

Maximum length of file and job variable names

The maximum length for a path name (with catalog and user ID) is 54 characters for file and job variables and 47 characters for file generation groups.

Structure of the path name:

```
:catid:$userid.<filename / fgg-name / jv-name>
```

The maximum length of the actual name (file, file generation group or job variable name) is 41 or 34 characters. This maximum length may be reduced if both the length of the catalogue ID and the length of the user ID are high. If the following calculation results in a value less than 41 or 34, this reduced value applies:

max. path name length (54 or 47)

- number of delimiters (always 4 (::$.))
- length of the catalog ID (1..4)
- length of the user ID (1..8)

Format

CHECK-FILENAME-LENGTH
CATID = <catid 1..4> ,NEW-CATID = <catid 1...4>

Operands

CATID = <catid 1..4>
The current catalog ID.

NEW-CATID = <catid 1..4>
The required catalog ID.

Example

```
(IN)  /CHECK-FILENAME-LENGTH CATID=A,NEW-CATID=ABC                      1.
(OUT) %PVR0134 The following object names exceed the maximum length     2.
(NL)  :<catid>:<userid>.<jv-name>    (JV)                               3.
(NL) *:<catid>:<userid>.<filename> (FGG)                                4.
(NL)  :<catid>:<userid>.<filename> (FGG)                                5.
(NL) *:<catid>:<userid>.<filename>                                      6.
(NL)  :<catid>:<userid>.<filename>                                      7.
```

1. The new catalog ID has 3 characters; this means that (with an 8-character user ID) the maximum length for file and job variable names is 39 characters, and 32 characters for file generation groups.
2. This message is output if names whose lengths exceed the maximum length are detected.
3. The output name represents ... a job variable.
4. ... a private file generation group. (* = private)
5. ... a public file generation group.
6. ... a private file. (* = private)
7. ... a public file.

11.6.2.2 MODIFY-JOINFILE - Modify default catalog ID in user catalog

The MODIFY-JOINFILE statement can be used to change the default catalog ID in the user catalog of an SF or SM pubset. There is one exception: under the TSOS user ID, a prerequisite for conversion is that the values for the PUBSET and NEW-DEFAULT-CATID operands are the same.

In order to execute this statement, the pubset must be imported.

Format

MODIFY-JOINFILE
PUBSET = <catid 1..4> ,DEFAULT-CATID = <catid 1..4> ,NEW-DEFAULT-CATID = <catid 1..4>

Operands

PUBSET = <catid 1..4>
Specifies the catalog ID of the pubset in which the user catalog in question resides.

DEFAULT-CATID = <catid 1..4>
Specifies the default catalog ID in the user catalog to be modified.

NEW-DEFAULT-CATID = <catid 1..4>
Specifies the catalog ID that is to replace the old one and become the new default catalog ID.

11.6.2.3 MODIFY-LOGGING-OPTIONS - Modify default logging option values

The MODIFY-LOGGING-OPTIONS statement enables default logging values for PVSREN to be modified globally for all program functions.

The user can request the current values via the SHOW-LOGGING-OPTIONS statement.

Format

MODIFY-LOGGING-OPTIONS
INFORMATION = *UNCHANGED / *MEDIUM / *MINIMUM / *MAXIMUM ,OUTPUT = *UNCHANGED / list-poss(2): *SYSOUT / *SYSLST

Operands

INFORMATION =
Gives the scope of the log generated by PVSREN.

INFORMATION = *UNCHANGED
The default logging option value remains unchanged. The default value is INFORMATION = *MEDIUM.

INFORMATION = *MEDIUM
Positive acknowledgments of special significance to the user are to be logged in addition to error messages.

INFORMATION = *MINIMUM
Only error messages are to be logged.

INFORMATION = *MAXIMUM
All PVSREN messages are to be logged.

OUTPUT =
Defines the output medium for the PVSREN log.

OUTPUT = *UNCHANGED
The default value for the output medium is to remain unchanged. The default value is OUTPUT=*SYSOUT.

OUTPUT = *SYSOUT
In interactive mode the output is to go to the terminal and in batch mode to the system file SYSOUT.

OUTPUT = *SYSLST
The output medium for logging is the system file SYSLST.

If SYSOUT and SYSLST are specified as lists, the logs are output to the system files SYSLST and SYSOUT (or to the terminal in interactive mode).

11.6.2.4 RENAME-PUBSET-OR-VOLUME-SET - Convert pubset notation or rename SF or SM pubset or volume set

This statement can be used to rename SF and SM pubsets and volume sets (also control volume sets) or convert their pubset notation.
The statement is executed only if the pubset has the status INACCESSIBLE.

The actions carried out depend on the mode.
When MODE=*NORMAL-RENAME is specified, the following steps are carried out one after the other by RENAME-PUBSET-OR-VOLUME-SET:

1. Check the length of the names of files, file generation groups and job variables (not for volume sets).
   If the maximum length is exceeded, processing is aborted. The CHECK-FILENAME-LENGTH statement should therefore always be called first.
2. Change the catalog ID of the pubset or volume sets
   - in the file catalog TSOSCAT (not for SM pubsets)
   - in the file name of the system files (paging area, TSOSCAT.<catid>)
   - in the SVL of all disks that belong to the SF pubset or volume set
   - in the pubset configuration file on the control volume set of the SM pubset
   - in the SVL of all main disks of the volume set of the corresponding SM pubset
3. Delete the old MRSCAT entries and create all new ones
4. Following confirmation, change the default catalog ID in the user catalog (SYSSRPM) of the home pubset and the modified pubset
5. Change the affected entries in the IMON-SCI on the data pubset and (following confirmation) HOME pubset.

If MODE=*COMPLETE-SHC-RENAME is specified, only actions 4 and 5 are executed, i.e.:

1. Following confirmation, change the default catalog ID in the user catalog (SYSSRPM) of the home pubset and the modified pubset
2. Change the affected entries in the IMON-SCI on the data pubset and (following confirmation) HOME pubset.

For a detailed description of execution, including any necessary reworking, please see "PVSREN operation" and "Restrictions and reworking".

Format

RENAME-PUBSET-OR-VOLUME-SET
NAME = <catid 1..4> ,NEW-NAME = <catid 1..4> ,SYSID = *STD / *SAME / <integer 65..192> ,MODE = *NORMAL-RENAME / *COMPLETE-SHC-RENAME ,PUBSET-PARAMETERS = *STD / *SAME

Operands

NAME = <catid 1..4>
Specifies the catalog ID of the pubset or the volume set to be modified (output catalog ID).

NEW-NAME = <catid 1..4>
Specifies the new catalog ID that the pubset or volume set is to receive after conversion or renaming (destination catalog ID).

SYSID =
The system ID is relevant only for home pubsets and is therefore interpreted only for SF pubsets. The system ID is entered in the SVL of the SF pubset and identifies the system which uses this home pubset in a shared pubset network. For SM pubsets and volume sets, the SYSID operand is ignored.

SYSID = *STD
Depending on the initial and target catalog IDs, the system ID is defined as follows:

catid-old	catid-new	Calculation of sysid
Single-digit	Single-digit	sysid = catid
Multi-digit	Single-digit	sysid = catid
Single-digit	Multi-digit	sysid s converted: A=65 ... Z=90, 0=91 ... 9=100
Multi-digit	Multi-digit	sysid is accepted

SYSID = *SAME
The system ID is taken from the initial pubset. This is possible only if the initial and target catalog IDs are multi-digit ones.

SYSID = <integer 65..192>
This specification is possible only if the target catalog ID is a multi-digit one. For a singledigit target catalog ID, the system ID is the same as the catalog ID and cannot be defined.

MODE =
This operand specifies the processing mode and thus the individual functions to be executed.

MODE = *NORMAL-RENAME
The specified pubset or volume set is completely renamed. In other words, the catalog ID, user catalog and SCI are modified.

MODE = *COMPLETE-SHC-RENAME
The partial renaming of a pubset by SHC-OSD at separation, i.e. the VSNs are renamed but not the links in IMON's SCI and in the user catalog (SYSSRPM), is completed by PVSREN. The user catalogs and the IMON-SCI are modified.
The modifications are only carried out following confirmation. If the HOME pubset has been separated by SHC-OSD, the following functions are not offered:

- renaming of user entries on the HOME pubset
- renaming of the IMON-SCI on the HOME pubset

Note that in this mode the NAME operand must be specified with the catalog ID of the original pubset and the NEW-NAME operand must be specified with the catalog ID of the pubset created by SHC-OSD.

PUBSET-PARAMETERS =
This operand defines which settings are transferred from the original pubset's MRSCAT to the MRSCAT of the renamed pubset. The operand is evaluated only when an SF or SM pubset is renamed, but not when a volume set is renamed.

PUBSET-PARAMETERS = *STD
The following MRSCAT settings of the original pubset are transferred:

- Pubset type
- Device type of the Pubres disk
- Settings for shareability on the local host
- BCAM name of the partner system

PUBSET-PARAMETERS = *SAME
In addition to the MRSCAT settings for PUBSET-PARAMETERS=*STD, the following MRSCAT settings of the original pubset are also transferred

- Number of buffers
- Buffer storage location
- Wait times for batch and interactive jobs
- Access restriction for users with any special permission
- XCS pubset property
- Remote import options
- Mnemonic device name of the Pubres disk
- EAM parameters

The following are also transferred in the case of SF pubsets:

- Settings for using SPEEDCAT
- Direct allocation restrictions
- Specific allocation parameters

> If renaming with PUBSET-PARAMETERS = *SAME fails, PVSREN displays message PVR0142 and tries a second run with PUBSET-PARAMETERS = *STD.

11.6.2.5 RESTART-RENAMING - Restart conversion or renaming

This statement restarts and completes an abnormally terminated or interrupted conversion of pubset notations or renaming of pubsets or volume sets.

If the processing of the RENAME-PUBSET-OR-VOLUME-SET statement terminates abnormally, the pubset or volume set is in an inconsistent state. One part of the data (e.g. the file catalog) has been modified, the other part (e. g. the VSN in the SVL) still has the old contents.

An inconsistency of this kind can be corrected using the RESTART-RENAMING statement. In doing so, the requested conversion or renaming is carried out so that the pubset can receive the specified new catalog ID.

Prerequisite for a successful restart is that the original disk configuration was not changed. A so-called logging file is also necessary; PVSREN automatically creates this file at the start of conversion or renaming and deletes it again on normal termination. This file contains all the data that is required for a restart of the aborted function.

The logging file is created under the user ID TSOS by default and has the name SYSLOG.PVSREN.<catid-old>.<catid-new>. It must be password-protected and must not be deleted, renamed or modified.

Format

RESTART-RENAMING
FROM-FILE = <filename 1..54>

Operands

FROM-FILE = <filename 1..54>
Name of the logging file created for abnormally terminated or interrupted conversion or renaming.

The name of the file is SYSLOG.PVSREN.<catid-old>.<catid-new> and is always stored under the user ID TSOS.

11.6.2.6 RESTORE-LABELS-OF-PUBSET - Restoring the names of a pubset from mirror pubsets in double-point notation

After a pubset has been restored from a mirror pubset in “double-point notation” (SPECIAL-VSN), this statement is used to rename the pubset’s VSN in POINT or PUB notation.

All disks belonging to a particular unit are reserved one after the other, and the double-point notation is replaced by point or PUB notation.

If the renaming process is aborted, it is possible to initiate a restart at any time. If this occurs, a warning is output for each disk already renamed, indicating that this disk is not in doublepoint notation. If a required disk is in neither double-point nor standard notation, the session is terminated. In this case, the missing disk must be made available and the renaming process restarted.

Format

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

Operands

CATID = <catid 1..4>
Catalog ID of the SF pubset.

11.6.2.7 SHOW-LOGGING-OPTIONS - List current logging option values

The SHOW-LOGGING-OPTIONS statement requests a list of the current logging option values. The values are specified by the MODIFY-LOGGING-OPTIONS statement.

Format

SHOW-LOGGING-OPTIONS

This statement has no operands.

The values that have been set are listed as follows:

```
%CURRENT LOGGING OPTIONS:
%  INFORMATION:<value>
%  OUTPUT:<value>
```

12 RMS REP Mounting System

Version: **RMS V7.1G**

Privileges: **TSOS**

The **REP Mounting System** (RMS) is used for the management, documentation, delivery and mounting of REP packets and preliminary corrections. The product is used both internally by service centers (user groups 1 and 2, see below) and by external customers (user groups 3 and 4, see below).

RMS was conceived as a four-level application for the following four user groups (see also "User groups and operations"):

- User group 1: development or equivalent
- User group 2: support center or service center
- User group 3: customer
- User group 4: customer's internal data center

All user groups have access to the same RMS functionality. The administration and implementation of all necessary operations is thus standardized and simplified. No additional effort is required on the part of the customer.
Each user group has its own file (depot) containing the following information specific to each user group:

- all REP corrections and descriptions
- their origin and the related product
- scope of all operations and the time at which they were carried out
- loader definition and history

All functions can be used in interactive mode and are menu-driven. Some functions can also be used in batch mode. This allows for the automatic operation of all essential functions - for all four user groups - without manual intervention and taking due consideration of all user-specific requirements.

> ¡ In RMS dialog, the term "system loader" is occasionally used as a synonym for "loader" (subsystem loader).

Use of RMS

The following diagram clarifies the sequence of events with RMS with respect to the four user groups.

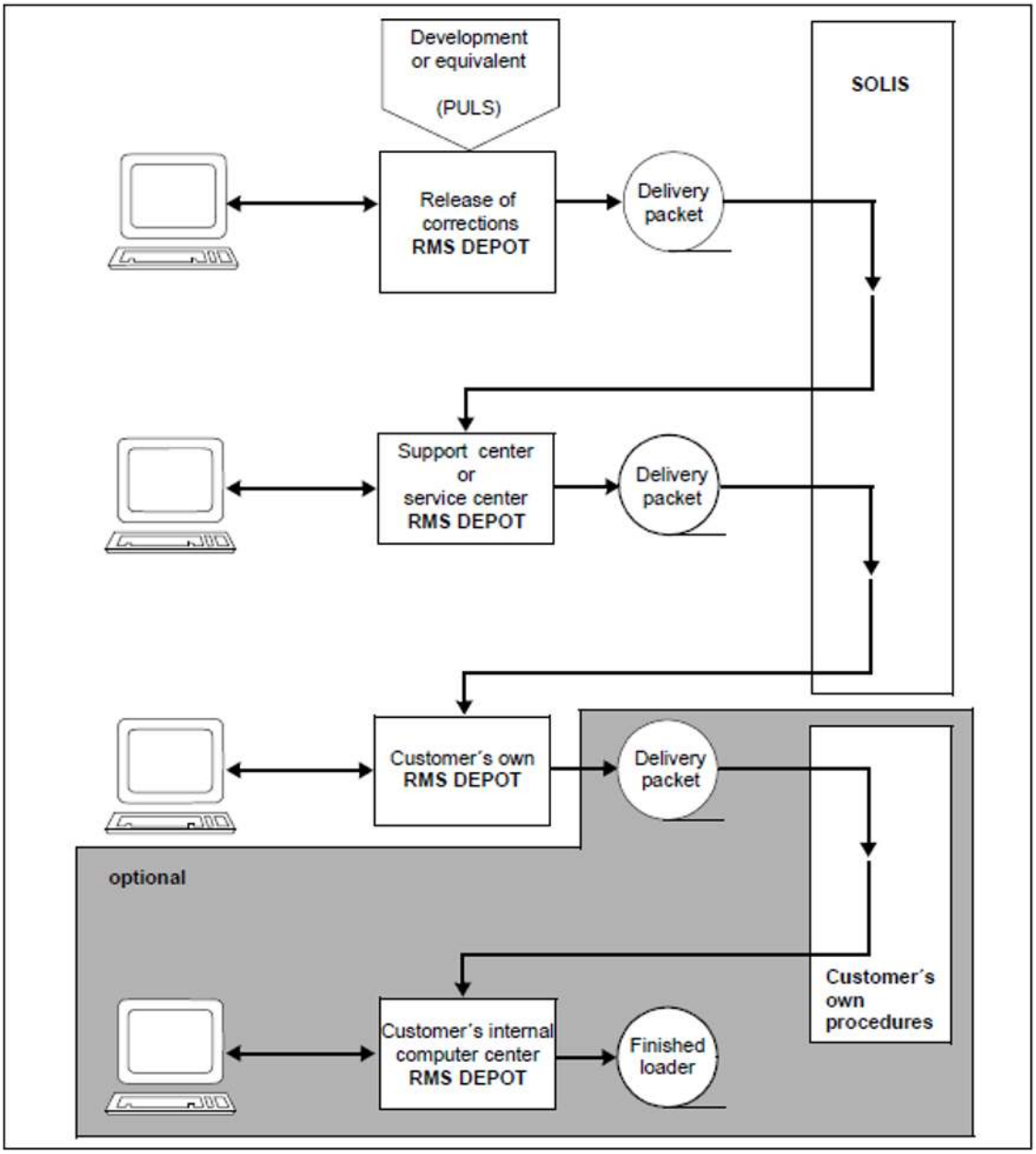


Figure 16: Use of RMS

Notes

1. A packet (or set) of software corrections (REPs) for a specific version of a product is generated from the PULS procedure. This packet is stored in a REP file.
2. This REP file (RMS delivery packet or RMS revision packet; both terms are used) is received by RMS and placed in a depot. All software product version-specific REPs, comments on these REPs, status (active, not activated) etc. are entered in this depot. The newly-input RMS delivery packets can be edited in the depot using RMS. For instance, they can be supplemented with your own additional REPs or comments or activated and deactivated.
3. From the depot, in turn an RMS delivery packet can be generated to be forwarded to the next user group. It is forwarded via SOLIS or the customer's own procedure. The next user group then uses RMS to import this RMS delivery packet into its depot and edit it further, if required.

A loader can also be generated from the depot. A loader is a REP file whose contents are applied when a product is started. It is generated by RMS as a user-group-specific selection of REPs for a particular product version, and is saved under a fixed, productspecific name.

4. Point 3 can be carried out by user groups 1, 2 and 3. User group 4 can only generate loaders.

12.1 User groups and operations

As a rule, the four user groups and the operations carried out between these groups can be defined as follows:

User group 1 - development or equivalent

Manages all released products as well as products in the pilot stage.

All corrections are subjected to a technical and format quality control. For released products, a defined correction packet is forwarded to the next user group at regular intervals, depending on the weighting of the product and the errors that have been corrected.

Products in the pilot stage are not forwarded via SOLIS2.

User group 2 - support center or service center

All deliveries from user group 1 are accepted into user group the own depot.

Modifications to the delivered correction packet may need to be made due to reported problems or preliminary corrections. These modifications are entered in user group the own depot. This ensures that the correction packets are always kept up to date.

When the deliveries or corrections are transferred to the customer, these modifications are automatically taken into account.

User group 3 - customer

All deliveries from user group 2 are accepted into user group the own depot.

Modifications to the delivered correction packet may need to be made due to reported problems or preliminary corrections. These modifications are entered in user group the own depot. This ensures that the correction packets are always kept up to date.

When the deliveries or corrections are transferred to the customer’s internal data center, these modifications are automatically taken into account.

User group 4 - customer’s internal data center

The deliveries from user group 3 are accepted. No additional steps are taken in the customer s internal data center.

12.2 RMS depot

The RMS depot is a BS2000 file and represents the central data pool of the RMS application. In the depot, all the corrections (REPs) are entered according to product, error number and date. Each user group has its *own* depot containing only that data which is required by that particular user group. All product-specific data required for the efficient and easy-to-follow management of corrections is thus contained in one file. All necessary operations can be carried out automatically or manually, depending on the product.

All user operations are entered into the depot with information on the type of operation and the time it was carried out. This guarantees transparency in the correction management process. Information about all operations carried out is easily obtainable. All operations can be reproduced at any time. Since all operations are clearly entered in the depot, every user has access to information about all operations carried out and can influence them to meet his or her own requirements. This guarantees audit security and data protection.

In addition, any required servicing is simplified and made transparent for all user groups.

All texts are organized according to text level name and text level key.

The logical structure of a depot looks like this:

- Product-specific section
  - REPs
  - Descriptions of corrections
  - Delivery information

  Depending on product scope of this user, the correction packets are created in the previous user group and imported into this depot.
- User-specific section
  - Modification of correction packets
  - Own or preliminary corrections
  - Rejected corrections
  - Correction documentation
  - All types of documentation
  - Data on loader construction management and generation
  - Selection of optional corrections
  - Data on automatic or manual loader generation/installation

12.3 Flexible use of RMS

RMS offers several options for the configuration of applications according to user requirements

- Any menu set-up can be created using a menu definition file (default file name `SYSDAT.RMS.071`).
- An *info RMS*, i.e. an information module purely for RMS, can be created by combining all info modules to form a separate module library.
- The user's own function modules can be entered in the RMS loader library and in the menu definition file in order to expand the functionality of RMS.
- All tasks which should run automatically can be saved as statements in a procedure or an ENTER job.

By selecting the corresponding functions and creating the corresponding batch jobs, RMS systems such as the following can be generated for specific requirements:

The simple application

The files required for RMS are to be added to the system. This includes the depot which was initialized by the previous user group.
All corrections are automatically added to the depot for each correction delivery. All required loaders are automatically generated and installed. Manual intervention is only necessary for modifications in product or operation system. It is run automatically using batch procedures (generally supplied with delivery) which are called up by the installation procedures belonging to the software product. This procedure is generally used at the customer level (user groups 3 and 4). The product selection must be redefined for new products with corrections which must be merged with the BS2000 system loader. These corrections are then taken into account during each new correction delivery.

The standard application

The scope of the loaders (products and optional corrections) is defined independently. If there are any errors, preliminary corrections are imported into the depot. All further operations run automatically, as in the simple application.

The complex application

The full functionality is used. The degree of automation for the running of operations is defined according to user requirements.

12.4 RMS operation

- Starting and exiting RMS
- Inputs to RMS

12.4.1 Starting and exiting RMS

RMS is started up with a startup procedure using the BS2000 statement

```
/CALL-PROCEDURE <procedure_name>
```

The supplied default startup procedure is called SYSPRC.RMS.071. This specifies the basic parameters for RMS. A description of the procedure can be found on "Startup procedure and basic parameters".

RMS is exited by pressing K1 on the highest menu level. Answer the prompt asking you if you would like exit RMS with Yes.

12.4.2 Inputs to RMS

The RMS functions are available via menu control in interactive mode. K and F keys are used for control purposes.

The MAR key is used for selecting or deselecting markings. These selections or the inputs in a screen must be confirmed with the DUE key (send key). The K1 key is used to cancel a function and return to the next-highest menu level.

You mark the required entries in a selection list by entering the character “/” or “x” in the input field ahead of the entry. You reset your selection by deleting the selector or overwriting it with blanks. These selections or the inputs in a screen must be confirmed with the DUE key (send key). The K1 key is used to cancel a function and return to the next-highest menu level.

You can scroll to an fro between the list pages by marking “next page” or “previous page” with the MAR key and confirming this with the DUE key.

Information on controls can be found at the bottom of every function screen. The K9 or ESC + ? keys can be used to request help information about the individual functions. A list of the function keys can be found on "Uses of the function keys".

The overall scope of RMS and therefore the functionality which is offered in the individual screens is controlled using the individual function modules (see "Module list") and is therefore fixed. However, a function file can be used to put together your own menu contents and the functions offered therein, as well as to change the menu titles and function names (see "Defining functionality"). This makes it possible, for example, to offer the same function in several menus. In the section “Overview of functions found on the default menu” you will find a description of the default RMS menu. Remember, however, that this may not be identical with the menu you are actually working with.

In batch mode, the RMS functions are available via a statement interface. Examples can be found in the section “Typical RMS applications”, and a description of all the statements on "Statements".

The following terms and abbreviations are used in the RMS menus and the supplied RMS documentation:

Term / abbreviation	**Function**
RMS delivery packet	On the one hand, the REP file for a product version obtained from PULS, which is incorporated into the depot using RMS. On the other hand, the transfer package generated from RMS, which is transferred to the next user group and incorporated into the depot there using RMS.
Loader/ system loader	A REP file whose contents are applied when a product is started. The loader is generated by RMS as a user-group-specific selection of REPs for a particular product version, and saved under a fixed, product-specific name.
REP file	A file containing all REPs for a product version. If this file is saved under a fixed, product-specific name, then it functions as a loader.
SUBNO	Correction number or error message number for a product version in the PULS procedure.
jd	Date according to the Julian calendar. For notation, see also "JD notation"
SC	Release unit

Table 14: RMS terms and abbreviations

12.5 Function overview

Every SOLIS2 shipment includes an RMS version assembled for SOLIS2. It is therefore not necessary to have previously installed RMS. The depot is managed automatically.

The following section provides a short overview of the RMS functions. At the end of the section you will find a list of the functions offered in the default menu supplied with delivery. Since these settings can be changed using the definition file, the actual menu may deviate from what is described. The current menu status can be referenced at any time, however, due to module assignments.

A description of the function masks available in interactive mode can be found starting on "Addition of correction packets"; the corresponding statements, if available, are described starting on "Statements".

12.5.1 Addition of correction packets

The correction transfer is the interface between the user groups. The functions available for this can be used in interactive or batch mode. Its ability to run in batch mode makes fully automatic operation possible.

The following functions are available:

- Deliver
  Delivery of defined correction packets and selected documentation, in fixed format, with automatic version numbering.
- Input
  The following operations are carried out with this function:
  - addition of new correction packets and documentation
  - automatic cross-matching with existing notebook operations
  - creation of a difference list

12.5.2 Loader services

For individual settings and running of operations, the following functions are available in interactive and batch mode.

- Define
  This function is used to define the scope and contents of a system loader. In addition to the product selection, the user can also execute the logical operation with OPT-REP settings and/or loader modifications.
- Install
  System loaders are generated at installation and cataloged under the agreed installation name. The required parameters are taken from the respective system loader definition.
- Create
  This runs the same as installation, up to the point of cataloging. If the created system loaders are to be used, then the files must be renamed manually to the valid installation names

The functions can to a certain degree be set to run automatically so that the user does not have to intervene manually.

The **system loader definition** defines the following settings:

- Installation attributes
  Setting customer-specific installation features.
- Product or product version selection
  Selection of products or product versions associated with this system loader.
- OPT-REP settings
  Selection of optional corrections which, if present, should be taken into account at every loader generation.
- Loader modifications
  Selection of corrections which should be taken into account at every loader generation regardless of its status.

The system loader definition can be saved under any name.

12.5.3 Correction management - Notebook

In correction management, modifications can be made with respect to the accepted correction packets using the notebook feature. The following functions are available:

- addition of user s own and preliminary corrections
- rejection of unsuccessful corrections
- exchange of corrections
- Reset notebook operations
- Information options

The entered notebook operations are automatically taken into account at the acceptance (input) of new correction packets, at correction transfer (delivery) and at loader generation.

12.5.4 Information services

Extensive product-specific and correction-specific information options are available in interactive mode.

Product-specific information

- Overview of all the products found in the depot (RMSST00 modules)
- Creation, acceptance and release date (MATRIX modules)
- Information about the notebook operations which have not yet been supplied (NBKINF modules)

Correction-specific information

- Assignment to product, input packets, and delivery packets (INFODATA modules)
- Global list of assignment to product, input packets and delivery packets (MATRIX modules)
- Comparison between original input packets, delivery packets and system loader files to determine possible differences (VGL modules)
- Notebook operations (NBKINF modules)
- Correction contents (INFODATA modules)
- Types of corrections (INFODATA modules)
- Correction-specific texts (INFODATA modules)

The INFODATA module can be called up using the function selection screen or using the K12 key.

When the K12 key is used, any other functions running are not closed. After this interactive information function is closed, the user returns to the interrupted function.

For most function groups, additional interactive and list information options are offered.

12.5.5 Editing

Texts for corrections or products can be entered, modified or deleted. These texts can be stored as passive or active information by means of a freely-definable hierarchical structure.

12.5.6 RMS management

The following tasks and functions are part of the administrative activities required for the running of RMS:

- Installation
- definition of overall functionality
- definition of functionality and access authorization for users
- depot maintenance (EXPORT/IMPORT)

12.5.7 Overview of functions found on the default menu

Option	Suboption	Module	Details[1] see
REP management			
	Correction management - Notebook (Utility Routines, #308)	RMSNOTIZ	
	Input revision packet into depot (Utility Routines, #306)	RMSEINF	
	Loader creation and installation (Utility Routines, #307)	RMSLGEN	
	Loader administration and installation	RMSLVWIN	Help
	Editing (Utility Routines, #316) (product notices, text entries)	RMSEDIT	
	Create revision packet (Utility Routines, #306)	RMSLIEF	
	Build all loaders waiting to be built	MONTAGE	[2]
	Call EDT	CALLEDT	EDT manual
Information services			
	Information services (Utility Routines, #309)	INFODATA	
	Character-oriented search (Utility Routines, #315)	RMSFIND	
	Comparing REP collections (Utility Routines, #314) (new format)	VGL	
	Comparing REPs collections (old format)	VGLSTART	Help
	Call EDT	CALLEDT	EDT manual
Global lists			
	Correction - global list (Utility Routines, #311)	Matrix	
	Notebook - product and REP information (Utility Routines, #312)	NBKINFO	
	Notebook - global list (Utility Routines, #313)	NOTBLIST	
	Summary list of depot contents	RMSST00	[2]
	Call EDT	CALLEDT	EDT manual
Administration			
	DEPOT maintenance (Utility Routines, #322) (Export - Import)	RMSEXIM	
	Reorganize delivery-revision packet versions	RMSREORG	Help
	Adjust RMS options	RMSOPTS	Help
	Modify a product version	RMSCNGPR	Help
	Create revision packet (RMS V6.0 Format)	LIEFERN	Help

	Call EDT	CALLEDT	EDT manual

[1] Detailed information on the individual function masks is also available via the help function; if "Help" is entered in the column, only via the help function.

[2] An output file is immediately generated.

12.6 Interactive application

The following section contains descriptions of the RMS functions for the menu-driven interactive mode. Help information can be requested for each function using the K9 key or ESC+?. Hilfeinformation anfordern.

12.6.1 Addition of correction packets

Create revision packet

This function can be used to create product-specific correction packets (supply components). These serve as the input for the depot of the next user group. Each supply component (SC) is identified with a version number (SC-Vers) which is automatically allocated in user groups 1, 2 and 3 and entered in the depot as the delivered packet.

In user group 4 (customer s internal computer center), only the delivery packets added to the depot can be recreated as a file.

Creation of current delivery packet

This function can only be used by user groups 1, 2 and 3.

To deliver current packets, i.e. supply components taking due account of the notebook, the corresponding supply component version must be input or marked in the selection screen.

When creating a delivery packet from user group 1, the loader letter suggested by RMS can be changed in the confirmation screen. In the following screen, the user can choose between a complete delivery of all text entries or a packet showing only the differences for a specific supply component version.

Before the delivery is carried out, a check is performed to determine if a description exists in the text level SC-OPT DESCRIPTION for all optional corrections. If a description is missing, then the SUBNO of the missing texts is listed in the log file.

If texts are missing, the delivery is terminated and the SUBNO of the missing texts is listed in the log file.

For the rest of the corrections, *all* text levels are searched and the first entry is written in the output file.

Each delivery can be checked. The field `Check Delivery` must be selected for this. All details of the desired delivery are checked.

Output file name:
The output file name can be defined in the statement line. Otherwise the following default file name is generated:

`SYSRMS.<product>.<version>.<sc-version>`

Create original delivery packet

This function is available to all 4 user groups. Each user can carry out this operation, since no entries and modifications are carried out in the depot.

To create original packets, i.e. accepted or already delivered packets, the field `Original delivery` must be selected.

Supply component version number

A version number has the following form: B01.03

The first three characters are assigned automatically by the central service, the last two by the regional service.

The loader letter (first character) has only been retained for historical reasons. It determines the order in which the delivery packets are created by the central service. (B00.00 before C00.00 etc.).

The numerical values are only raised in a current delivery if modifications (notebook entries) were made to the previous version in the respective depot.

In the case of a customer-internal delivery, the loader letter is changed from uppercase to lowercase (e.g. B ==> b).

Input revision packet into depot

This function is used to integrate supply components (REP packets, text packets and software configurations) which were created in the previous user group's depot into the user group's depot.

Every current supply component to be accepted contains all of the delivery depot's corrections which are valid at the time of creation

The following input logic applies:

- CORRECTIONS (REP)A correction is only physically saved in the depot once for each product. For future acceptance procedures, only a logical assignment is created.
- TEXTSEvery text is accepted, i.e. texts already existing in the depot are overwritten if they have the same text level name and text key.
- SW-KONFEvery product group is compared with the groups already saved in the depot and if they are not already in the depot, then the software configuration is extended to include these product groups. The attributes are always saved, but are only used actively if no modifications of the software configuration attributes are made in the depot.

A delivery packet can consist of several supply components (SC), text levels (TL) and software configurations (SW-C). An input packet can consist of several delivery packets.

The product-specific software configuration is a fixed component of the delivery file. Whenever a new delivery is added, the configurations found in the depot are automatically brought up to date.

Every supply component and text level is identified using its own control line. These are automatically generated during a delivery. The control lines must not be modified.

The control lines for a text level (TL) can be created by any user to set up his own TLs. 32767 text levels are available. On input of a packet, the contents of existing TLs can be modified. The creation and modification of text levels can also be performed using the text editor.

The format of the inputs for text levels has the following form:

Control line format

```
KKKTTTTTTTTTTTTTTTTTTTT*********dd/mm/yy**********************************
```

Explanation:

`KKK`	Text level identifier
`TTT...TTT`	Text level name (22 characters)
`dd/mm/yy`	Date

Example:

```
*T*SC-REP DESCRIPTION *********<date>**********************************
```

Text line format

```
PPPPPPPPBBBBBtexttext...
```

Explanation:

`PPPPPPPP`	PM (8 characters)
`BBBBB`	5 blanks or, if applicable, -xxx- whereby xxx is the jd
`texttext...`	PATCH-DESCRIPTION from PULS

Example:

```
*T*SC-REP DESCRIPTION   *********<date>***********************************
A0326168     FUNCTION NAME: Name of the OPT correction
A0346168     Here comes the PATCH-DESCRIPTION from PULS!
```

> For optional corrections, the text must be in both text levels (SC-REP and SC-OPT). The texts for optional corrections must begin with “FUNCTION NAME:'BLANK' ”.

The input packets can be assigned using the link name “EINDAT”, otherwise a prompt will come up in interactive mode asking for the file name.

For every product which is added to the depot, a comparison list is created in the log file.

Output file (edited for printing): INPUT.LIST.hhmmss

This log contains an overview of the differences for the last input packet of the product and the notebook cross-match. If the new packet is identical, interactive users can decide whether to accept it or reject it.

12.6.2 Loader services

RMS offers a closed loader service function for the maintenance of all components of a system loader.

The central element is a system loader definition. This is where all settings are saved which are required for the creation or installation of a system loader file.

Contents of a system loader definition

- The associated products.
  Normally, only one product is in the subsystem loader; in a so-called “BS2000 loader”, several products can be merged together and classified according to REP class.
- How the installation features are to be installed.
  A system loader file is installed under the path name defined in the system loader definition and receives the contents and settings defined there.
- Which, if any, OPT-REP packets are to be used.
  An OPT-REP packet is a selection of corrections which should be taken into account whenever a loader is generated. If a correction is not in the selected supply component version, this does *not* lead to an error.
- Which, if any, loader modification packets are to be used.
  A loader modification packet is a selection of corrections which should be taken into account at every loader generation, regardless of its status. A system-specific or taskspecific setting is possible in this case - e.g. hardware-specific corrections which must always be present in the system loader file.

Setting the scope and contents of a new system loader definition

The definition procedure consists of 7 steps:

1. Enter loader definition name and select “Administration”
2. Assign software configuration (optional)
3. Select product (mandatory)
4. Assign loader modification packets (optional)
5. Assign OPT-REP settings (optional)
6. Adapt installation attributes (optional)
7. Confirm settings and save system loader definition in depot

The following is an explanation of steps 2 to 7.

Assign software configuration

An overview displays all software configurations contained in this depot.
A software configuration can be selected to: simplify the product selection accept the default settings of the loader.

Select product

If no software configuration is selected, then all the products saved in the depot are offered for selection. Any combination is possible.

A system loader file can *never* be created if the combination does not conform to any software configuration!

Select the desired products or supply component version. If only the product is selected, then the current supply component packet will always be used for every system loader generation.

Assign loader modification packets

Overview of all the loader modifications to be found in the depot from which a selection can be made.

The current settings in the loader modification packets are taken into account at every system loader generation.

Assign OPT-REP setting

Overview of all the OPT-REP settings to be found in the depot from which a selection can be made.

The current settings in the OPR-REP packets are taken into account at every system loader generation.

Adapt installation attributes

The following are displayed:

- the already adapted attributes from an existing system loader definition
- an empty setting, if no software configuration was assigned
- the default settings for the assigned software configuration.

These can be modified for the system loader definition. A description of the installation attributes can be found in the following table.

Description of the installation attributes:

Attribute	Value	Remark
Password		A password can be used to protect the attributes from being overwritten. The password consists of up to 8 characters. If 8 blanks are entered, they are ignored. To delete a password, enter *NONE. If a password has been forgotten, it can be reset under TSOS.
Share	yes no	Definition of access authorization
Backup	yes no	Backup of the system loader file which is overwritten at installation. System loader files in BS2LOADER format are always backed up.
Installation	yes no	Installation should be carried out under the installation path name.
Information	Max	The following information is displayed on-screen when the system loader is installed: Name of the system loader and the generation version of the products in the loader, as well as the supply component version they used. All correction numbers for notebook entries which were active at creation. The system loader modifications carried out. All activated OPT-REPs.
Information	Med	The following information is displayed on-screen when the system loader is installed: Name of the system loader and the generation version. The products in the loader and the supply component version they used.
Information	Min	The following information is displayed on-screen when the system loader is installed: Name of the system loader and the generation version.
Function name	yes no	For the SUBNOs of the activated OPT-REPs, the function names are output. Only the PM numbers are output.
PUBLIC or PRIVATE		The output file can be PRIVATE if volume and device are entered.
Installation path name		Can be modified; *OWN as the user ID means that the user ID of the caller is always used. If *STD is input in the file name field, the values from the last supply component are used.
LOADER-FORMAT	BS2000_SUB-SYSTEM	BS2000 format is required at system loading; the subsystem is processed by DSSM.

Table 15: Description of installation attributes

Confirm and save

The settings for the system loader definition are now completely defined, but not yet saved in the depot. The following options are available:

DUE	Outputs an overview of the defined settings for further decisions
F1	Repeats modification process
F2	Saves
F3	Saves and then checks the system loader
K1	Terminates a process

Loader creation and installation

At system loader generation, an installable loader file is generated. The installation catalogs this loader file under the file name given in the system loader definition and extends the file with information about its predecessors.

The specifications required for operation, such as contents, scope and installation attributes, are contained in the system loader definition.

When selecting the system loader definition to be set up, you can stipulate whether:

1. all system loaders are to be installed regardless of their definition
2. no system loaders are to be installed, also regardless of their definition
3. the system loaders are to be generated/installed in accordance with the settings in their definitions.

Another option is the specification of an alternative catalog ID and/or user ID, with its prefix if applicable. This has the following consequences:

1. The path name of the loader file to be generated/installed is modified accordingly.
2. The “backup” parameter is ignored if
    - a prefix is entered, or
    - a user ID is entered which is not identical to the user ID in the system loader definition, or
    - *OWN is entered as the user ID in the system loader definition and the specified user ID is not identical to the caller s user ID.

System loader generation

An installable loader file with the following file name is created under the calling user ID: `loaderdefname.loaderversion`

System loader installation

When a system loader is installed, the selected system loader is created and cataloged under the installation name contained in the system loader definition.

Unless it has been modified manually, this installation name is the same as the loader file name required by the respective product. If a file with the same name already exists in the system, then this file will only be overwritten if it is a SAM file and also a loader file with the same product and product version. If there is a version change, such a check is not necessary and can be prevented by activating an RMS option. In interactive mode, this is performed by selecting OFF under “Check last product/version in loader file” in the “RMS options” function mask. In batch mode, this is performed by specifying NO-PRODUCT-CHECK=YES in the CREATE-RMS-OPTIONS statement.

With BS2000 system loader files and system loaders which have the setting *backup = yes* in their system loader definition, a file created by RMS that is to be overwritten is always backed up under the generation name. Other files are always backed up under their file names plus the suffix `.RMS.ddmmyy.hhmmss`. If this construction is too long for BS2000, the default file name `LOADER.BACKUP.RMS.ddmmyy.hhmmss` is used. This file name is incorporated into the new REP file as a comment and output at startup (REP processing), so that the old REP file can be used for a startup in case of emergency.

Unless otherwise specified, the loader files are created with predefined file protection criteria.

Repeat system loader generations

1. The `System loader` field must be selected in the loader generation mask.
2. Items are selected by being marked.
   In addition, the field `Old version` must also be marked.
3. Then an overview of all the system loaders generated with this name is output, together with the creation dates. After the desired loader version has been selected, the same system loader is generated as at the initial generation.

OPT-REP setting

All selectable corrections that are to be taken into account when the system loader is set up can be specified here. A selectable correction can be classified as belonging to one of the following groups:

- Type: O or U (an expansion of BS2000 functionality)
- Type: O or U (the reconstruction of an old procedure)
- Type: T (activates or deactivates a trace)
- Type: D (generates documents for the analysis of a problem)

To avoid problems during a version change, the selection of optional (functional) corrections is always carried out according to “family”, i.e. BS2000-GA (basic configuration) is offered, but without version designation.

Only the corrections which are found in the newest supply component version for each member of a family are offered.

Loader modifications

A loader modification packet offers the option of defining system loader-specific operations irrespective of the REP type. An example of this is a hardware-dependent REP which may only be used with certain systems.

Modifications options

Elements of the system loader functionality can be modified, duplicated, renamed and deleted.

- Modify everything (default)
  All settings and logical operations saved under the appropriate name in the depot can be modified.
- Modify attributes
  This field is only output for system loader definitions and software configurations. If this field is selected, the attribute settings can be edited.
- Delete
  The selected name is deleted when marked and sent off with the F1 key. If an OPT-REP setting management group or system loader modification which has already been linked with a system loader is involved, the DEL operation is performed in all system loader definitions containing this name.
- Rename
  The selected name can be given another name. All links are given the new name. The old name is no longer available.
- Copy
  The selected name can be copied under a new name. This is then available for further processing. The old name is not modified. You can only rename or copy to a name which is not already in the depot for the management group.
- Password
  This field is only output if the selected name is password-protected. To continue, the password must be entered.

Software configuration

A software configuration contains the default assignment of products to the installation file (system loader). This prevents erroneous merging and indicates any dependencies.

The product-specific software configuration is a fixed component of the delivery file. Whenever a new delivery is added, the configurations found in the depot are automatically brought up to date.

Modifying installation attributes

> ¡ All the settings, except the file name, are forwarded to the next user level upon acceptance (delivery) of an REP packet.

The values modified by the user are not changed at the next acceptance (input) of the software configuration. A description of the installation attributes can be found in the table on "Loader services".

With the aid of the function “Administration software configuration”, the user can change the supplied installation attributes for every software configuration and thus set his or her own default values.

12.6.3 Correction management - Notebook

The notebook is used to modify the contents of the depot. A modification is the change of status of a correction in the depot (deactivation, activation, entry of new corrections).

The notebook contains all deviations from the accepted correction packets and has the following attributes:

- Every operation is valid until it is accepted in a newly accepted supply component or another notebook operation is performed on this correction.
- All operations are always specific to a particular product version.
- Every operation is documented.
- All entries are taken into account whenever access is performed (exception: if the original packets are specifically requested).

A notebook session begins by entering the user ID and is ended by pressing the K1 key. The user ID is entered for every modification.

> i In the case of hardware-specific corrections, identifiers are used with which the first letter recognized by PULS becomes S (for /390 servers) or K (for x86 servers). When a PULS identifier is specified and the original identifier cannot be found, RMS searches for a corresponding hardware-specific version in the case of the following actions:
>
> - Acceptance of advance corrections (ADD with specification of a file)
> - Activation of depot corrections (ADD)
> - Rejecting of corrections (REJ)

All notebook operations can be output in interactive mode and on file.

The screen is the same for all functions. At the bottom of the screen, information about possible operations is given after an internal consistency check. The entry is not accepted until a suggested operation has been confirmed.

The corrections in the depot can be classified as follows:

- Source of correction
  - supply route (SOLIS2 or PULS)
  - ADD operation in notebook.
- Type of correction
  - normal correction
  - correction for selectable unit
  - optional correction
  - preliminary correction
  - correction for the creation of diagnostic documentation
  - correction for (de)activating trace routines

- Correction status in RMS
  - normal - i.e. no notebook operations were performed on these corrections.
  - ADD with assignment to some of the supply components (Some-ADD)
  - ADD with assignment to all supply components (All-ADD)
  - REJ (Rejected).

A notebook operation is dependent on the correction status and, where applicable, on the source of the correction, but not on the type of correction.

The notebook operations are shown in the following table and explained on the subsequent pages:

Operation	Description
ADD/CMD /SIS from file	A new correction has been made known (via FM2000, SIS or a telephone call), and it appears to be a good idea to use this correction and possibly forward it (deliver it).
ADD from depot	It has been determined that there is no current correction for this problem, but that a previous correction is good enough to be used.
REJ	It has been determined that a correction is erroneous and should no longer be included when generating or delivering system loader files.This operation remains valid until the correction is missing in a future delivery.
RST	Notebook operations were inadvertently carried out for a SUBNO. With this operation, *all* operations are undone.

Table 16: Overview of possible notebook operations

Acceptance of preliminary corrections - ADD with specification of a file

An ADD operation can be used to select a correction from a file, add it to the depot and assign it to a product.

This correction is assigned to all the supply components of the selected product and remains valid until a new correction is accepted for this SUBNO or another notebook operation (ADD, REJ, CMD, SIS or RST) is carried out. If there is already a correction in the depot for a SUBNO and a product, this becomes invalid (implicit REJ operation).

Mandatory specifications are:

```
subno jd 'ADD' comment product version filename
```

Acceptance of several preliminary corrections - CMD with specification of a file

The CMD operation is used to offer all corrections from the specified file for addition to the depot for the given product as individual ADD operations.

If an offered correction is to be added to the depot, a comment must be entered, followed by DUE. If a correction is not to be added, only DUE is necessary.

Mandatory specifications are:

```
'CMD' product version filename
```

Automatic acceptance of operations from SIS - SIS with specification of a file

The SIS operation is used to automatically process all corrections from the given file as individual ADD or REJ operations for addition to the depot for all the products contained in the file.

This operation is assigned to all supply components of the selected product and remains valid until a new correction is accepted for this SUBNO or until another notebook operation (ADD, REJ, CMD, SIS or RST) is carried out.

Mandatory specifications are (only for user group 2):

```
'SIS' filename
```

Activate depot corrections - ADD

An ADD operation without file specification will activate the specified depot correction for all supply components of the given product.

Mandatory specifications are:

```
subno jd 'ADD' comment product version
```

Rejecting of corrections - REJ

The REJ operation is used to reject a correction. This guarantees that this correction will no longer be contained in subsequent system loader generations or deliveries.

Mandatory specifications are:

```
subno jd 'REJ' comment
```

Product and version can also be specified. Regardless of whether or not this is done, all products containing this correction will be output. This makes it possible to restrict REJ to the specified product or implement it for all products.

If this correction is not in the depot, the operation can be noted for future reference. In such a case, the operation is only carried out when the correction is in a delivery packet.

This operation remains valid until another notebook operation (ADD, CMD, SIS or RST) is carried out.

Reset notebook operation - RST

The RST operation resets all notebook entries for a SUBNO. The original version (input packet) for this correction is returned to the depot.

Mandatory specifications are:

```
subno 'RST' comment
```

Product and version can also be specified. Regardless of whether or not this is done, all products containing this correction will be output. This makes it possible to restrict RST to the specified product or implement it for all products.

12.6.4 Information services

The information services consist of the following independent functions:

- Correction - product and REP information
- Correction - global list
- Notebook - product and REP information
- Notebook - global list
- Comparing REP collections
- Character-oriented search

12.6.4.1 Correction - product and REP information

Find correction number (PM no.)

The PM number to be found (SUBNO) is to be entered. To limit the search, a product family or product version can be entered.

An overview is output as the result of the search. By marking the individual fields, other information options are made available:

Correction assignment	When a product version is selected the resulting correction overview shows the original assignment “*” of the JD to the respective input supply component version.
Correction contents and texts	When the JD is selected, the correction contents and any related notebook entries or selected texts are output.

Product supply component information

When one of the following values is input in the “product” and/or “version” field, product transfer information is output.

*	All of the products saved in the depot are displayed (alphabetically). The information displayed is the delivery, the SOLIS date, the input date and the supply component.
?	An overview of all the products saved in the depot is offered and you can mark the appropriate entries to select information about specific products.
Product name	Transfer information for all the versions of this product is output
Only version	Transfer information for all products with this version is output
Product and version	Transfer information for this product with this version is output.

First the transfer information is displayed, i.e. all supply component versions are listed together with their creation and transfer dates.

12.6.4.2 Correction - global list

In the overview list, all correction numbers for one or more products and their affiliation to the individual supply component versions. A maximum of 16 supply component versions can be listed.

A log of all the input deliveries with the associated SOLIS2 date is written at the beginning of the log file.

Description of selectable fields

“Input - correction packet”
“Deliver - correction packet”

When one of these fields is selected, the input-specific or delivery-specific correction status can be listed. If neither of these fields is selected, the current correction status is listed, i.e. the input packet plus notebook operations.

“Packet to be selected”

If this field is selected, you have the option of determining the product scope of the list.

“Selection of first supply component version of the list”

If *yes* is selected, the following screen gives the option of determining the first supply component version of the evaluation.

“Listing”

If the “Summary” field is selected, a list is created with a summary of how many NEW, CNG or DEL corrections are in each supply component version.

“Module-specific subpacket”

This gives you the option of creating a list only for specific modules or modules plus displacement. On the following screen, the desired module and displacement, where applicable, can be entered.

12.6.4.3 Notebook - product and REP information

The procedure (search criteria) for this function is identical to that for the correction number search.

The desired notebook entries are output:

- Product-specific information

  Select the “File” field and the desired product.

  Output file: `NBK.ALL.ENTRIES.product.vers`
- JD-specific information

  Select the “File” field and the desired *jd*.

  Output file: `NBK.subno.jd.product.vers`

12.6.4.4 Notebook - global list

This list contains all input and notebook operations in the order in which they were carried out.
Output file (edited for printing): `#NOTEBOOK.LIST.hhmmss`. Concepts used:

EARMARK	The REJ operation cannot yet be carried out, since the correction is not yet in the depot. The entry is only taken into account once the correction is actually present in a newly-accepted supply component.
ACTIVE	The specified correction is the depot. This operation is taken into account until it is implemented in a newly-accepted supply component or changed by a notebook operation.
COMPLETE	The entry has been completed through either a new notebook operation or a newly-accepted supply component. This entry has no effect on newly delivered supply components.

12.6.4.5 Comparing REP collections

This function can be used to compare two REP packets. All new, missing and technical or formatting corrections are determined. The data is edited for printing and written to a temporary file.

Output file (edited for printing): `#hh-mm-ss.LSTVGL`

The following can be used as input:

- System loader file
- Other REP files
- A supply component version in the depot
- A supply component version in the depot plus all notebook operations
- A supply component version delivered from the depot

The REP packets to be compared are defined on-screen. In addition, the scope of the list can be determined by selecting the files “Short info” and/or “Without REPs”.

Short info	Identical corrections are not displayed in the global list.
Without REPs	The content of missing, new or changed corrections is not output.

Short information about the number of new (New), deleted (Del) and changed (Cng) corrections is output on the screen.

12.6.4.6 Character-oriented search

This function is used to find one or more strings in the depot. Either a selection or all of the products and/or text levels is searched.

A character-oriented or module-oriented search can be selected.

Per product or text level, an overview of the correction or text entries is output in which the specified string is found. Detailed information (corrections or text) can be output by selecting the desired entries.

For the following (default) setting, the entire depot is searched:

Same scope	off
Products	on
Texts	on

The default settings *Lower off* and *Case on* convert the input search string and the internal texts into uppercase and compare them. The output is in the original form.

The following concepts can be activated (“on”) and deactivated (“off”). This enables what is to be searched to be determined and influenced:

Define scope	A product selection screen follows, in which products can be selected to limit the search to these products. *Same scope* is automatically activated.
Same scope on	The previous product selection is retained.
Texts on	Text levels are searched.
Products on	Corrections and notebook entries are searched.
Lower on	The specified search string is not converted into uppercase letters.
Case on	Lowercase letters in the depot are converted into uppercase for the search.

12.6.5 Editing

The texts can be entered as

Global	PRODUCT is blank, SUBNO is blank, JD is blank
Product-specific	SUBNO is blank, JD is blank
SUBNO-specific	JD is blank
SUBNO-JD-specific	Everything is defined.

Text levels

Individual text level keys are accessed in the different text levels. To edit text levels, the “Text level” field must be selected. Then the text levels to be edited must be selected on the follow-up screen.

Product notes

These notes are only used to draw attention to something. When a new delivery packet is accepted, these are output in the input protocol when the product and, where applicable, the SUBNO/JD match up

To edit product notes, the product must be selected for which a note is to be entered.

Text editor

The text editor is used to edit (enter, delete and modify texts) text levels and product notes. A distinction is made between SUBNO-oriented and global texts.

Example

> In the text level “SC-OPT DESCRIPTION”, the text for the SUBNO 41000077 is to be changed. First the “Text level” field is selected, then on the follow-up screen, the text level “SC-OPT DESCRIPTION” is selected. Now the text editor is activated. After entering the SUBNO, the text can be modified.

Explanation of the editing screen

The screen is divided into three sections,

1. A control area with the texts found in the depot. This window cannot be overwritten.
2. A modification area; in this window, the original text can be modified.
3. A selection column. When specific characters are entered in the selection column, lines in the work window can be marked for particular applications.

   d The sentences marked with *d* are deleted.

   n Number of lines to be added (1-9). The lines are added before the lines marked with *n*.

Operation fields and operating notes

Low OFF /ON	“Low” determines whether or not the text editor is to convert lowercase letters into uppercase. When this field is selected, the default setting can be set to ON or OFF. Please note: in contrast to EDT, the “Low” setting is evaluated before the text is edited. OFF: The editor converts the lowercase letters into uppercase. ON: Characters are processed as input.
Delete (+F1)	By selecting this field and sending it off with the F1 key, the text for the displayed SUBNO is deleted from the depot.
F2	The modified text is saved in the depot.
F3	The changes to the displayed SUBNO are ignored.
K1	Exits the text editor.

After a SUBNO has been edited, the next highest SUBNO is offered for editing.

12.6.6 RMS management

- Installation
- Startup procedure and basic parameters
- Defining functionality
- Example of a definition file
- DEPOT maintenance

12.6.6.1 Installation

The following files are required for the running of RMS:

`SYSPRC.RMS.071` or any name	Procedure for calling up the RMS program. In this startup procedure, the basic parameters must be agreed upon for RMS
`SYSOML.RMS.071` or any name	Object module library containing all the modules required for running the program.
`SYSDAT.RMS.071` or any name	Defines functionality and access authorization. An additional option exists which allows for the loading mode of the modules and the format and contents of the menu screens to be determined.
`RMS.DEPOT` or `RMS.DEPOT.`*`suffix`*	Unless otherwise specified, RMS accesses the RMS.DEPOT file. When a .*suffix* is entered, it is possible to keep several depots under one user ID. Any .*suffix* can be selected. This type of file name can be used by entering the basic parameter “FAM” in the startup procedure. If the expected file is not available, a depot with the defined name is created.

12.6.6.2 Startup procedure and basic parameters

This section describes the default startup procedure which is supplied with RMS. This can be modified to suit requirements. Of key importance in this respect are the basic parameters, with which computer center and user-specific function can be set.

Minimum scope of the startup procedure

```
/PROC N
/     OPTION DUMP=YES
/     FILE SYSOML.RMS.071,LINK=RMSOML
/     SYSFILE SYSDTA=(SYSCMD)
/     EXEC (RMS,SYSOML.RMS.071)
```

Maximum scope of the startup procedure (shipped with RMS)

```
/PROC N,(&ID=TSOS,/
        &DEF=TSOS,/
        &DEPOT=TSOS,/
        &FAM=*NONE,/
        &K1OPT=JA,/
        &LCASE=JA,/
        &LMAX=60,/
        &MODE=DIALOG,/
        &V7SYNTAX=JA,/
        &LAN=D,/
        &PTASTEN=JA,/
        &TEMPID=#),/
        SUBDTA=&
/        REMARK ALL SETTINGS ARE RMS DEFAULT SETTINGS
/        OPTION DUMP=YES                                          1.
/        FILE $&DEF..SYSDAT.RMS.071,LINK=DEF                      2.
/        FILE $&ID..SYSOML.RMS.071,LINK=RMSOML                    3.
/        SYSFILE SYSDTA=(SYSCMD)
/        SKIP  .&MODE
/.DIALOG REMARK ** CALL RMS IN INTERACTIVE MODE (MENU-DRIVEN)
/.DIA    EXEC (RMS,$&ID..SYSOML.RMS.071)                          4.
MODE=DIALOG/&LAN,DEPOT-USER=&DEPOT,FAM=&FAM,K1OPT=&K1OPT,-        5.
LCASE=&LCASE,PTASTEN=&PTASTEN,TEMPF=&TEMPID,LMAX=&LMAX
/        ENDP
/.SOLIS  REMARK ** RMS ACTS LIKE SOLRMS **
/.BATCH  REMARK ** RMS IS STARTED UP IN BATCH MODE **
/        EXEC (RMS,$&ID..SYSOML.RMS.071),IDA=NO
MODE=&MODE/&LAN,DEPOT-USER=&DEPOT,FAM=&FAM,LCASE=&LCASE,
TEMPF=&TEMPID,LMAX=&LMAX,V7SYNTAX=&V7SYNTAX
/BREAK
/ENDP-RESUME
```

1. A user dump is generated if RMS terminates abnormally.
2. Assignment of the definition file if it is found under a different user ID. This entry is optional. If no FILE statement is given or the file is not found under the link name, RMS accesses the local file SYSDAT.RMS.071. If this is not available either, RMS outputs an error message.
3. The module library from which the module is to be loaded is assigned.
4. The RMS pre-linked module is loaded from the module library.

5. Supply the basic parameters required for loading RMS

Description of basic parameters

DEPOT-USER	User ID under which the depot is cataloged, with catalog number, if applicable. Freely-selectable parameter, default setting is *OWN (caller s user ID).
FAM	A string can be defined which is added to the default name RMS.DEPOT. This extended DEPOT file name is then used. FAM=*NONE means the default name is RMS.DEPOT. Freely-selectable parameter, the default setting is *NONE
K1OPT	JA / YES when exiting RMS using the K1 key, you are asked to confirm that you really want to exit RMS. NEIN / NO RMS is exited with no additional request for confirmation. Freely-selectable parameter, the default setting is YES
LCASE	JA / YES printout uppercase/lowercase NEIN / NO printout only uppercase Freely-selectable parameter, the default setting is YES.
LMAX	Determines the number of lines per log or list page generated by RMS. Possible range: 20-99. Freely-selectable parameter, the default setting is 60.

MODE / M	BATCH Batch mode is simulated. RMS reads the inputs with RDATA from SYSDTA. COMMAND Batch mode is simulated. RMS reads the inputs with RDATA from SYSDTA. DIALOG RMS is called up as a mask-oriented interactive program. SOLIS Batch mode is simulated. RMS reads the inputs with RDATA from SYSDTA. If RMS is called up with MODE=SOLIS, generation of the loader files with FUNCTION=SELECT-REPFILE-TO-BUILD is suppressed. This loader file is generated using BUILD-REPFILE. IMON Just like SOLIS. The function SELECT-REPFILE-TO-BUILD also permits specification of the RUNAME, RUVERS and SCI operands. Then, the IMON macro SETINSP is supported for the generated loader (see the “IMON” manual [15 (Related publications)]). TEST Batch mode is simulated but the statements are only checked and not executed. This parameter should only be used in connection with V7SYNTAX=YES. Entering /D or /E after the MODE parameter results in all texts being output in German or English, as appropriate. Freely-selectable parameter, the default setting is DIALOG/D
PTASTEN PKEYS	JA / YES the P keys are loaded NEIN / NO the P keys are not loaded. Freely-selectable parameter, the default setting is NO
TEMPF	ID for temporary files. Any special characters for temporary files supported by the operating system are permitted. Freely-selectable parameter, default setting is #.
V7 SYNTAX	JA / YES all statements are checked according to V7 syntax rules and executed, where applicable. NEIN / NO the syntax rules from Version 6 are used. Freely-selectable parameter, the default setting is YES

12.6.6.3 Defining functionality

The function file is required to load RMS. It can be defined with a FILE statement (LINK=DEF) in the startup procedure. If no name is assigned, then the default file name SYSDAT.RMS.071 is used. An example of a definition file can be found on "Example of a definition file".

The following function statements are available:

MODE MEISTER	This statement is used to define the write authorization for the depot.
LANGUAGE	Selects the language for the texts - only effective if no language character (D/E) is specified in the MODE parameter.
MENU	The contents and number of menu screens can be adapted to user requirements.
MOD	Defines the overall functionality of RMS. In addition, the load mode of the function modules and their titles for the menu screens can be defined.
ENDMENU	End statement for the MENU statement.
comment	For documentation purposes.

Description of the statements

MODE MEISTER

This statement permits all functions defined with MOD to be executed and must always be the first statement in the function file. If missing, the depot can only be accessed in read-only mode, regardless of the defined functionality.

LANGUAGE=DEUTSCH or ENGLISH

The texts can be output in German or English.

This statement is only evaluated if no language is defined in the MODE parameter in the startup procedure.

MENU,TEXT='menu-title max. 48 characters'

The “menu-title” appears as the title line of the menu screen or as a function title identifying it as a submenu. A submenu is always its own menu screen.

This statement must precede the first MOD statement.

MOD=modulename[,TEMPORARY | RESIDENT],TEXT='function name'

The `modulename` is the entry/module name of the function and must be written in uppercase letters.

The `function name` appears on the menu screen as the description of this function.

The user can decide when the module is loaded:

- `TEMPORARY`: The module is not loaded until the function is called up and it is unloaded on exiting.
- `RESIDENT`: The module is loaded when starting RMS and remains resident for the entire RMS session.

If neither of these entries is present, the function module will be loaded at first call-up and will remain resident for the entire session.

The text accompanying the TEXT statement may be up to 48 characters long and appears on the menu screen as the heading for this function.

For a list of the module names, see appendix A.

ENDMENU

End criterion for a menu (menu screen). If another MENU statement precedes the ENDMENU statement, all the MOD statements (functions) in between are displayed in a submenu.

comment

If the first character of a sentence is an asterisk (*), the sentence will be recognized as a comment and skipped. The comment must not exceed 99 characters.

12.6.6.4 Example of a definition file

This example shows the construction of a definition file with maximum functionality. It shows such things as the meaning of the MENU and MOD statements and their effects on the menu or submenu screens.

Multiple definitions (same module) are possible. See EDT in this example. This makes this function appear on all three menu screens.

EDT and LMS are not required by RMS. They can be defined, however, to avoid any program changes.

```
MODE MEISTER
LANGUAGE=ENGLISH
MENU,TEXT='RMS - Production system'
   MOD=INFODATA,TEXT='PRODUCT AND REP INFORMATION'
   MOD=NOTIZERF,TEMPORARY,TEXT='NOTEBOOK MANAGEMENT'
   MOD=RMSLGEN,TEXT='LOADER CREATION and INSTALLATION'
   MOD=RMSLVWIN,TEXT='LOADER ADMINISTRATION And INFORMATION'
   MOD=EINFAHR,TEMPORARY,TEXT='INPUT REVISION PACKET INTO DEPOT'
   MOD=LIEFERN,TEXT='CREATE REVISION PACKET' MOD=CALLEDT,TEXT='EDT'
   MENU,TEXT='INFORMATION SERVICES and LISTS'
      MOD=INFODATA,RESIDENT,TEXT='PRODUCT AND REP INFORMATION'
      MOD=RMSFIND,TEXT='CHARACTER-ORIENTED SEARCH'
      MOD=NBKINFO,TEXT='NOTEBOOK INFORMATION (product-specific)'
      MOD=VGLSTART,TEMPORARY,TEXT='COMPARE REP COLLECTIONS'
      MOD=CALLEDT,TEXT='EDT'
      ENDMENU
   MENU,TEXT='GLOBAL LISTS'
      MOD=MATRIX,TEXT='CREATE MATRIX LISTING'
      MOD=NBKINFO,TEMPORARY,TEXT='SELECTED NOTEBOOK ENTRIES'
      MOD=NOTBLIST,TEXT='GLOBAL NOTEBOOK LIST'
      MOD=CALLEDT,TEXT='EDT'
      ENDMENU
   MENU,TEXT='ADMINISTRATIVE FUNCTIONS'
      MOD=RMSEXIM,TEMPORARY,TEXT='DEPOT MAINTENANCE (export - import)'
      MOD=RMSEDIT,TEXT='TEXT EDITING (notes, text levels)'
      MOD=CALLEDT,TEXT='EDT'
      ENDMENU
   ENDMENU
```

The tabs are not essential, but they make the example easier to read.

The INFODATA module is always loaded as a component of the resident RMS and entry of TEMPORARY has no effect.

12.6.6.5 DEPOT maintenance

EXPORT function - Overview

The export function is used for depot maintenance. Products, text levels and the loader definitions for system loaders, loader modifications and OPT-REP settings can be deleted from the depot in use. The data to be deleted (except text levels) is always written to a backup depot which is regenerated for every export function. Special backup depots can be created without deleting in the original depot.

Output file name of the backup depot: `RMS.DEPOT.yymmdd.hhmmss`

This backup depot can be

- processed as a normal depot with RMS
- used as an input depot for the IMPORT function

Output file name of text levels: `RMS.TEXTLEVEL.name`

This file can be accepted only with the RMS function “Input”.

IMPORT function - Overview

The IMPORT function can be used to accept components (products, system loader definitions, loader modifications and OPT-REP settings) into the open depot from another depot. Only components which are not yet in the open depot can be accepted.

It is not possible to import text levels.

Operation

The desired operations are selected by marking the individual fields. Follow-up screens enable you to make a selection within an existing supply component.

The IMPORT function is only activated by entering a depot file name.

Description of selectable fields

Products	The products selected on the follow-up screen are imported or exported along with their associated notebook entries and deliveries. This option enables you to delete a product which is no longer required.
System loaders	The system loader definitions selected on the follow-up screens are exported or imported along with their associated loader modifications, OPT-REP settings and software configuration
Loader modification	The loader modifications selected on the follow-up screen are imported or exported.
OPT-REP setting	The OPT-REP settings selected on the follow-up screen are imported or exported
Text level	The text levels selected on the follow-up screen are output as SAM files with the name RMS.TEXTLEVEL.name if <name> is the name of the text level. It is not possible to import text levels.
Delete	Components should be deleted after export; RMS checks for any dependencies on the system loader definitions and outputs any it finds on the screen. The option enables you to decide whether or not deletion is to be carried out. At this point, the backup depot has already been created. F1 All selections are deleted. DUE None of the selections are deleted. The contents of the screen can be output to SYSLST by overwriting the “?” field with a character which is not a blank and sending it off with the DUE key.

12.7 Typical RMS applications

This section describes the most important details involved in the execution of a number of typical RMS applications.

12.7.1 Setting up a depot

Task

A brand new depot is to be set up for a system configuration. RMS always creates a new depot whenever it is started and tries to address a depot which does not exist. The name of the new depot is RMS.DEPOT.<family>, where “family” is the parameter of the RMS startup procedure.

Procedure using the statement interface

Call up RMS with the following BS2000 statement and input the RMS statements; this generates a depot with the name RMS.DEPOT.TEST. Please note that the parameters ID, DEF and DEPOT are assigned to TSOS by default.

```
CALL-PROC SYSPRC.RMS.071,(FAM=TEST,V7SYNTAX=YES,MODE=BATCH)
   INPUT-DELIVERY-PACKET INPUT=<filename>
   END
```

The type of depot created depends on the version of the accepted delivery packet:

Delivery packet version	Type of depot created
#nn.nn	PULS depot
Ann.00	Support center or service center depot
Ann.xx	Customer level 1 depot
ann.nn	Customer level 2 depot

Explanation:

nn = 00 .. 99
xx = 01 .. 99

12.7.2 Inputting a new delivery packet into the depot

Task

New delivery packets (corrections) are to be input into the depot, after which a new loader is to be built for an application.

Procedure using the dialog interface

In interactive mode, several delivery packets can be added to the depot at the same time. Under the function “Input revision packet into depot”, enter the name of the file which contains the delivery packet. When a wildcard is entered, a screen follows which allows you to select several files and thereby input several delivery packets at the same time.

12.7.3 Activating optional REPs

Task

Optional REPs for the loader of a SPOOL version are to be activated. Optional REPs are always deactivated and must be activated in the depot before a loader is created.

Procedure using the statement interface

First the desired optional REP packet must be created. To do this, start up RMS as follows and generate the packet using the corresponding RMS statement:

```
CALL-PROC SYSPRC.RMS.071,(FAM=TEST,V7SYNTAX=YES,MODE=BATCH)
   CREATE-OPT-PACKET NAME='OPT.SPOOL.049A.2',INPUT=*SYSDTA
   ADD=(A1111111,A2222222)
   ADD=A3333333
   END
   END
```

Now connect the optional REP packet with the loader.

```
MODIFY-REPFILE-DEFINITION NAME='LADER.SPOOL.049.2',-
                          ADDOPTS='OPT.SPOOL.049A.2'
END
```

12.7.4 Building loaders in interactive mode

Task

New loaders are to be generated and installed for a product. Before this can be done, both the loaders and the software configuration must be defined in the depot.

Procedure using the statement interface

Start up RMS and select the product for which loaders are to be generated:

```
CALL-PROC SYSPRC.RMS.071,(FAM=TEST,V7SYNTAX=YES,MODE=BATCH)

SELECT-REPFILE-TO-BUILD PROD=SPOOL V4.9A
```

All loaders which have SPOOL V4.9A in their software configuration are selected. Now the selected loaders can be generated.

```
BUILD-REPFILE

END
```

12.7.5 Building loaders using a batch file

The following example shows how a loader is built using a batch file. The operation can be fully automated if the statement interface is used.

```
/ BEGIN-PROCEDURE
/           SET-PROC-OPTIONS LOGGING=YES,DATA-ESCAPE-CHAR='&&'
/           BEGIN-PARAMETER-DECLARATION
/           DECL-PARAM NAME=RMS         (TYPE=STRING ,INIT='RMS')
/           DECL-PARAM NAME=RMSOML      (TYPE=STRING ,INIT='SYSOML.RMS.071')
/           DECL-PARAM NAME=RMSFUNC     (TYPE=STRING ,INIT='SYSDAT.RMS.071')
/           END-PARAMETER-DECLARATION
/           MODIFY-JOB-SWITCHES OFF=*ALL
/           &*
/           &* RMS CALL
/           &* --------
/           &*
/           ADD-FILE-LINK LINK-NAME=RMSOML,FILE-NAME=&RMSOML
/           ADD-FILE-LINK LINK-NAME=DEF,FILE-NAME=&RMSFUNC
/           ASSIGN-SYSDTA TO-FILE=*SYSCMD
/           START-EXECUTABLE-PROGRAM *MOD(ELEMENT=&RMS,LIBRARY=&RMSOML)
LMAX=55,FAM=EX,V7SYNTAX=JA,MODE=BATCH
CREATE-RMS-OPTIONS PROTOCOL=EX.PROT,ERROR=ON(30)                          1.
CREATE-SW-CONF NAME='SYSREP.SPOOL.049',ADD=SPOOL V4.9A,ERROR=ON(30)       2.
CREATE-REPFILE-DEFINITION NAME='LADER.SPOOL.049.1',PASS='SPOOL',-          3.
           ADDPROD=SPOOL V4.9A,ERROR=ON(30)
SELECT-REPFILE-TO-BUILD PROD=SPOOL V4.9A,-                                4.
              USERID=$RMSD,ERROR=ON(30)
BUILD-REPFILE ERROR=ON(30)                                                5.
END
/           ASSIGN-SYSDTA TO-FILE=*SYSCMD
/           SKIP-COMMANDS TO-LABEL=TCERR,IF=JOB-SWITCHES(ON=30)
/           SKIP-COMMANDS TO-LABEL=TCOK
/.TCERR     REMARK
/           REMARK **** ERROR PROCESSING ****
/           SKIP-COMMANDS TO-LABEL=ENDP
/.TCOK      REMARK
/           REMARK **** OK PROCESSING ****
/.ENDP      REMARK
/           MODIFY-JOB-SWITCH OFF=*ALL
/EXIT-PROCEDURE
```

1. RMS option settings. The file EX.PROT will be used as a log file.
2. Definition of a software configuration SYSREP.SPOOL.049 including the product SPOOL V4.9A. This is a precondition for any loader creation as a loader must be linked with a software configuration.
3. Definition of a loader for product SPOOL V4.9A. This is automatically linked with the software configuration SYSREP.SPOOL.049.
4. Select the loader to be built. Here all the loaders containing the product SPOOL V4.9A in their definition will be selected.
5. Build the selected loader. The selected loaders are generated/installed in accordance with the options specified in their definition. The generation/installation user ID will be $RMSD as it has been specified in the SELECT-REPFILE-TO-BUILD statement.

12.7.6 Transferring a REP packet to the next user group

Task

A REP packet for SPOOL V4.9A and delivery packet A00.01 is to be transferred to the next user group (this applies to user groups 1 to 3). A corresponding delivery packet must be generated.

Procedure using the statement interface

Start up RMS with the statement

```
CALL-PROCEDURE SYSPRC.RMS.071,(FAM=TEST,V7SYNTAX=YES,MODE=BATCH)
```

Now the delivery packet can be generated with the statement

```
CREATE-DELIVERY-PACKET PRODUCT=SPOOL V4.9A/A00.01
```

If you have received several delivery packets in one RMS session and wish to forward them directly to another user group, then this can be done using the following statement:

```
CREATE-DELIVERY-PACKET PROD=*PRODSAVE
```

Procedure using the dialog interface

Select the function “Create revision packet” in the “REP management” menu. In the following screen, you can select the supply component for which a delivery packet is to be generated.

12.7.7 Rejecting an invalid REP

Task

You wish to reject an invalid REP (here REP A9999999/280 for SPOOL V4.9A) which was received with a delivery packet in order to prevent a loader or delivery packet from being generated with this REP. This is performed with a notebook function.

Procedure using the statement interface

Start up RMS with the statement

```
CALL-PROCEDURE SYSPRC.RMS.071,(FAM=TEST,V7SYNTAX=YES,MODE=BATCH)
```

Now enter the following RMS statement to generate the corresponding note:

```
MODIFY-DELIVERY-PACKET IDENT='myname',REJ,A9999999/280,SPOOL V4.9A
```

Procedure using the dialog interface

Select the function “Notebook management” in the “REP management” menu. The following screen allows you to enter the REP you wish to reject. More detailed information about the individual fields can be obtained using the online help for the function (input ESC + ? or K9).

12.8 Statements

The statements described in this section replace all the statements from RMS versions before V7. For reasons of compatibility, the old statements can still be used. A description of the old statements can be found starting on "Statements of RMS before V7.0".

To use the new statements, enter the following RMS initiation parameter:

```
V7SYNTAX = JA / YES
```

The simultaneous use of old and new statements is not possible.

The metasyntax of the statements is that for ISP statements and is described on "Notational conventions".

Operation

The statements are first checked for syntax errors and written to a temporary file. Depending on what is specified for the MODE operand, either the statement is processed from the temporary file (normal case) or, in the case of MODE=TEST, the run is ended. If a syntax error is detected, the run is ended without any processing taking place.

If products, OPT packets or MOD packets are missing, deleted entries are entered in the depot during the second editing phase, i.e. no context check is carried out.

12.8.1 Generating and editing a software configuration

- CREATE-SW-CONF - Create new software configuration
- MODIFY-SW-CONF - Modify software configuration
- DELETE-SW-CONF - Delete software configuration

12.8.1.1 CREATE-SW-CONF - Create new software configuration

A new software configuration is created, an already existing software configuration is overwritten. All settings previously found in this entry are lost. At the end, all the system loader definitions are checked. If a default setting is determined, the settings from the newly-entered software configuration are used instead.

Format

```
CREATE-SW-CONF

NAME = 'swkonfname'

[,PASS = 'password']

,ADD = productversion / (productversion,...)

[,SETTINGS = (

      ,SHARE = NO / YES

      ,SAVE = NO / YES

      ,INSTALL = NO / YES

      ,INFORMATION = MAX / MED / MIN

      ,FUNCTION-NAME = NO / YES

      [,VOLUME = vol,DEVICE = dev]

      [,PATH = [:catid:][$userid.]filename]

      ,LOADER-FORMAT = SUB-SYSTEM / BS2000 )]

[,ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)]
```

Operands

NAME = 'swkonfname'
Name of the software configuration to be created.

PASS = 'password'
Password used to protect the software configuration.

ADD = productversion / (productversion, ...)
Product and, where applicable, product version to be added to the software configuration. The product name can be up to 15 characters long. The product version is to be entered in the format V[n]n.nn[nn].

SETTINGS =
Determines the loader attributes with respect to the software configuration. The first operand named can be entered without a leading comma; any order can be used.

SHARE = NO / YES
Defines access authorization.

SAVE = YES / NO
Backup of the system loader file which is overwritten at installation. System loader files in BS2LOADER format are always backed up.

INSTALL = YES / NO
Install under the installation path name.

INFORMATION = MED / MAX / MIN
If MAX is specified, the following information is output on the screen when the system loader is installed:

- Name of the system loader and the generation version of the products in the loader, as well as the supply component version they used.
- All correction numbers for notebook entries which were active at creation.
- The system loader modifications carried out.
- All activated OPT-REPs.

For information on the output for MIN/MED, see "Loader services".

FUNCTION-NAME = NO / YES
For the SUBNOs of the activated OPT-REPs, the function names are output. If NO is selected, only the PM numbers are output.

VOLUME = vol, DEVIVE = dev
Volume and device under which the corresponding loader is stored.

PATH = :catid:$userid.filename
Path name under which the corresponding loader is stored.

LOADER-FORMAT = SUB-SYSTEM / BS2000
BS2000 format required at system loading; the subsystem is processed by DSSM.

ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)
In the event of an error, the specified job switch is set (ON) or reset (OFF).

12.8.1.2 MODIFY-SW-CONF - Modify software configuration

An existing software configuration is modified. If the software configuration is not available in the depot, *no* software configuration is created

Format

```
MODIFY-SW-CONF

NAME = 'swkonfname'

[,NEW-NAME = 'swkonfname']

[,PASS = 'password'][,NEW-PASS = 'password']

[,{ ADD / DEL } = productversion / (productversion,...)]

[,SETTINGS = (

      ,SHARE = NO / YES

      ,SAVE = NO / YES

      ,INSTALL = NO / YES

      ,INFORMATION = MAX / MED / MIN

      ,FUNCTION-NAME = NO / YES

      [,VOLUME = vol,DEVICE = dev]

      [,SHARE = [:catid:][$userid.]filename]

      ,LOADER-FORMAT = SUB-SYSTEM / BS2000 )]

[,ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)]
```

Operands

NAME = 'swkonfname'
Name of an existing software configuration that is to be modified.

NEW-NAME = 'swkonfname'
New name of the software configuration.

PASS = 'password', NEW-PASS = 'password'
Password with which the software configuration is protected. If the software configuration is protected with a password, this must be entered before any attributes can be modified. A new password can be assigned with NEW-PASS= password .

ADD / DEL = productversion/ (productversion, ...)
Product and, where applicable, product version to be added to or removed from the software configuration. The product name can be up to 15 characters long. The product version is to be entered in the format V[n]n.nn[nn].

SETTING = (...)
For a description of the SETTINGS operands, see the CREATE-SW-CONF (Utility Routines, #333) statement.

ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)
In the event of an error, the specified job switch is set (ON) or reset (OFF).

12.8.1.3 DELETE-SW-CONF - Delete software configuration

An existing software configuration is deleted.

Format

DELETE-SW-CONF
NAME = 'swkonfname' [,PASS = 'password'] [,ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)]

Operands

NAME = 'swkonfname'
Name of an existing software configuration that is to be deleted.

PASS = 'password'
Password with which the software configuration is protected. If the software configuration is protected with a password, this must be entered before the software configuration can be deleted.

ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)
In the event of an error, the specified job switch is set (ON) or reset (OFF).

12.8.2 Optional REP selection

- CREATE-OPT-PACKET - Create optional REP selection packet
- MODIFY-OPT-PACKET - Extend optional REP selection packet
- DELETE-OPT-PACKET - Delete optional packet

12.8.2.1 CREATE-OPT-PACKET - Create optional REP selection packet

A new element is created in the depot under the name `optpacketname`. An existing element is overwritten.

Format

CREATE-OPT-PACKET
NAME = 'optpacketname' ,INPUT = *SYSDTA / filename [,ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)]

Operands

NAME = 'optpacketname'
Name of the optional REP selection packet to be created. The name can be up to 22 characters long.

INPUT = *SYSDTA / filename
The following ADD statements are read from SYSDTA or are in the specified file. “filename” can be a fully-qualified name.

The *SYSDTA input or the input file has the following format:

```
ADD = subno / (subno, ...) [,productfamily]
...
ADD = subno / (subno, ...) [,productfamily]
END
```

“`productfamily`” is the product name without a version number.

ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)
In the event of an error, the specified job switch is set (ON) or reset (OFF).

12.8.2.2 MODIFY-OPT-PACKET - Extend optional REP selection packet

An element in the depot is modified. If `optpacketname` is not in the depot, a new element is created.

Format

MODIFY-OPT-PACKET
NAME = 'optpacketname' [,NEW-NAME = 'optpacketname'] ,INPUT = *SYSDTA / filename [,ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)]

Operands

NAME = 'optpacketname'
Name of the optional REP selection packet to be modified. The name can be up to 22 characters long.

NEW-NAME = 'optpacketname'
New name of the optional REP selection packet to be modified. The name can be up to 22 characters long.

INPUT = *SYSDTA / filename
The following ADD statements are read from SYSDTA or are in the specified file. “filename” can be a fully-qualified name.

The *SYSDTA input or the input file has the following format:

```
ADD = subno / (subno, ...) [,productfamily]
...
ADD = subno / (subno, ...) [,productfamily]
DEL = subno / (subno, ...) [,productfamily]
END
```

ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)
In the event of an error, the specified job switch is set (ON) or reset (OFF).

12.8.2.3 DELETE-OPT-PACKET - Delete optional packet

The specified element is deleted.

Format

DELETE-OPT-PACKET
NAME = 'optpacketname' [,ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)]

Operands

NAME = '*optpacketname*'
Name of the optional REP selection packet to be deleted. The name can be up to 22 characters long.

ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)
In the event of an error, the specified job switch is set (ON) or reset (OFF).

12.8.3 Loader modifications

- CREATE-MOD-PACKET - Create loader modification element
- MODIFY-MOD-PACKET - Modify loader modification element
- DELETE-MOD-PACKET - Delete loader modification element

12.8.3.1 CREATE-MOD-PACKET - Create loader modification element

In the depot, a new element `modpacketname` is created. An existing element is overwritten.

Format

CREATE-MOD-PACKET
NAME = 'modpacketname' ,INPUT = *SYSDTA / filename [,ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)]

Operands

NAME = 'modpacketname'
Name of the loader element to be created. The name can be up to 22 characters long.

INPUT = *SYSDTA / filename
The following ADD/DEL statements are read from SYSDTA or are in the specified file. “filename” can be a fully-qualified name.

The *SYSDTA input or the input file has the following format:

```
ADD / DEL = subno[-jd],productversion,TEXT='reason'
...
ADD / DEL = subno[jd],productversion,TEXT='reason'
END
```

> For ADD, the date -jd must be entered.

ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)
In the event of an error, the specified job switch is set (ON) or reset (OFF).

12.8.3.2 MODIFY-MOD-PACKET - Modify loader modification element

An existing element is modified. If `modpacketname` is not in the depot, a new element is created.

Format

MODIFY-MOD-PACKET
NAME = 'modpacketname' [,NEW-NAME = 'modpacketname'] ,INPUT = *SYSDTA / filename [,ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)]

Operands

NAME = 'modpacketname'
Name of the loader modification element to be modified. The name can be up to 22 characters long.

NEW-NAME = 'modpacketname'
New loader element name. The name can be up to 22 characters long.

INPUT = *SYSDTA / filename
The following ADD/DEL statements are read from SYSDTA or are in the specified file. “filename” can be a fully-qualified name.

The *SYSDTA input or the input file has the following format:

```
ADD / DEL = subno[-jd],productversion,TEXT='begründung'
...
ADD / DEL = subno[-jd],productversion,TEXT='begründung'
END
```

> For ADD, the date -jd must be entered.

ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)
In the event of an error, the specified job switch is set (ON) or reset (OFF).

12.8.3.3 DELETE-MOD-PACKET - Delete loader modification element

The specified element is deleted.

Format

DELETE-MOD-PACKET
NAME = 'modpacketname' [,ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)]

Operands

NAME = '*modpacketname*'
Name of the loader modification element to be deleted. The name can be up to 22 characters long.

ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)
In the event of an error, the specified job switch is set (ON) or reset (OFF).

12.8.4 REPFILE definitions

- CREATE-REPFILE-DEFINITION - Create new loader definitions
- MODIFY-REPFILE-DEFINITION - Modify loader definitions
- DELETE-REPFILE-DEFINITION - Delete loader definition element

12.8.4.1 CREATE-REPFILE-DEFINITION - Create new loader definitions

The desired loader definitions are created under the element `loaderdefname`. An existing element is overwritten.

An attempt is made to assign a software configuration to the product definition. If this is not possible, the run is terminated. If the definition is incomplete, a warning is output and the run is continued.

Format

CREATE-REPFILE-DEFINITION

NAME = 'laderdefname'

[,PASS = 'password']

,ADDPROD = *PRODSAVE / productversion / (productversion,...)

[,ADDMODS = 'modpacketname' / ('modpacketname',...) ,...]

[,ADDOPTS = 'optpacketname' / ('optpacketname',...) ,...]

[,SETTINGS = (

,SHARE = NO / YES

,SAVE = NO / YES

,INSTALL = NO / YES

,INFORMATION = MED / MAX / MIN

,FUNCTION-NAME = NO / YES

[,VOLUME = vol,DEVICE = dev]

[,PATH = [:catid:][$userid.]filename]

,LOADER-FORMAT = SUB-SYSTEM / BS2000)]

[,ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)]

Operands

NAME = 'loaderdefname'
Name of the loader definition to be created.

PASS = 'password'
Password used to protect the loader definition.

ADDPROD = *PRODSAVE / productversion / (productversion, ...)
The products to be contained in the REP file are defined.

*PRODSAVE means that a list of products is accessed which was specified in a previous statement. RMS remembers this list and uses it again here.

A product or a series of products (max. 10) can be contained in the REP file.

ADDMODS = '**modpacketname**' / **('modpacketname', ...)**
The loader modification elements which are to be contained in the REP file are defined. The loader modification elements are created or processed using the statement CREATE- or MODIFY-MOD-PACKET.

An element or a series of elements (max. 10) can be contained in the REP file.

ADDOPTS = 'optpacketname' / ('optpacketname', ...)
The optional REP selection packets which are contained in the REP file are defined. The optional REP selection packets are created or processed using the statement CREATE- or MODIFY-OPT-PACKET.

A packet or a series of packets (max. 10) can be contained in the REP file.

> The sum of the specified ADDMODS / ADDOPTS operands must be <= 9.

SETTINGS = (...)
For a description of the SETTINGS operands, see the CREATE-SW-CONF (Utility Routines, #333) statement.

ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)
In the event of an error, the specified job switch is set (ON) or reset (OFF).

12.8.4.2 MODIFY-REPFILE-DEFINITION - Modify loader definitions

This statement modifies the specified attributes of the selected element `loaderdefname`. If the loader definition is not in the depot, a new element is created.

An attempt to execute DELPROD, DELOPTS or DELMODS for a loader definition which does not exist is only noticed when the operations are carried out and is rejected with a warning. RMS responds likewise when the product version/modpacketname/optpacketname of the DEL operand cannot be found in the depot.

When a new loader definition is created, an attempt is made to establish an unambiguous assignment to a software configuration saved in the depot. If this is successful, the preset values from the software configuration are used and, where applicable, overwritten by the operand values in the statement.

Format

MODIFY-REPFILE-DEFINITION

NAME = 'loaderdefname'[,NEW-NAME = 'loaderdefname']

[,PASS = 'password'[,NEW-PASS = 'password']][,{ ADDPROD / DELPROD } = *PRODSAVE / productversion

/ (productversion,...)]

[{ { ADDMODS / DELMODS } = 'modpacketname' / 'modpacketname',...) /

    { ADDOPTS / DELOPTS } = 'optpacketname' / 'optpacketname',...)

} ,...]

[,SETTINGS = (

    ,SHARE = NO / YES

    ,SAVE = NO / YES

    ,INSTALL = NO / YES

    ,INFORMATION = MED / MAX / MIN

    ,FUNCTION-NAME = NO / YES

    [,VOLUME = vol,DEVICE = dev]

    [,PATH = [:catid:][$userid.]filename]

    ,LOADER-FORMAT = SUB-SYSTEM / BS2000)]

[,ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)]

Operands

NAME = 'loaderdefname'
Name of an existing loader definition to be modified.

NEW-NAME = 'loaderdefname'
Name of the new loader definition.

PASS = 'password', NEW-PASS = 'password'
Password used to protect the loader definition. When assigning a new password, the old password must be entered under PASS, and the new password must be entered under NEW-PASS.

ADDPROD / DELPROD = *PRODSAVE / productversion / (productversion, ..)
see CREATE-REPFILE-DEFINITION

ADDMODS / DELMODS = 'modpacketname' / ('modpacketname',...)
see CREATE-REPFILE-DEFINITION

ADDOPTS / DELOPTS = 'optpacketname' / ('optpacketname',...)
see CREATE-REPFILE-DEFINITION

> The sum of the specified ADDMODS / DELMODS / ADDOPTS / DELOPTS operands must be <= 9.

SETTINGS = (...)
For a description of the SETTINGS operands, see the CREATE-SW-CONF (Utility Routines, #333) statement.

ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)
In the event of an error, the specified job switch is set (ON) or reset (OFF).

12.8.4.3 DELETE-REPFILE-DEFINITION - Delete loader definition element

The selected element is deleted.

Format

DELETE-REPFILE-DEFINITION
NAME = 'loaderdefname' [,PASS = 'password'] [,ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)]

Operands

NAME = 'loaderdefname'
Name of the loader definition to be deleted.

PASS = 'password'
Password used to protect the loader definition.

ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)
In the event of an error, the specified job switch is set (ON) or reset (OFF).

12.8.5 Creating the system loader file

- SELECT-REPFILE-TO-BUILD - Select system loader definitions
- BUILD-REPFILE - Create and install system loader

12.8.5.1 SELECT-REPFILE-TO-BUILD - Select system loader definitions

This statement selects the elements for the loaders to be used. The system loader definitions are created along with the next BUILD-REPFILE statement.

Format

SELECT-REPFILE-TO-BUILD
{ NAME = ’laderdefname’ / (’laderdefname’,...) /
,PRODUCT = *PRODSAVE / productversion / (productversion,...) }
,INSTALL = DEF / ABS / NO
[,USERID = [:catid:][$userid.]]
[,PREFIX = prefix]
[,RUNAME = releaseunitname]
[,RUVERS = releaseunitversion]
[,SCI = imoninventoryname]
[,ADDOPTS = 'optpacketname' / ('optpacketname',...)]
[,OVERWRITE = YES / NO]
[,SETTINGS = STD / SWK]
[,ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)]

Operands

NAME = 'loaderdefname'
Name of the system loader definition to be selected. A list of up to 30 names can be entered.

If NAME is not specified, PRODUCT must be specified. In this case, all the loaders with this product in their definition are selected.

If NAME and PRODUCT are specified, the specified loaders are only selected if they contain one of the products specified under PRODUCT.

PROD[UCT] = *PRODSAVE / productversion / (productversion, ...)

Defines the products to be used for generation of the loader.

*PRODSAVE means that a list of products is accessed which was specified in a previous statement. RMS remembers this list and uses it again here.

A product or a series of products (max. 10) can be specified.

If more than one product is specified, each one is consulted independently of the others and all the loaders which contain this product are selected.

INSTALL = DEF / ABS / NO
Determines how the value for INSTALL in the loader definition is to be evaluated.

INSTALL = DEF
Installation is carried out depending on the value for INSTALL in the loader definition.

INSTALL = ABS
Installation is always carried out, irrespective of the value for INSTALL in the loader definition.

INSTALL = NO
Installation is not carried out, irrespective of the value for INSTALL in the loader definition.

USERID = :catid:$userid
Determines where the generated loader is to be saved. If no user ID is specified, the loader is saved under the user ID of the RMS depot.

PREFIX = prefix
Specifies the prefix with which the loader file is to be saved.

RUNAME = releaseunitname
Specifies the release unit name for IMON (see the “IMON“ manual [15 (Related publications)]). The name can be up to 30 characters long. The operand must only be specified if MODE=IMON was specified when the start procedure was called.

RUVERS = releaseunitversion
Specifies the release unit version for IMON (see the “IMON“ manual [15 (Related publications)]). The length of version specification must not exceed 8 characters. The operand must only be specified if MODE=IMON was specified when the start procedure was called.

SCI = imoninventoryname
Specifies the software configuration inventory for IMON (see the “IMON“ manual [15 (Related publications)]). The operand must only be specified if MODE=IMON was specified when the start procedure was called.

ADDOPTS = 'optpacktename' / ('optpacketname',...)
see CREATE-REPFILE-DEFINITION statement.

The total number of ADDOPTS operands specified must be <= 9.

OVERWRITE = YES / NO
Option for IMON (see the manual “IMON” [15 (Related publications)]. If the SYSREP file to be generated already exists, it must be cataloged with OVERWRITE = NO as SYSREP.<nnn>, where n = 0...9.

SETTINGS = STD / SWK
The settings to be used during installation.

SETTINGS = STD
The settings from the loader definitions are used (see the CREATE-REPFILE-DEFINITION statement).

SETTINGS = SWK
The settings from the software configurations are used (see the CREATE-SW-CONF statement).

ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)
In the event of an error, the specified job switch is set (ON) or reset (OFF).

12.8.5.2 BUILD-REPFILE - Create and install system loader

The REP file is generated from all the elements selected with SELECT-REPFILE-TO-BUILD.

Format

BUILD-REPFILE
[,ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)]

Operands

ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)
In the event of an error, the specified job switch is set (ON) or reset (OFF).

12.8.6 Transporting correction delivery packets

- INPUT-DELIVERY-PACKET - Accept delivery packet
- MODIFY-DELIVERY-PACKET - Record deviations (notebook)
- CHECK-DELIVERY-PACKET - Check delivery packet
- CREATE-DELIVERY-PACKET - Create delivery packet

12.8.6.1 INPUT-DELIVERY-PACKET - Accept delivery packet

All the REP packets, texts, software configurations and any OPT settings contained in the REP packets are checked and added to the depot if no errors are detected.

At the end, all the system loader definitions are checked. If a default setting is detected, the settings from the new software configuration are used (valid for user groups 2, 3 and 4).

Format

INPUT-DELIVERY-PACKET
INPUT = filename [,ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)]

Operands

INPUT = filename
Defines the file name containing the REPs to be added to the depot.

ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)
In the event of an error, the specified job switch is set (ON) or reset (OFF).

12.8.6.2 MODIFY-DELIVERY-PACKET - Record deviations (notebook)

Preliminary corrections are added, erroneous corrections are invalidated and earlier operations are undone. Each operation is noted in the notebook.

Format

MODIFY-DELIVERY-PACKET
IDENT = 'userid' { { ADD,subno/jd,productversion[,INPUT=filename] / REJ,subno/jd,productversion / RST,subno } ,REASON=’text’ / SIS,INPUT=filename } [,ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)]

Operands

IDENT = 'userid'
Defines the user name under which all deviations are to be logged. The name can be up to 44 characters long.

ADD / REJ / RST ..., REASON = 'text'
A detailed description of these operands can be found in "Correction management - Notebook" section.
“text” can be a maximum of 44 characters long.

> In the statements ADD, REJ and SIS, the slash (/) is part of the input and must be entered between subno and jd!

SIS, INPUT=filename
File containing the REPs to be added to the depot. This option is only available for user group 2.

ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)
In the event of an error, the specified job switch is set (ON) or reset (OFF).

12.8.6.3 CHECK-DELIVERY-PACKET - Check delivery packet

A product version or supply component is checked for completeness.

Format

CHECK-DELIVERY-PACKET
PRODUCT = *PRODSAVE / productversion / lb [,ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)]

Operands

PROD[UCT] = *PRODSAVE / productversion / lb
Product version or supply component to be checked.

*PRODSAVE means that a list of products is accessed which was specified in a previous statement. RMS remembers this list and uses it again here.

ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)
In the event of an error, the specified job switch is set (ON) or reset (OFF).

12.8.6.4 CREATE-DELIVERY-PACKET - Create delivery packet

A delivery packet is created and noted in the depot. This function is available up to the first customer level.

Format

CREATE-DELIVERY-PACKET
PRODUCT = *PRODSAVE / productversion / lb [,TEXT = *ALL / *NONE / 'textname' / ('textname',...)] [,START = *ALL / lb] [,ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)]

Operands

PROD[UCT] = *PRODSAVE / productversion / lb
Product version with the supply component which is to be used for the creation of the delivery packet.

*PRODSAVE means that a list of products is accessed which was specified in a previous statement. RMS remembers this list and uses it again here.

TEXT = *ALL / *NONE / 'textname' / ('textname',...)
Controls the list of the text levels to be included in the delivery. Regardless of the specification, the text level “SC-OPT DESCRIPTION” is always contained.

TEXT = *ALL
All text levels are to be included.

TEXT = *NONE
No text level is to be included.

TEXT = 'textname' / ('textname',...)
The specified text level or the specified list of text levels is to be included.

START = *ALL / lb
Specifies whether all text levels (*ALL) or only the text level (sc) which was introduced with a particular delivery component is to be included.

ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)
In the event of an error, the specified job switch is set (ON) or reset (OFF).

12.8.7 Information functions

- CREATE-MATRIX-LIST - Create matrix list
- COMPARE-REPFILE - Compare REP files
- CREATE-RMS-OPTIONS - Modify RMS settings
- CALL-FUNCTION - Call module

12.8.7.1 CREATE-MATRIX-LIST - Create matrix list

This statement creates a matrix list for one or more products.

Format

CREATE-MATRIX-LIST
ACTUAL / DELIVERED / ORIGINAL ,PRODUCT = *PRODSAVE / productversion / (productversion,...) ,LIST = SUMMARY / DETAIL [,ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)]

Operands

ACTUAL / DELIVERED / ORIGINAL
Determines how to deal with the REPs.

ACTUAL
The REPs are output as they were transmitted to RMS, including notebook operations.

DELIVERED
The REPs which were delivered are output.

ORIGINAL
The REPs are output as they were transmitted to RMS.

PROD[UCT] = *PRODSAVE / productversion
Product version or supply component for which a matrix list is to be created.

*PRODSAVE means that a list of products is accessed which was specified in a previous statement. RMS remembers this list and uses it again here.

LIST = SUMMARY / DETAIL
Determines which information is to be output for each REP and the associated error messages.

LIST = SUMMARY
For each version of a supply component, the following is output:

- the number of REP cards
- the number of error messages
- the number of new, missing, identical or modified error messages
- the number of missing error messages

LIST = DETAIL
In addition to a summary, the development of an error message in successive versions of a supply component is output.

ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)
In the event of an error, the specified job switch is set (ON) or reset (OFF).

12.8.7.2 COMPARE-REPFILE - Compare REP files

Two REP files/products/deliveries are compared and a list of differences is generated.

Format

```
COMPARE-REPFILE

{ { PRODUCT = productversion / lb } /
      { DELIVERY = productversion / lb } /
      { LOADER = 'loadername' [Vnnnn] /
      FILE = filename }
WITH
{ { PRODUCT = { *SAME / productversion } / lb} /
      DELIVERY = {*SAME / productversion } / lb } /
      LOADER = 'loadername' [Vnnnn] /
      FILE = filename }
[WITH [REPS] [LONG INFO]
[,ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)]
```

Operands

PRODUCT = productversion / lb
Defines the product version and supply component which are to be used for comparison.

DELIVERY = productversion / lb
The delivered REPs are used for comparison.

LOADER = 'loadername' Vnnnn
A loader is used for comparison. The loader must be specified along with the version (from 1 ... 9999) in the form V1 ... V9999.

FILE = filename
Name of the REP file (fully-qualified name) which is to be used for comparison.

WITH ...
The first WITH part determines the basis for the comparison (product, delivery, loader or REP file; see above). *SAME means that the same product version which was previously specified is to be consulted for comparison.

WITH [REPS] [LONG INFO]
The second WITH part controls which information is contained in the generated list. By default, the list contains only the REPs changed in the new REP selection. The LONG INFO specification also contains the unchanged REPs. The REPS specification also lists the REP definitions.

ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)
In the event of an error, the specified job switch is set (ON) or reset (OFF).

12.8.7.3 CREATE-RMS-OPTIONS - Modify RMS settings

A log file can be specified and certain settings can be changed for *this* RMS run.

Format

```
CREATE-RMS-OPTIONS
{ { PROTOCOL = { *NONE / filename[/EXTEND] } /
      ALL-OPTS-ON = { NO/ YES } /
      NO-PRODUCT-CHECK = { NO/ YES } /
      NO-TYPIO = { NO/ YES } /
      LARGE= { NO/ YES } /
      BUILD-UNCHANGED= { NO/ YES } /
      PRODSAVE = { NO / YES / CLEAR } }
[,ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)]
```

Operands

PROTOCOL = *NONE / filename / EXTEND
Determines whether a log of the RMS run is to be created and, if so, to which file the log is to be written. If the file name is extended using /EXTEND, the file is extended.

ALL-OPTS-ON = NO / YES
Determines whether all optional corrections - depending on their status, but independent of the OPT-REP setting - are to be added to the system loader.

NO-PRODUCT-CHECK = NO / YES
Determines whether a check of the product version of existing files is to be carried out at installation.

NO-TYPIO = NO / YES
Determines whether a message is to be output at the system console upon successful installation of a loader.

LARGE = NO / YES
Determines whether more than one correction packet is to be added to the depot in one input operation.

BUILD-UNCHANGED = NO / YES
Determines whether a system loader is to be created and installed in a SOLIS run, even when no new corrections have been added to the depot.

PRODSAVE = ON / OFF / CLEAR
Controls the behavior of commands that use the PROD[UCT]=*PRODSAVE operand.

PRODSAVE = ON
Causes RMS to record the names of all products accessed in the following statements, and ensures that this list is used in statements in which the operand PROD[UCT]=*PRODSAVE is specified.

PRODSAVE = OFF
Switches off the recording of products accessed by RMS statements.

PRODSAVE = CLEAR
Deletes the list of products recorded by RMS.

ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)
In the event of an error, the specified job switch is set (ON) or reset (OFF).

12.8.7.4 CALL-FUNCTION - Call module

Every module entered in the file SYSOML.RMS.071 can be called with this statement.

Format

CALL-FUNCTION
modulename [,INPUT = filename] [,ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)]

Operands

modulename
Name of the module to be called from the module library SYSOML.RMS.071.

INPUT = filename
Name of the file containing the input for the called module.

ERROR = ON(<integer 0..31>) / OFF(<integer 0..31>)
In the event of an error, the specified job switch is set (ON) or reset (OFF).

12.8.8 Control statements

- END - End inputs or exit RMS in batch mode

12.8.8.1 END - End inputs or exit RMS in batch mode

The END statement ends inputs to RMS statements via *SYSDTA or file and, if RMS was started with MODE=BATCH, causes the previous statements to be executed.

If END is the last statement in a batch procedure, it exits the RMS program.

Format

END

This statement has no operands.

12.9 Reference information

- Uses of the function keys
- Uses of the P keys
- JD notation
- Naming conventions for input files
- Output files
- Module list

12.9.1 Uses of the function keys

The function keys have the following uses in RMS:

Key	Function
K1	Goes back one logical step.
K2	BREAK.
K3	New screen set-up with same contents.
K4	
K5 or ESC+ W	Text output changes from German to English or vice versa.
K6	
K7	
K8	
K9 or ESC+ ?	HELP function - online documentation on the screen, can be accessed from any screen, then return to current processing.
K10	
K11 or ESC+ =	The sorting order for the selection screens can be changed from increasing to decreasing order and vice versa.
K12 or ESC+ <	Interactive information is accessed and the current function is interrupted. Return with K1 key.
K13	
K14 or ESC+ :	exits the current function or program.
K15	
F1	This key is function-specific and is described on each screen.
F2	This key is function-specific and is described on each screen.
F3	This key is function-specific and is described on each screen.
F4 or ESC+ ^	This key is function-specific and is described on each screen.

12.9.2 Uses of the P keys

To simplify function selection, the P keys can be loaded and used as desired:

P1 ---> Selection function 1

. . .

P12 --->Selection function 12

P13 --->Goes to the last selectable field

P18 --->Hard copy of the entire screen (if a printer connection is available)

12.9.3 JD notation

* in front of JD	This correction was added to the depot from an external file using the notebook operation ADD.
Blank in front of JD	This correction was added to the depot using the input function.
- after JD	This correction is no longer valid for supply components as a result of an REJ notebook operation.
+ after JD	This correction is valid for all supply components as a result of an ADD notebook operation.
Blank after JD	No notebook operation. This correction is valid for the supply components with which it was input.

12.9.4 Naming conventions for input files

The names of the input files conform to BS2000 conventions. The file name can be entered in wildcard syntax in interactive mode for all screens in which an input file is requested. This syntax conforms to the SDF data type *filename* with the suffix *with-wild* (see SDF metasyntax in the “Commands” manual [1 (Related publications)]).

If there is more than one file which corresponds to the specified selection, an overview is output. The desired file is to be selected in the overview.

If only one file matches the criteria, this file is automatically processed.

12.9.5 Output files

The following functions create files with the following names:

Function	File name
DIALOG-INFORMATION, output text contents	TXT.subno.text-levels-names
DIALOG-INFORMATION, output correction contents	REP.subno.jd.product.version
EXPORT - Products/loader definitions	RMS.DEPOT.yymmdd.hhmmss
EXPORT - Text level	RMS.TEXTLEVEL.name
LOADER GENERATION - Install	installationname
LOADER GENERATION - Generate	systemloader-definition.version
CREATE DELIVERIES	SYSRMS.product.version.sc-vers
CREATE DELIVERIES, for missing texts	TEXTNO.ERROR.product.version.sc-vers
CREATE DELIVERIES, original delivery	DELIV.product.version.sc-vers.ORG
NOTEBOOK LIST, selected products	NBK.ALL.ENTRIES.product.vers
COMPARE - Create up-to-date REP packet	product.version.sc-vers
CORRECTION-GLOBAL LIST (MATRIX)	#MATRIX.LIST.hhmmss
DELIVERIES - Accept	INPUT.LISTE.hhmmss
NOTEBOOK LIST - Global list	#NOTEBOOK.LIST.hhmmss
NOTEBOOK LIST selected correction number	NBK.subno.jd.produCt.vers
COMPARE - Packet with packet	#hh-mm-ss.LSTVGL

12.9.6 Module list

Central modules

RMS	Contains basic routines, interactive information function, Pascal-XT runtime system and FHS runtime system.
RMSCONV	Routine to convert a depot from V3.20 to the current version.
RMSHELP	Contains background information about screens, in German.
RMSHELPE	Contains background information about screens, in English.
RMSINIT	Executes all initialization tasks and calls RMSCONV or RMSIN2 ifnecessary.
RMSIN2	Is called if a depot update (input, notebook records or import) was not properly ended. The status of the function before call-up is restored.
RMSMONT	Contains all loader building routines.
RMSSYN	Contains all batch/SOLIS2-specific routines.
RMS7SYN	Contains all batch/SOLIS2-specific routines for the V7 syntax.

Function modules which require MODE MEISTER

EINFAHR	Inputs SOLIS2 or PULS REP packets.
LIEFERN	Generates supply components for SOLIS2.
LIEFER7	Generates supply components for SOLIS2. (new function).
MATRIX	Creates cross-reference list of corrections.
MONTAGE	Generates/installs all system loaders to be built.
NOTIZERF	Notebook records.
NBKINFO	Outputs notebooks information.
NOTBLIST	Global list of notebook entries.
RMSEDIT	Sets up new text levels, sets up, modifies and deletes text entries in text levels or product notes.

RMSEXIM	Export (plus deletion, where applicable) of packets from depot. Packets are products, loader definitions, OPT-REP settings, loader modifications and text levels. All the information is saved in a new depot. *Importing packets from another depot* For products, it is possible • to accept everything • to accept everything starting with the last input and delivered supply component • to accept all input supply components and the newest delivery correction version
RMSFIND	Character-oriented search of depot, module-oriented search of all corrections.
RMSLGEN	Connection to RMSMONT for loader selection
RMSLVWIN	Connection to RMSMONT for loader administration and information.
RMSOPTS	Setting options for this run.
RMSST00	Outputs statistical information on the depot contents.
VGL	Compare REPs - new format.
VGLSTART	Compare REPs.

Service program connection modules

These modules contain only the connection for the service program. The user must have installed the corresponding library.

CALLEDOR	Calls EDOR.
CALLEDT	Calls EDT as a function of the menu parameter (TEMPORARY or RESIDENT).
CALLLMS	Calls LMS.

Special functions modules

DEFKUDEP	Converts a service depot to a customer depot.
RMSCNGPR	Changes a product version (upon customer release).
RMSQM001	Creates an OPT-REP packet from a definition file.

12.10 Statements of RMS before V7.0

Interactive function		Operation function
DIALOG-INFORMATION		No
FUNCTIONAL INFORMATION and OVERVIEWS		1
CORRECTION MANAGEMENT		
	Accept preliminary corrections	2
	Activate depot corrections	2
	Delete corrections	2
	Reset notebook operations	2
	Information options	2
LOADER MANAGEMENT		
	Define system loader	3
	OPT-REP setting	4
	Automatic OPT-REP setting	No
	Local loader modification	4
	Modify and delete definitions	No
LOADER GENERATION and INSTALLATION		
	Loader generation	Yes
	Generate old loaders	No
	Loader installation	Yes
GENERATE DELIVERIES		
	Generate current deliveries	Yes
	Generate original deliveries	Yes
	Supply component version number	5
ACCEPT DELIVERIES		Yes
EDITING		
	Text levels	6

	Product notes	6
	Correction documentation	6
	Text editor	No
INSTALLATION and DEPOT MANAGEMENT		
	EXPORT function	No
	IMPORT function	No

Explanation:

Yes	This can be used as a statement in any situation.
No	This function can be only be used in interactive mode.
1	With these statement functions, special information and overview lists are output.
2	The operations input in interactive mode are taken into account in all statements.
3	The settings defined in interactive mode are taken into account by the statement SELECT-REPFILE-TO-BUILD. If no definitions are available and the parameter AUTODEF=YES was entered, a default definition is set up automatically.
4	The settings defined in interactive mode are taken into account by the function SELECT-REPFILE-TO-BUILD.
5	In the user groups 2 and 3, the version number of a delivery is assigned automatically in interactive and statement mode.
6	The acceptance and delivery of text levels are possible in statement mode.

12.10.1 Examples

The functions are also possible in interactive mode (line mode). The default startup procedure must be called up as follows:

```
/CALL-PROC SYSPRC.RMS.071,(MODE=BATCH)
```

By means of coupling individual statements, it is possible to create jobs which automatically execute certain applications.

Example 1

All the data in `DELIVERYFILE` is added to the depot. Notebook entries are taken into account.

The specified `LISTFILE` is extended to include the log file.

In the event of an error, job switch 30 is set to ON.

```
FUNCTION=INPUT,-
  INPUT=DELIVERY FILE,-
  LIST=LISTFILE/EXT,-
  ERROR=ON(30)
```

For all products which are added to the depot, utilizable loaders are generated. Existing system loader definitions are taken into account.

If definitions are missing, default definitions are used and entered.

In the event of an error, job switch 31 is set to ON.

```
FUNCTION=SELECT-REPFILE-TO-BUILD,-
  INSTALLATION=DEF,-
  AUTODEF=YES,-
  ERROR=ON(31) END
```

Example 2

The loaders are generated as defined in the system loader definitions with the names `S090REP ZE1` and `S090REP ZE2`.

```
FUNCTION=SELECT-REPFILE-TO-BUILD,-
  NAME='S090REP ZE1',-
  INSTALLATION=DEF
FUNCTION=SELECT-REPFILE-TO-BUILD,-
  NAME='S090REP ZE2',-
  INSTALLATION=DEF
```

For all system loader definitions in which SPOOL V4.9A is defined, loaders are generated and, where applicable, installed.

For missing definitions, a default definition is used and entered (AUTODEF=YES).

```
FUNCTION=SELECT-REPFILE-TO-BUILD,-
  PROD=SPOOL,VERS=4.9A,-
  AUTODEF=YES,-
  INSTALLATION=DEF
END
```

12.10.2 Description of the statements

General definitions

pathname	[:catid:][$userid.]filename
prodname	Product family name from PULS (up to 15 characters)
prodvers	Product version, without V and without the leading 0, but with entry of the full stop.
loaderdefname	Name of the system loader definition (up to 22 characters)

RMS-PARAMETER statement (V7.0 and earlier: CREATE-RMS-OPTIONS)

In the log file, the specified statements and output lists of the called functions are written and made ready for printing.

FUNKTION=RMS-PARAMETER
,PROT[OKOLL]= { *NONE / filename }[/EXT[END]

PROTOKOLL
A log file with the name `filename` is to be created.

EXTEND
The file is to be updated rather than created.

***NONE**
Closes the open log file.

EINFAHR statement (V7.0 and earlier: INPUT-DELIVERY-PACKET)

FUNKTION=EINFAHR
,{ DATEI / INPUT / FILE }=pathname[,L[IST]=pathname[/EXT[END]]]

LIST
A list describing the input operation and separate from the log file is to be created.

EXTEND
The file is to be updated rather than created.

LIEFER statement (V7.0 and earlier: CREATE-DELIVERY-PACKET)

FUNKTION=LIEFER
[,PROD=prodname,VERS=prodvers [,LB={ * / - / libvers }]][,{ DATEI / OUTPUT / FILE }={ *STD / *SAME / filename }] [,TEXT= { *NONE / 'textname' } =,UMFANG= { ALL / lbvers0 }]

LADERBAU statement (V7.0 and earlier: SELECT-REPFILE-TO-BUILD)

FUNKTION=LADERBAU
[,PROD=proname,VERS=prodvers][,NAME='loaderdefname'] [,AUTODEF=YES[,AUTOOPT='optpacketname']] [,INSTALL[ATION]={ ABS / DEF / NO }][,USERID= { *OWN / userid }][,PRE[FIX]= { *NONE / prefix }]

BAULADER statement (V7.0 and earlier: BUILD-REPFILE)

FUNKTION=BAULADER

ENDE statement (V7.0 and earlier: END)

E[NDE] / H[ALT]

13 SIR Creation of pubsets

Version: **SIR V21.0A** (<ver>=210)

Privileges: **STD-PROCESSING** (for nonprivileged functions)

TSOS (for privileged functions))

The utility routine SIR (System Install and Restore) is used to create a pubset.

In addition, SIR offers facilities for initializing and, if necessary, formatting disks (VOLIN utility routine).

Product structure

The product structure of SIR is made up of two components. The range of functions available to the user is defined by the TSOS privilege.

- If the SIR utility is running under the user ID TSOS, the full range of functions, including those for creating and extending pubsets as well as software installation functions, is available to the user.
  This functional scope is referred to as “privileged SIR”.
- If the SIR utility is **not** running under the user ID TSOS, only the functions for copying files between pubsets and private disks are available to the user.
  This functional scope is referred to as “nonprivileged SIR”.

Functions

The most important functions for the privileged SIR user are:

- initializing and formatting disks (complete VOLIN function)
  This function is described starting on "Volume sets".
- creating any required volume sets
- extending existing volume sets
- creating any required pubsets
- extending an existing pubset
- creating the home pubset when installing BS2000
- copying files between pubsets and private disks

The SIR utility routine allows you to create home pubsets for operation under the current version of BS2000 (target version), and to upgrade to the current version of BS2000 (target version) from various execution versions of SIR.

A SIR version is generally coupled with a particular version of BS2000. This means the following: execution version <= target version.

SIR V21.0 can be used to create a home pubset for BS2000 V21.0 and to upgrade to BS2000 V21.0 (target version) from the **execution versions BS2000 OSD/BC V10.0, V11.0 and BS2000 V21.0** of SIR V21.0.

With SIR V21.0 it is possible to create a (further) home pubset for BS2000 on x86 servers.

13.1 Creating pubsets with SIR

A **single-feature pubset** (SF pubset) has a pubset ID (syntactically identical to a catalog ID) and consists of a maximum of 255 disks. The format of all the disks must be the same in relation to allocation unit, formatting, PAM key use etc. Certain attributes are assigned to the SR pubset, e.g. high availability, which apply to all the files within the pubset.

Privileged SIR can also be used to set up a **system-managed pubset** (SM pubset) or a volume set. As far as the user is concerned, a system-managed pubset (SM pubset) is virtually identical to a SF pubset. In the main, the differences between the two pubset types are relevant to system administration.

By contrast with the SF pubset, an SM pubset consists of several volume sets. Like an SF pubset, each volume set has a certain format and certain attributes. The formats and attributes of the various volume sets that belong to an SM pubset can be different. Like an SF pubset, the SM pubset is addressed from outside only via a catalog identifier. The specification of specific file attributes defines the volume set in which a file is stored, with the result that files with different requirements in terms of availability and performance can be stored in one SM pubset.

SM pubsets cannot be used as a home pubset. All SIR statements which are exclusively needed to create a home pubset are therefore rejected for SM pubsets.

Dynamic pubset reconfiguration

The pubset configuration defined during creation can be changed (or reconfigured) during the life of the pubset.

Configuration data, the disk and the volume set configuration of a pubset can be reconfigured via various command interfaces dynamically, i.e. while the pubset is in use. For detailed information on dynamic pubset reconfiguration, see the “Introduction to System Administration” [5 (Related publications)].

13.1.1 Volume sets

Set up a volume set

First the disks for the volume set are initialized or formatted (VOLIN function) and entered in the Volres (system disk of the volume set). Then a catalog extent is set up on the Volres.

The following statements are available for the “Set up a volume set” function:

```
BEGIN-VOLUME-SET-DECLARATION ...,ACTION=*INSTALL
CREATE-VOLUME ...
CREATE-CATALOG 0,nnnn
END-VOLUME-SET-DECLARATION
```

The BEGIN-VOLUME-SET-DECLARATION statement starts the declaration of the volume set to be set up. The declaration ends with END-VOLUME-SET-DECLARATION. If several volume sets are to be set up in one SIR run, this statement block can be repeated for each volume set.

The CREATE-VOLUME statement must be specified for every disk that belongs to the volume set so that they can be stored in the SVL of the Volres. A CREATE-CATALOG statement can be specified for the Volres. Without the statement, a file catalog with
2000 PAM pages is created by default and its size is rounded up to the next multiple of the allocation unit size.

The function supports the setting up of volume sets that are used as dual recording and also of volume sets with disks of block size 4K.

The setup and extension of “free” volume sets, i.e. volume sets that do not belong to an SM pubset, is needed primarily for pubset reconfiguration.

Example

```
/START-SIR
//BEGIN-VOLUME-SET-DECLARATION VOLUME-SET=AB,
        ACTION=*INSTALL(FORMAT=*NK(*4K(8)),AVAILABILITY=*DRV)              1.
//CREATE-VOLUME DISK-NUMBER=000(OVERWRITE-DISK=*ANY,UNIT=MA(SUBUNIT=MB)),
        FORMAT=*YES                                                        2.
//CREATE-CATALOG DISK-NUMBER=000,FILE-SIZE=160                             3.
//END-VOLUME-SET-DECLARATION                                               4.
//BEGIN-VOLUME-SET-DECLARATION VOLUME-SET=X,
        ACTION=*INSTALL(FORMAT=*K,AVAILABILITY=*NORMAL)                    5.
//CREATE-VOLUME DISK-NUMBER=*RANGE(00,04,OVERWRITE-DISK=*ANY,
        UNIT=(MN00,MN01,MN02,MN03,MN04)),
        FORMAT=*NO                                                         6.
//CREATE-CATALOG DISK-NUMBER=00,FILE-SIZE=500                              7.
//END-VOLUME-SET-DECLARATION                                               8.
//END
```

1. A volume set with the ID AB, disk format NK4 and an allocation unit of 8 KB is to be set up. The volume set is to have DRV capability. Its logical properties (e.g. performance and availability) are defined when the volume set is assigned to a pubset. Before a volume set is defined as a DRV volume set with SIR, it must be entered in DRV with the command SET-DRV-PARAMETER UNIT=*VOLUME-SET(AB).
2. The first volume in the volume set AB is initialized with the VSN AB.000 and formatted without a prior VSN check.

3. A catalog section is created for the initialized and formatted volume 000. It has a size of 160 PAM pages.
4. The declaration for volume set AB is terminated. Such a BEGIN-/END statement block can be repeated as often as required in one SIR run.
5. An additional volume set is to be set up with the ID X and volume format K (i.e. with an allocation unit of 6 KB). The volume set is to be set up without a mirror volume (normal availability).
6. Five volumes are initialized for volume set X without a prior VSN check. These volumes are assigned the VSNs PUBX.00 through PUBX.04.
7. A catalog section is created for the initialized and formatted volume 00. It has a size of 500 PAM pages.
8. The declaration for volume set X is terminated.

Extending a volume set

A “free” volume set, i.e. one that does not yet belong to an SM pubset, can be extended by additional disks.

The following statements are available for the “Extend a volume set” function:

```
BEGIN-VOLUME-SET-DECLARATION ...,ACTION=*EXTEND
CREATE-VOLUME ...
END-VOLUME-SET-DECLARATION
```

The CREATE-VOLUME statement must be specified for each disk added to the volume set so that they can be stored in the SVL of the Volres.

The CREATE-CATALOG statement is not permitted for this function.

A volume set that already belongs to an SM pubset can be extended with the “Extend an SM pubset” function (see "SM pubsets").

Example

```
/START-SIR
//BEGIN-VOLUME-SET-DECLARATION VOLUME-SET=AB,ACTION=*EXTEND          1.
//CREATE-VOLUME DISK-NUMBER=*RANGE(001,002,OVERWRITE-DISK=*ANY,
        UNIT=(MP(SUBUNIT=MQ),MR(SUBUNIT=MS))),
        FORMAT=*YES                                                 2.
//END-VOLUME-SET-DECLARATION
//END
```

1. The volume set with ID AB is to be extended.
2. A second and third volume with VSN AB.001 and AB.002 now also belong to this volume set.

13.1.2 SM pubsets

Setting up an SM pubset

This function is used to put together volume sets to make up an SM pubset.

Set up is performed in two internal SIR stages:

1. First the disks are initialized and formatted for each volume set that is to be set up in the same SIR run. After that, the related disks are entered in the SVLs of the Volres of the individual volume, thus forming the volume sets. Then a file catalog is created on the Volres. On the control volume set, which is the preferred volume set of the SM pubset, further special files are created.
2. An SM pubset is set up from the volume sets created during stage one, or from volume sets that were previously created in earlier SIR runs.

   The volume set that is to become the control volume set must always be set up afresh in the same SIR run because its structure is different from that of a simple volume set.

The following statements are available for the “Set up an SM pubset” function:

```
DECLARE-PUBSET PUBSET-TYPE=*S-M(ACTION=*INSTALL)
BEGIN-VOLUME-SET-DECLARATION ...,ACTION=*ADD / *INSTALL
CREATE-VOLUME ...
CREATE-CATALOG ...
 ...
END-VOLUME-SET-DECLARATION
BEGIN-VOLUME-SET-DECLARATION ...,ACTION=*ADD / *INSTALL
 ...
END-VOLUME-SET-DECLARATION
COPY ...
 ...
END
```

The same conditions as for setting up a volume set apply to the declaration of the individual volume sets when setting up an SM pubset. However, when setting up an SM pubset, catalog extents can be set up for individual volume sets on disks other than Volres.
If no CREATE-CATALOG statement is given for a volume set that is to be set up, a catalog of at least 2000 PAM pages in size, rounded up to the next multiple of the allocation unit size, is automatically set up on the Volres.

In the declaration of a control volume set, ACTION=*INSTALL must be specified in the BEGIN-VOLUME-SET-DECLARATION statement.

COPY statements are only permitted outside the statement block BEGIN-/END-VOLUME-SET-DECLARATION. The position of the files to be copied within the SM pubset is defined by the performance attribute of the various source files.

The following statements are not permitted when setting up an SM pubset since it is not permitted to use an SM pubset as a home pubset:

`CREATE-SNAP-FILE, CREATE-IPL-VOLUME, MODIFY-IPL-VOLUME`

Example

```
/START-SIR
//DECLARE-PUBSET PUBSET-TYPE=*SYSTEM-MANAGED(PUBSET=SMP1,
        ACTION=*INSTALL(CONTROL-VOLUME-SET=XYZ))                          1.
//BEGIN-VOLUME-SET-DECLARATION VOLUME-SET=AB,
        ACTION=*ADD(VOLUME-SET-ATTR=*PAR(
          USAGE=*STD,AVAILABILITY=*HIGH,PERFORMANCE-ATTR=*PAR(
            PERFORMANCE=*HIGH,WRITE-CONSISTENCY=*IMMEDIATE)))             2.
//END-VOLUME-SET-DECLARATION
//BEGIN-VOLUME-SET-DECLARATION VOLUME-SET=XYZ,
        ACTION=*INSTALL(FORMAT=*K,AVAILABILITY=*DRV)                      3.
//CREATE-VOLUME DISK-NUMBER=00(OVERWRITE-DISK=*ANY,UNIT=XY(SUBUNIT=XZ)),
        FORMAT=*YES                                                       4.
//CREATE-CATALOG DISK-NUMBER=00,FILE-SIZE=96                              4.
//END-VOLUME-SET-DECLARATION
//END
```

1. An SM pubset with the catalog ID SMP1 is defined. The volume set with the ID XYZ is designated as the control volume set.
2. The volume set with the ID AB, which was defined in a prior SIR run, is to be included in the SM pubset with ID SMP1. The following logical properties are defined for it: it is to be used as the storage location for standard files and provide high failure security and performance. The consistency for writes is to be immediate.
3. Subsequently, a volume set with the ID XYZ is to be set up. It is known from (1) that this volume set is to be the control volume set. The volume set is to have DRV capability.
4. A volume is defined (VSN XYZ.00) and a range is set for the catalog.

Extending an SM pubset

An SM pubset or volume sets that belong to an SM pubset can be extended by volume sets or disks.

An SM pubset is extended SIR-internally in two stages in the same way as described under “Setting up an SM pubset”:

1. The specified volume sets are extended by disks as defined in the CREATE-VOLUME statement. The new disks are first initialized and formatted and then entered in the Volres of the individual volume sets. The creation of catalog extents is not supported.
2. In the second stage, additional volume sets are included in the SM pubset and the COPY statement is executed.

The following statements are available for the “Extend an SM pubset” function:

```
DECLARE-PUBSET PUBSET-TYPE=*S-M(ACTION=*EXTEND)
BEGIN-VOLUME-SET-DECLARATION ...,ACTION=*ADD / *INSTALL / *EXTEND
CREATE-VOLUME ...
 ...
END-VOLUME-SET-DECLARATION
BEGIN-VOLUME-SET-DECLARATION ...,ACTION=*ADD / *INSTALL / *EXTEND
 ...
END-VOLUME-SET-DECLARATION
COPY ...
 ...
END
```

Notes

- The pubset must not contain volume sets that are to be processed with ACTION= *ADD / *INSTALL.
- Volume sets that are to be processed with ACTION=*ADD must not belong to any other pubset.
- For ACTION=*EXTEND, the volume set must already be part of the pubset to be processed.
- A volume set can only be extended if it is in a consistent state.

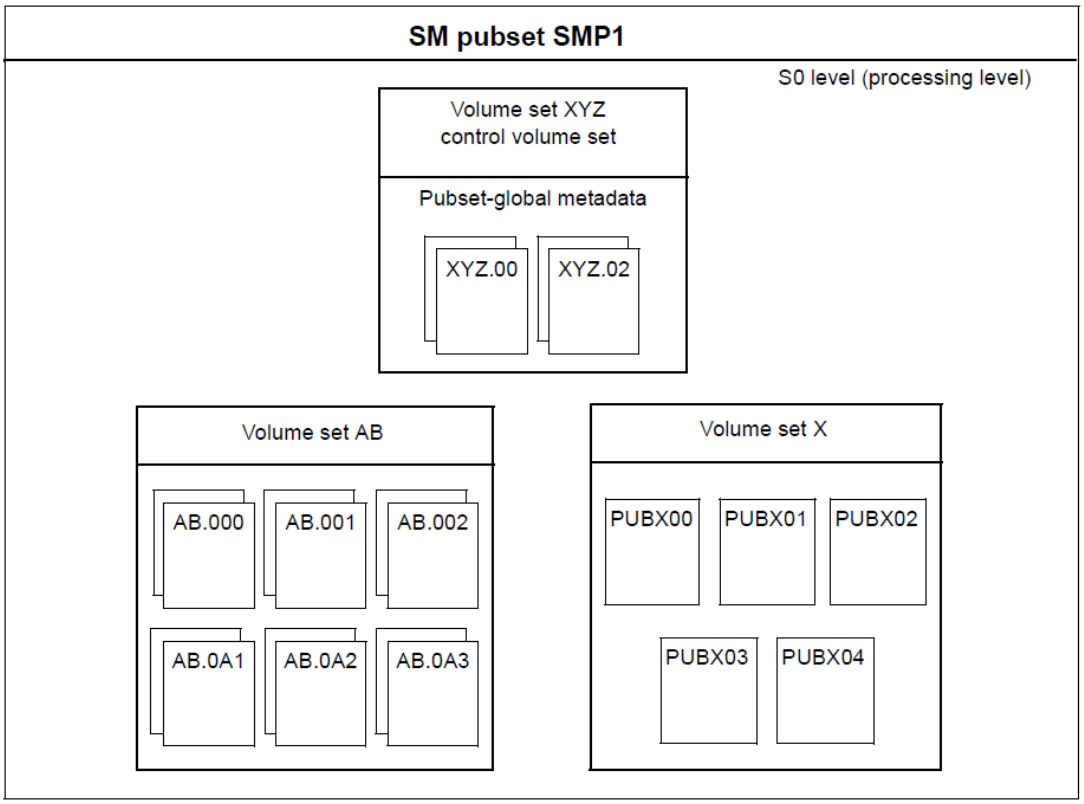
Example

```
/START-SIR
//DECLARE-PUBSET PUBSET-TYPE=*SYSTEM-MANAGED(PUBSET=SMP1,
        ACTION=*EXTEND(CONTROL-VOLUME-SET=XYZ))                          1.
//BEGIN-VOLUME-SET-DECLARATION VOLUME-SET=X,
        ACTION=*ADD(VOLUME-SET-ATTR=*PAR(PERFORMANCE-ATTR=*PAR(
          PERFORMANCE=*VERY-HIGH,WRITE-CONSISTENCY=*BY-CLOSE)))          2.
//END-VOLUME-SET-DECLARATION
//BEGIN-VOLUME-SET-DECLARATION VOLUME-SET=AB,
        ACTION=*EXTEND                                                   3.
//CREATE-VOLUME DISK-NUMBER=*RANGE(0A1,0A3,OVERWRITE-DISK=*ANY,
        UNIT=(MT(SUBUNIT=MU),MV(SUBUNIT=MW),MX(SUBUNIT=MY))),
        FORMAT=*YES                                                      4.
//END-VOLUME-SET-DECLARATION
//BEGIN-VOLUME-SET-DECLARATION VOLUME-SET=XYZ,
        ACTION=*EXTEND                                                   5.
//CREATE-VOLUME DISK-NUMBER=02(OVERWRITE-DISK=*ANY,UNIT=Y1(SUBUNIT=Y2)),
        FORMAT=*YES                                                      6.
//END-VOLUME-SET-DECLARATION
:
//COPY FILE=:SF01:SYSMES.(NEW=SYSMES.)                                   7.
//COPY FILE=ARCHIV.(NEW=*PREFIX(REMOVE=ARCHIV)),FROM=*PUBSET             8.
//END
```

1. The SM pubset SMP1 is to be extended.
2. The previously defined volume set X (see "Volume sets") is to be included in the SM pubset. Its logical properties are defined as follows: it is to be used for storing standard files and provide normal failure security with increased performance. The consistency for writes is to be after file closure.
3. The volume set AB is to be extended.
4. A fourth, fifth and sixth volume now also belong to this volume set with the VSNs AB.0A1, AB.0A2 and AB.0A3.
5. The control volume set XYZ is to be extended.

6. A second volume now also belongs to this volume set with the VSN XYZ.02.
7. All files from pubset SF01 whose names begin with “SYSMES” retain their file names and are copied to volumes of SM pubset SMP1 which was specified in the DECLARE-PUBSET statement (1).
8. All files in the default pubset whose names begin with “ARCHIV” are to be copied to SMP1. The “ARCHIV” prefix is to be omitted for the target files.

SM pubset SMP1 configuration after processing the examples



A pubset generated by SIR provides system administration with the basis for further management measures. Their order and chronological sequence is not fixed. They do not have to be carried out during pubset creation, but can be made as required during pubset maintenance. It is recommended to modify the pubset immediately after generation if the defaults which cannot be influenced during generation (e.g. usage restrictions, cache assignments, saturation thresholds) are to be changed. It is also thereby possible to generate the part of the pubset environment that was not taken into account during generation (storage classes, management classes, GUARDS profile, HSMS environment, etc.). The pubset must be in operation for this post-processing. A detailed description of post-processing newly generated SM pubsets can be found in the “SMS” manual [8 (Related publications)].

13.1.3 SF pubsets

Setting up an SF pubset

An SF pubset (single-feature pubset) is set up in the same way as a pubset has always been set up. In other words, the disks are initialized with the VSN, the archive numbers and device types of the disks in the pubset are anchored in the SVL of the system disk of the pubsets (pubres) and a file catalog is defined for the pubset.

The following statements are available for the “Set up an SF pubset” function:

```
DECLARE-PUBSET PUBSET-TYPE=*S-F(...,ACTION=*INSTALL)
CREATE-VOLUME
CREATE-CATALOG
CREATE-PAGING-FILE
```

If no CREATE-CATALOG statement is specified, a file catalog with a size of at least 2000 PAM pages is created by default on the Pubres and rounded up to the next multiple of the allocation unit size.

The CREATE-VOLUME statement must be specified for each disk of the pubset to be set up because all disks must be anchored in the SVL of the Pubres.

The function supports the setting up of DRV pubsets and of pubsets with disks with a physical block size of 4 Kbytes.

Attempts to set up an IPL disk with a physical block size of 4K will be rejected.

Extending an SF pubset

Additional disks can be included in an SF pubset that has already been set up. If the pubset is imported then this extension is performed dynamically, i.e. the pubset does not have to be exported first.

The following statements are available for the “Extend an SF pubset” function:

```
DECLARE-PUBSET PUBSET-TYPE=*S-F(...,ACTION=*EXTEND)
CREATE-VOLUME
CREATE-PAGING-FILE
COPY                        (only necessary when extending the home pubset)
CREATE-IPL-VOLUME                          - " -
MODIFY-IPL-VOLUME                          - " -
CREATE-SNAP-FILE                           - " -
```

A CREATE-VOLUME statement must be specified for each disk to be added to the pubset (see section “SM pubsets”). The disk number specified must not already exist in the pubset.

The disks added to a pubset must have the same disk formats (K/NK, NK2/NK4) as the ones already in the pubset. The allocation unit must also be homogeneous within a pubset. The volumes are therefore automatically initialized by SIR.

Creating a home pubset

If a BS2000 operating system is to be installed and its home pubset is to be created, all files required for the operating system (BS2000 V21.0 and the associated system-related software products) must be transferred to the pubset. At least one disk must be turned into an IPL disk (see next section below). A paging file of sufficient size (>= 200 MB) must be set up for system initialization.

Creation of a snapshot file is recommended when the SNAP dump is already to be available after system initialization. See the SNAP-ACTIVE-SWITCH parameter in the startup parameter service, “Introduction to System Administration” [5].

The following statements are available for the creation of a home pubset.

```
DECLARE-PUBSET PUBSET-TYPE=*S-F(...,ACTION=*INSTALL)
CREATE-VOLUME
CREATE-CATALOG
COPY
CREATE-IPL-VOLUME
CREATE-PAGING-FILE
CREATE-SNAP-FILE
```

The conditions described in section “SM pubsets” apply here, too.

Setting up an IPL disk

With the aid of SIR, public or private disks can be made bootstrappable, i.e. turned into IPL disks.

Public disks of an existing SF pubset can be set up as the IPL disks of the new BS2000 version.
When creating a new SF pubset, several IPL disks can be set up for different BS2000 versions, where the corresponding SIR version must be used for versions other than the current version of BS2000. Only one operating system version can ever be loaded from an IPL disk. Several IPL disks of the same version can also be set up.
Disks of a pubset with the attribute `LARGE-FILES-ALLOWED=*YES` can be made bootstrappable as well. Disks of a pubset with the standard SYS ID “250” **cannot** be made bootstrappable.

A disk is bootstrappable if the operator can use it to initiate the operating session (startup). For this to be possible, the disk must have the following attributes:

- It must contain
  the IPL module (SYSPRG.IPL.DSKxxx),

  as well as the save files

  BOOTSAVE (SYSPRG.BOOT.DSKxxx.SAVE),
  SLEDSAVE (SYSPRG.SLED.DSKxxx.SAVE) and
  the startup configuration data (SYSDAT.IPL-CONF.DSKxxx)

  and, if required, the

  IPL REP file (SYSREP.IPL.DSKxxx) and
  SLED REP file (SYSREP.SLED.DSKxxx)

  If the REP files for IPL and SLED are not present, a message is issued during anchoring and processing is continued.

  <.DSKxxx> is replaced by <.vsn> for private disks.
- The standard volume label (SVL) contains pointers to these files.

The BOOTSAVE, SLEDSAVE and IPL-CONF files are created as part of the “Set up an IPL disk” function. These files are anchored in the boot or SVL block of the disk.

These files must not reside on the disk to be turned into an IPL disk before SIR is called. By means of internal copying, SIR ensures that this condition is met when anchoring the IPL files in the SVL:
SIR copies the output files to the destination disk under a disk-specific name. The output files are neither anchored nor modified.
The files in question are the following: SYSPRG/SKMPRG.IPL.<ver>, SYSREP.IPL.<ver> and SYSREP.SLED.<ver>

The anchored files are not taken into account during a logical backup, but their output files are.

The following statements are available for the “Set up an IPL disk” function:

```
DECLARE-PUBSET PUBSET-TYPE=*S-F(...,ACTION=*IPL)
CREATE-IPL-VOLUME
MODIFY-IPL-VOLUME
       (may be additionally or alternatively added)
```

During the run, all CREATE-IPL-VOLUME statements are executed by the MODIFY-IPL-VOLUME statements.

If only private disks are to be turned into IPL disks, no DECLARE-PUBSET or BEGIN-VOLUME-SET-DECLARATION statement may be issued.

If the IPL files do not reside on the pubset, they must first be transcribed there. The COPY statement must then be added to the statements specified above.

For private disks, the IPL files in the home pubset must be in a format that is compatible with the private disk (e.g. usually NK format).

You should bear in mind when setting up the IPL disk: Modifying the output files does not affect the anchored files. If the anchored files are to be modified, SIR must be called again.

All other files required for BS2000 system initialization (startup files, see the “Introductory Guide to System Support” [5 (Related publications)]) are sought in the catalog. They can be located on any of the home pubset disks, so do not necessarily have to be contained on the IPL disk.

Modifying an IPL disk

This function modifies an IPL disk. Unlike when setting up the IPL disk, the specified source files (IPL, IPL-Rep, SLED-Rep) may have names deviating from the standard. The files are then copied to the IPL disk under a disk-specific name and anchored afresh in the boot/SVL block. When IPL is anchored, the BOOTSAVE, SLEDSAVE and IPL-CONF files are created anew. The anchors for the remaining files that are required for an IPL disk remain unchanged.

If all the necessary files are specified, this function is the same as the function “setting up an IPL disk”. In this case the disk concerned need not have been bootstrappable.

The following statements are available for the “Modifying an IPL disk” function:

```
DECLARE-PUBSET PUBSET-TYPE=*S-F(...,ACTION=*IPL)
MODIFY-IPL-VOLUME
```

The notes under “Setting up an IPL disk” also apply to the function “Modifying an IPL disk”.

13.2 Pubset states

The functions of the SIR utility routine create and modify pubsets and thereby also their states. In this case, you must bear in mind a number of accompanying conditions in order to avoid execution problems. This section outlines the effects of the SIR actions on the pubsets involved.

The ACTION operand of the DECLARE-PUBSET statement defines the processing mode of a pubset.

ACTION=*INSTALL (pubsets)

- The desired pubset ID may already have been entered in the MRS catalog of the home pubset. In this case, the device type is checked. If an inconsistency is detected, creation is aborted. If an MRS catalog entry is missing, it is added by SIR.
- If the pubset is to be created in the existing BS2000 environment, no pubset with the same pubset ID may have been imported at execution time.
- If new volume sets are created or existing volume sets added while an SM pubset is being set up, the volume set must not be used at that time or entered in the MRS catalog.
- SIR always executes an /IMPORT-PUBSET when setting up an SM pubset, and when setting up an SF pubset if any of the following statements were specified:

```
CREATE-CATALOG DISK-NUMBER not equal 00
CREATE-PAGING-FILE
CREATE-SNAP-FILE
COPY
CREATE-IPL-VOLUME
MODIFY-IPL-VOLUME
```

ACTION=*EXTEND (pubsets)

- The pubset must have been entered in the MRS catalog of the home pubset.
- If the pubset has already been imported, it should have been imported exclusively. Although it is possible to extend shared pubsets, it will only work for certain if the local system is the only sharer.
- If the pubset has not been imported yet, it must be possible to import it using /IMPORT-PUBSET ACTUAL-JOIN=*STD. To be able to do that, an import with ACTUAL-JOIN=*FIRST must have taken place (see the corresponding note under ACTION=*INSTALL).
- If volume sets are set up or added while an SM pubset is being extended, the same conditions apply as for setting up an SM pubset.

ACTION=*IPL (SF pubset only)

- The pubset must have been entered in the MRS catalog of the home pubset.
- The pubset must have been either imported or exported in its entirety. Intermediate states such as may arise in the course of execution of /IMPORT-PUBSET or /EXPORT-PUBSET are not permitted; if they occur, the SIR function is aborted.

The following applies to the allocation of the pubset with respect to other systems or tasks:

- The disks remain exclusively allocated until processing with CREATE-VOLUME has been completed.
- During execution of the subsequent statements, the disks are always allocated as shareable. During these phases, it is basically possible for other systems or tasks to share the pubset. However, allocation by other systems can lead to inconsistencies, particularly during creation or extension of a DRV (Dual Recording by Volume) pubset. These allocations by other systems cannot be prevented by SIR. No suitable functions which work across several systems are available for this purpose. System administration should implement the necessary organizational measures to ensure that no other users access the pubset during the SIR run.

> The subsequent setting up of paging files is supported outside SIR by commands.

“Large” disks and files

Each pubset has two attributes which signal whether or not disks and files with a capacity in excess of 32 Gbytes (referred to simply as large disks/files) are permitted for this pubset.

Large files (without a PAM key) can be created on NK and K disks (large files generally have no PAM key).

The attributes are set by SIR when creating a pubset using the `LARGE-DISKS-ALLOWED` and `LARGE-FILES-ALLOWED` operands of the `//DECLARE-PUBSET` statement. Alternatively, large disks/files can be permitted subsequently using the BS2000 command `/SET-PUBSET-PARAMETER`.

If a large disk is specified while creating or extending a pubset despite the fact that this is not permitted by the relevant attribute, the disk will be rejected with a corresponding message. If you wish to add this disk to the pubset nonetheless, the operand `LARGE-DISKS-ALLOWED=*YES` must be specified with `ACTION=*INSTALL`, and the attribute must be changed beforehand using the BS2000 command `/SET-PUBSET-ATTRIBUTES` with `ACTION=*EXTEND`.

> A home pubset may contain large disks and large files.
> Once a pubset is set to permit large disks/files, this setting cannot be reversed.
>
> A disk/file with a capacity of less than 32 Gbytes (small disk) can be included in any pubset, irrespective of the attribute settings.

13.3 Overview of SIR functions and statements

SF pubset

Function / Statements	Set up pubset	Extend pubset	Set up IPL disk	Modify IPL disk
DECLARE-PUBSET TYPE=*S-F	X	X	X [2)]	X [2)]
CREATE-VOLUME	X	X		
CREATE-CATALOG	(X)			
CREATE-PAGING-FILE	(X)	(X)	(X) [2)]	(X) [2)]
CREATE-SNAP-FILE	(X)	(X)		
COPY	(X)	(X)	(X)	(X)
CREATE-IPL-VOLUME	(X)	(X)	X [1)]	
MODIFY-IPL-VOLUME	(X)	(X)	(X)	X
DELETE-IPL-FACILITY	(X)	(X)	(X)	(X)
INITIALIZE-PRIVATE-VOLUME	(X)	(X)	(X)	(X)
INITIALIZE-PUBLIC-VOLUME	(X)	(X)	(X)	(X)

Key:

X The statement must be specified.

(X) The statement may be specified.

1) The statement can be replaced by a MODIFY-IPL-VOLUME statement.

2) The statement must be omitted if only private disks are to be converted to IPL disks.

SM pubset

Statements outside the statement block BEGIN-VOLUME-SET-DECLARATION and END-VOLUME-SET-DECLARATION:

Function / Statements	Set up pubset	Extend pubset
DECLARE-PUBSET TYPE=*S-M	X	X
BEGIN-VOLUME-SET-DECLARATION ACTION=*INSTALL	X	(X)
BEGIN-VOLUME-SET-DECLARATION ACTION=*ADD	(X)	(X)
BEGIN-VOLUME-SET-DECLARATION ACTION=*EXTEND		(X)
END-VOLUME-SET-DECLARATION	X	X
DELETE-IPL-FACILITY	(X)	(X)
INITIALIZE-PRIVATE-VOLUME	(X)	(X)
INITIALIZE-PUBLIC-VOLUME	(X)	(X)
COPY	(X)	(X)

Statements within the statement block BEGIN-VOLUME-SET-DECLARATION and END-VOLUME-SET-DECLARATION:

Function / Statements	Set up volume set	Extend volume set	Add volume set
CREATE-VOLUME	X	X	
CREATE-CATALOG	(X)		
CREATE-PAGING-FILE	(X)	(X)	
DELETE-IPL-FACILITY	(X)	(X)	(X)
INITIALIZE-PRIVATE-VOLUME	(X)	(X)	(X)
INITIALIZE-PUBLIC-VOLUME	(X)	(X)	(X)

Key:

- X The statement must be specified.
- (X) The statement may be specified.

A detailed description of the statements can be found on "Statements".

Editing the specifications for pubsets/volume sets

The START-EXECUTION editor statement terminates the input sequence of the statements for setting up or extending pubsets/volume sets and initiates processing. After a logical check (see "Statement editor") the specifications are output to the pubsets in edited format:

Example of an SF pubset overview

```
*********************************************************************
* PUBSET-TYPE: SINGLE-FEATURE                                       *
* LARGE-DISKS-ALLOWED: NO                                           *
* LARGE-FILES-ALLOWED: NO                                           *
*                                                                   *
*     P U B S E T - D I S K  T A B L E        (ACTION:  *INSTALL)   *
*                                                                   *
*   PVS-ID  SYS-ID FORMAT  PHYSICAL-   ALLOCATION-  AVAILABILITY   *
*                          BLOCK-SIZE  UNIT                       *
*                                                                   *
*     DEMO     65      K        2K           6         NORMAL      *
*                                                                   *
*   VSN  DEVICE-  VOLIN- MAIN- SUB-  CHECK- CAT-  PAG-  SNAP- IPL-*
*        TYPE     ACTION UNIT  UNIT  VSN    SIZE  SIZE  SIZE  VOL *
* DEMO.0 D3435 I                     OLD.00 1000         48    Y  *
* DEMO.1 D3435 I                     OLD.01 1000   150         Y  *
* DEMO.2 D3435 I                     OLD.02 1000   150            *
*                                                                   *
*********************************************************************
*********************************************************************
*           C R E A T E - I P L - V O L U M E  T A B L E            *
*                                                                   *
* PUBLIC DISKS                                                      *
*                                                                   *
* VSN:DEMO.0    DEVICE:D3435                                        *
*                                  HSI:390                          *
*                                                                   *
* VSN:DEMO.1    DEVICE:D3435                                        *
*                                  HSI:X86                          *
*                                                                   *
*********************************************************************
```

Example of an overview when editing an SM pubset

```
*********************************************************************
* PUBSET-TYPE:        SYSTEM-MANAGED                                *
* PUBSET:             DEMO                                          *
* CONTROL-VOLUME-SET: CTL                                           *
* LARGE-DISKS-ALLOWED: NO                                           *
* LARGE-FILES-ALLOWED: NO                                           *
*                                                                   *
* VOLUME- ACTION  FORMAT  PHYSICAL-  ALLOCATION- AVAILABILITY       *
* SET                     BLOCK-SIZE UNIT                           *
*                                                                   *
*  CTL    INSTALL   NK        2K          8          DRV           *
*                                                                   *
* LOGICAL VOLUME-SET ATTRIBUTES:                                    *
*                                                                   *
* PERFORMANCE WRITE-            USAGE       AVAILABILITY          *
*             CONSISTENCY                                           *
*                                                                   *
*    STD       BY-CLOSE          STD           STD                 *
*                                                                   *
*   VSN  DEVICE-  VOLIN- MAIN- SUB-  CHECK- CAT-  PAG-  SNAP- IPL-*
*        TYPE     ACTION UNIT  UNIT  VSN    SIZE  SIZE  SIZE  VOL *
* CTL.00 D3435 I                     OLD.00  2500                   *
* CTL.01 D3435 I                     OLD.01  2500                   *
* CTL.01 D3435 I                     OLD.01  2500                   *
*                                                                   *
* VOLUME- ACTION  FORMAT  PHYSICAL-  ALLOCATION- AVAILABILITY       *
* SET                     BLOCK-SIZE UNIT                           *
*                                                                   *
*  VOL1   INSTALL   K         2K          6         NORMAL         *
*                                                                   *
* LOGICAL VOLUME-SET ATTRIBUTES:                                    *
*                                                                   *
* PERFORMANCE WRITE-            USAGE       AVAILABILITY          *
*             CONSISTENCY                                           *
*                                                                   *
*    STD       BY-CLOSE          STD           STD                 *
*                                                                   *
*   VSN  DEVICE-  VOLIN- MAIN- SUB-  CHECK- CAT-  PAG-  SNAP- IPL-*
*        TYPE     ACTION UNIT  UNIT  VSN    SIZE  SIZE  SIZE  VOL *
*                                                                   *
* VOL1.0 D3435 I                     OLD.02  2500                   *
* VOL1.1 D3435 I                     OLD1.1                         *
*                                                                   *
*********************************************************************
```

Example of an overview when editing free volume sets

```
*********************************************************************
*                                                                   *
* VOLUME- ACTION  FORMAT  PHYSICAL-  ALLOCATION- AVAILABILITY       *
* SET                     BLOCK-SIZE UNIT                           *
*                                                                   *
*  VOL1   INSTALL   NK        2K         64         NORMAL         *
*                                                                   *
*   VSN  DEVICE-  VOLIN- MAIN- SUB-  CHECK- CAT-  PAG-  SNAP- IPL-*
*        TYPE     ACTION UNIT  UNIT  VSN    SIZE  SIZE  SIZE  VOL *
*                                                                   *
* VOL1.0 D3435 I                    OLD.00  2500                    *
* VOL1.1 D3435 I                    OLD.01                          *
*                                                                   *
*********************************************************************
```

In both overviews, the section for describing a volume set is repeated for each volume set to be processed.

13.4 Installing and operating SIR

Version dependencies and system integration

When creating pubsets, there are two types of version dependency: target version dependency and execution version dependency.

SIR V21.0 can be used to create a home pubset for BS2000 from the **execution versions BS2000 OSD/BC V10.0, V11.0 and BS2000 V21.0** and make it bootstrappable.

You should bear in mind that only device types belonging to the device spectrum of BS2000 (both in the execution version and in BS2000 V21.0) are supported.

A SIR version is generally coupled with a particular version of BS2000. Therefore, the following must always apply: Execution version <= target version.

This means that downgrading is not possible (e.g. setting up a pubset with version-specific features of BS2000 OSD /BC V11.0, which is then to be executed on BS2000 V21.0 with SIR V21.0).
It is recommended to keep a suitable home pubset or IPL disk, in case you might need to return to an earlier version in the future.

Upgrading from BS2000 OSD/BC V10.0 and V11.0

In the legacy versions BS2000 OSD/BC V10.0 and V11.0, SIR V21.0 can only be installed and temporarily activated using BS2GA.MIGRATE V21.0. A comprehensive description of the upgrade can be found in the “System installation” manual [7].

Privileges in SIR

Using the SIR functions leads to structural changes in disk contents and is therefore reserved for users with the appropriate authorization (TSOS privilege and STD-PROCESSING privilege). In addition, the operations to be performed on the disks are only possible in conjunction with privileged system functions. For this reason, on completion of parameter input, the SIR function executes as a DSSM subsystem after an SVC. Attempts by parallel tasks to use the same pubset ID at the same time will be rejected.

Calling SIR

The SIR utility routine can be called in either interactive or batch mode using the command `/START-SIR` or `/SIR`.

The statements are read from the system file SYSDTA. Input errors can be corrected at once in interactive mode as long as SYSDTA has not been reassigned. Otherwise input is aborted as soon as an invalid statement is encountered.
Messages are routed to SYSOUT.

Example of a SIR run

A pubset is extended by a disk which is made bootstrappable.

```
/start-sir
%  SIRLOAD PROGRAM SIR, VERSION <version> OF <date> LOADED FROM FILE
         :CAM1:$TSOS.SYSLNK.SIR.<version>
%  SIR0000 PROGRAM SIR (VERSION <version>) READY FOR INPUT
declare-pubset pubset-type=*single-feature(pubset=9701,action=*extend)
create-volume disk-number=2(overwrite-disk=c0002r,unit=A901)
create-ipl-volume disk-number=2
list-statements
*********************************************************************
*                                                                   *
* OPTION AND TASK ORDERS GENERATED :                                *
*                                                                   *
*    STATEMENTS GENERATED :                                         *
*      NUMBER  STATEMENT                                            *
*          1   //DECLARE-PUBSET PUBSET-TYPE=*SINGLE-FEATURE(PUBS-  *
*              ET=9701,ACTION=*EXTEND,SYS-ID=*CAT-ID)               *
*          2   //CREATE-VOLUME DISK-NUMBER=2(OVERWRITE-DISK=C000-  *
*              2R,UNIT=A901(SUBUNIT=*NO)),DEVICE-TYPE=*BY-UNIT,F-  *
*              ORMAT=*NO                                            *
*          3   //CREATE-IPL-VOLUME DISK-NUMBER=2,HSI-TYPE=*BY-OWN  *
*              -HSI                                                 *
*                                                                   *
*********************************************************************
%  SIR7054 ENTER LIST SUBFUNCTION OR 'END'. REPLY ('+'=SCROLL FORWARD;
         '-'=SCROLL BACK; END=TERMINATE LST FUNCTION)
end
create-paging-file disk-number=2,file-size=150
start-execution
*********************************************************************
* PUBSET-TYPE: SINGLE-FEATURE                                       *
* LARGE-DISKS-ALLOWED: NO                                           *
* LARGE-FILES-ALLOWED: NO                                           *
*                                                                   *
*     P U B S E T - D I S K  T A B L E        (ACTION:  *EXTEND)     *
*                                                                   *
*   PVS-ID  SYS-ID FORMAT  PHYSICAL-   ALLOCATION- AVAILABILITY     *
*                          BLOCK-SIZE  UNIT                         *
*                                                                   *
*     9701     65     K         2K           6        NORMAL       *
*                                                                   *
*   VSN  DEVICE-  VOLIN- MAIN- SUB-  CHECK- CAT-  PAG-  SNAP- IPL-*
*        TYPE     ACTION UNIT  UNIT  VSN    SIZE  SIZE  SIZE  VOL *
* 9701.2 D3435      I    A901        C0002R        150         Y   *
*                                                                   *
*********************************************************************
*********************************************************************
*           C R E A T E - I P L - V O L U M E  T A B L E            *
*                                                                   *
* PUBLIC DISKS                                                      *
*                                                                   *
* VSN:9701.2    DEVICE:D3435                                        *
*                                HSI:390                            *
```

```
*                                                                    *
**********************************************************************
%  SIR0040 DO YOU WANT TO CORRECT INPUT? REPLY (Y=YES; N=NO)?
n
%  SIR0300 'CREATE-VOLUME' FUNCTION STARTED
%  NVL0000 VOLIN VERSION <version> READY
%  NVL0031 INITIALIZATION STARTED FOR VOLUME '9701.2' ON UNIT 'A901' IN
           FORMAT 'K(A-U=6)'
%  NVL0017 INITIALIZATION OF VOLUME '9701.2' ON UNIT 'A901' COMPLETED.
           VOLUME FORMAT: 'K(A-U=6)'
%  SIR0301 'CREATE-VOLUME' FUNCTION TERMINATED NORMALLY
%  SIR0169 IMCAT TASK STARTED
%  SIR0400 CREATION OF PAGING FILES STARTED
%  SIR0401 CREATION OF PAGING FILES TERMINATED NORMALLY
%  SIR0523 //CREATE-IPL-VOLUME: FILE CHECK STARTED
%  SIR0524 //CREATE-IPL-VOLUME: CHECK OF FILES TERMINATED
%  SIR0500 //CREATE-IPL-VOLUME: PROCESSING STARTED
%  SIR0501 //CREATE-IPL-VOLUME: PROCESSING TERMINATED NORMALLY
**********************************************************************
* PUBSET-TYPE: SINGLE-FEATURE                                        *
* LARGE-DISKS-ALLOWED: NO                                            *
* LARGE-FILES-ALLOWED: NO                                            *
*                                                                    *
*     P U B S E T - D I S K  T A B L E        (ACTION:  *EXTEND)     *
*                                                                    *
*   PVS-ID  SYS-ID FORMAT  PHYSICAL-   ALLOCATION- AVAILABILITY      *
*                          BLOCK-SIZE  UNIT                          *
*                                                                    *
*     9701     65     K         2K           6        NORMAL        *
*                                                                    *
*   VSN  DEVICE-  VOLIN- MAIN- SUB-  CHECK- CAT-  PAG-  SNAP- IPL-* *
*        TYPE     ACTION UNIT  UNIT  VSN    SIZE  SIZE  SIZE  VOL  *
* 9701.2 D3435      I    A901        C0002R        150         Y   *
*                                                                    *
**********************************************************************
**********************************************************************
*           C R E A T E - I P L - V O L U M E  T A B L E             *
*                                                                    *
* PUBLIC DISKS                                                       *
*                                                                    *
* VSN:9701.2    DEVICE:D3435                                         *
*                                 HSI:390                            *
*                                                                    *
**********************************************************************
end
%  SIR1010 'SIR' TERMINATED
```

13.5 Copying files with SIR

The utility routine can be used under any user ID with the privilege STD-PROCESSING. It is called in either interactive or batch mode using the `/START-SIR` or `/SIR` command.

Messages are routed to SYSOUT. Statements are read from SYSDTA. Errors can be corrected in interactive mode (and immediately if SYSDTA=(PRIMARY) is set). In batch mode, input is aborted as soon as an invalid statement is encountered.

Statements to “nonprivileged SIR” - an overview

Statement	Brief description
COPY	Copies files
END	Terminates an input sequence

13.6 Statements

- Overview of SIR statements
- Input rules
- Statement editor
- Description of the statements
  - BEGIN-VOLUME-SET-DECLARATION - Set up a volume set
  - COPY - Copy files
  - CREATE-CATALOG - Define file catalog
  - CREATE-IPL-VOLUME - Convert disk to IPL disk
  - CREATE-PAGING-FILE - Create paging file
  - CREATE-SNAP-FILE - Create snapshot file
  - CREATE-VOLUME - Initialize disks
  - DECLARE-PUBSET - Define pubset and type of processing
  - DELETE-IPL-FACILITY - Delete IPL capability and IPL files of disk
  - END - Terminate SIR
  - END-VOLUME-SET-DECLARATION - Terminate statement sequence for volume set
  - INITIALIZE-PRIVATE-VOLUME - Initialize private disks
  - INITIALIZE-PUBLIC-VOLUME - Initialize public disks
  - MODIFY-IPL-VOLUME - Modifying an IPL disk

13.6.1 Overview of SIR statements

Overview of statements to the “privileged SIR”

Statement	Brief description
DECLARE-PUBSET	Defines general parameters for the pubset that is to be processed
BEGIN-VOLUME-SET-DECLARATION	Defines global parameters for the volume set that is to be processed (initializes the statement block)
END-VOLUME-SET-DECLARATION	Terminates the volume set statement block
CREATE-VOLUME	Initializes disks
CREATE-CATALOG	Creates a file catalog
CREATE-PAGING-FILE	Creates paging files
CREATE-SNAP-FILE	Creates the snapshot file for recording snapshot dumps
COPY	Copies files
CREATE-IPL-VOLUME	Creates an IPL disk (SF pubset only)
MODIFY-IPL-VOLUME	Modifies an IPL disk (SF pubset only)
DELETE-IPL-FACILITY	Removes the IPL facility and boot files of a disk (SF pubset only)
INITIALIZE-PRIVATE-VOLUME	Initializes a private disk
INITIALIZE-PUBLIC-VOLUME	Initializes a disk without adding it to a pubset
START-EXECUTION	Executes the statement sequence (see "Statement editor")
END	Terminates the SIR

13.6.2 Input rules

Because more than one volume set can be generated in a SIR run, the statements must be unambiguously assignable to a volume set. The statement block is designed for this purpose to create a volume set which starts with the statement BEGIN-VOLUME-SET-DECLARATION and ends with END-VOLUME-SET-DECLARATION. The statements can be in any sequence within a given scope. If several statements of the same type are input which contradict one another, the last statement is the valid one.

The following statements are processed immediately during input of a sequence of statements:

DELETE-IPL-FACILITY
INITIALIZE-PRIVATE-VOLUME
INITIALIZE-PUBLIC-VOLUME
START-EXECUTION
END

If one of these statements is entered as part of a sequence of statements, the sequence remains unaffected by the immediate execution of the function. Input of the sequence of statements can then be continued.
The last two statements shown are exceptions to this. START-EXECUTION starts processing of the entered sequence. The current sequence of statements is deleted once the statement is executed. SIR terminates if the statement END is read in.

Correcting statements

No correction dialog is possible for statements entered incorrectly. The statement in question can, however, be deleted or replaced by a statement of the same type, provided that you are still in the same scope.

Example

```
X  DECLARE-PUBSET  PUBSET-TYPE=*S-F(PUBSET=A)
X  CREATE-VOLUME   02,D3435,FORMAT=YES
   CREATE-VOLUME   03,D3435,FORMAT=YES
X  CREATE-VOLUME   03,D3435,FORMAT=NO
```

or

```
o  BEGIN-VOLUME-SET-DECLARATION VOLUME-SET=A
o  CREATE-VOLUME 02,D3435
o  END-VOLUME-SET-DECLARATION
X  BEGIN-VOLUME-SET-DECLARATION VOLUME-SET=A
X  CREATE-VOLUME 00,D3435
X  END-VOLUME-SET-DECLARATION
X  START-EXECUTION
```

The statements marked with an X are valid. Those marked with an o must be explicitly deleted.

Incorrect statements can be explicitly deleted from the sequence of statements using the editor command DELETE-STATEMENTS. The LIST-STATEMENTS command returns the number of statements in the sequence. The INSERT-STATEMENTS command can be used to insert a new statement in place of the incorrect one.

An input sequence is not terminated until the editor statement START-EXECUTION is issued. After that, the logical check of all the statements is started.

The DELETE-IPL-FACILITY and INITIALIZE-PRIVATE/PUBLIC-VOLUME statements can be input at any point in the input procedure before the END statement. They are then executed immediately.

Initializations cannot be processed in parallel. The complete processing of the pubset/volume set is carried out serially in one task. To avoid serial processing, the disks in question can be initialized with a specific bit pattern in parallel tasks using the INITIALIZE-PUBLIC-VOLUME statement before the pubset/volume set generation is performed.
The specification FORMAT=*NO (default) in the CREATE-VOLUME statement is then sufficient for these disks in the subsequent pubset/volume set generation.

If, during the SIR run, the processing of a pubset or volume set with a specific ID is active in a user task, that ID is locked for other tasks. SIR calls with user tasks of the same pubset ID running in parallel are aborted with an appropriate message.

i A maximum of 32,767 SIR statements can be specified.
Maximum number of disks per SF pubset or volume set:

- 255 for two-digit or three-digit catalog ID and an allocation unit size not equal 6 Kbytes
- 36 for four-digit catalog ID and an allocation unit > 6 KB
- 32 in all other pubsets

Maximum number of volume sets in an SM pubset: 255

The general SDF rules apply to the use of positional operands, abbreviations and continuation lines.

13.6.3 Statement editor

After SIR is started, the first thing that is called is the statement editor. It allows the user to enter new statements or modify existing ones interactively during a session.

The statement editor recognizes the following editor commands:

DELETE-STATEMENTS STATEMENT-NUMBER=*ALL / 1..32767

Deletes a statement or an entire sequence.

INSERT-STATEMENTS STATEMENT-NUMBER=*APPEND / 1..32767

Inserts the subsequent statements at the specified statement number.
*APPEND adds the subsequent statements at the end of the sequence.
The statements are also added at the end of the sequence if the statement number specified is greater than that of the last statement.

RENUMBER-STATEMENTS

Renumbers the statements in ascending order.

LIST-STATEMENTS

Lists the statements in the current sequence.

The statement editor remains active during the SIR run. The user initiates processing of the sequence of statements from the editor using START-EXECUTION.

The sequence undergoes a logical check following START-EXECUTION (i.e. the sequence is checked for completeness and consistency). The user's input is then output in edited format (see statement output example on "Overview of SIR functions and statements").

If the check detects any suspect entries, the user is given the opportunity to correct the sequence of statements by branching back to the statement editor. Otherwise, the functions are started.

SIR returns to the statement editor once the sequence of statements has been processed. The user can then terminate by specifying a further sequence or entering the END statement.

13.6.4 Description of the statements

- BEGIN-VOLUME-SET-DECLARATION - Set up a volume set
- COPY - Copy files
- CREATE-CATALOG - Define file catalog
- CREATE-IPL-VOLUME - Convert disk to IPL disk
- CREATE-PAGING-FILE - Create paging file
- CREATE-SNAP-FILE - Create snapshot file
- CREATE-VOLUME - Initialize disks
- DECLARE-PUBSET - Define pubset and type of processing
- DELETE-IPL-FACILITY - Delete IPL capability and IPL files of disk
- END - Terminate SIR
- END-VOLUME-SET-DECLARATION - Terminate statement sequence for volume set
- INITIALIZE-PRIVATE-VOLUME - Initialize private disks
- INITIALIZE-PUBLIC-VOLUME - Initialize public disks
- MODIFY-IPL-VOLUME - Modifying an IPL disk

13.6.4.1 BEGIN-VOLUME-SET-DECLARATION - Set up a volume set

This statement initiates the declaration of the volume set to be set up. Together with the END-VOLUME-SET-DECLARATION statement which terminates the declaration it embraces a statement block which must be repeated for each volume set to be set up (see also the “Setting up a volume set” function on "Volume sets").

Notes

- When defining the structure of the volume set, bear in mind that these settings cannot be modified later without the volume set being set up once more. This includes all settings specified by means of the FORMAT operand.
- All logical attributes of the volume set which can be specified by means of the VOLUME-SET-ATTR operand can be modified during operation using the BS2000 command /MODIFY-PUBSET-DEFINITION-FILE.

BEGIN-**VOL**UME-**SET-DECL**ARATION

```
VOLUME-SET=volume-set-id
,ACTION= *INST ALL(...) / *ADD(...) / *EXTEND
    *INST ALL(...)
        | FORMAT= *K / *NK(...)
        |     *NK(...)
        |         | PHYSICAL-BLOCK-SIZE= *2K (...) / *4K(...)
        |         |     *2K (...)
        |         |         | ALLOCATION-UNIT= 6 / 8 / 64
        |         |     *4K(...)
        |         |         | ALLOCATION-UNIT= 8 / 64
        | ,AVAILABILITY= *NORM AL / *DRV
        | ,VOLUME-SET-ATTR=*PARAMETERS(...)
        |     *PARAMETERS(...)
        |         | USAGE= *STD / *WORK / *HSMS-CONTROLLED
        |         | ,AVAILABILITY= *STD / *HIGH
        |         | ,PERFORMANCE-ATTR= *STD / *PARAMETERS(...)
        |         |     *PARAMETERS(...)
        |         |         | PERFORMANCE=list-poss(3): *STD / *HIGH / *VERY-HIGH
        |         |         | ,WRITE-CONSISTENCY= *BY-CLOSE / *IMMEDIATE
    *ADD(...)
        | VOLUME-SET-ATTR=*PARAMETERS(...)
        |     *PARAMETERS(...)
        |         | USAGE= *STD / *WORK / *HSMS-CONTROLLED
        |         | ,AVAILABILITY= *STD / *HIGH
        |         | ,PERFORMANCE-ATTR= *STD / *PARAMETERS(...)
        |         |     *PARAMETERS(...)
        |         |         | PERFORMANCE=list-poss(3): *STD / *HIGH / *VERY-HIGH
        |         |         | ,WRITE-CONSISTENCY= *BY-CLOSE / *IMMEDIATE
```

VOLUME-SET=volume-set-id
Identifier for the volume set. This is a syntactic component of the VSN of all disks that belong to the volume set. The volume set identifier has two formats:

volume-set-id1:

The identifier is 1 character long (alphanumeric) and is the fourth character in the VSN of each disk that belongs to the volume set.
Format of the VSN:
PUB**p**xx where:

p = volume-set-id1

xx = number of the disk in the volume set
Value range: 00 - 31

volume-set-id2:

The identifier is 2 to 4 characters long (alphanumeric) and is located at the beginning of the VSN. Format of the VSN:
pp.xyz or **ppp**.yz or **pppp**.z where:

p = character from volume-set-id2

x,y,z = number of the disk in the volume set

Value range for allocation unit > 6
x,y,z: 0 - 9, A - Z.

Value range for allocation unit = 6

x,y: The positions for x and y contain 0;

z: 0 - 9, A - V.

Because the volume set can contain up to 32 disks, the values W to Z are not permissible.

volume-set-id2 must not begin with the string PUB.

ACTION
Specifies the processing mode for the volume set.

ACTION=*INSTALL(...)
The volume set specified with the volume set ID is to be created.

FORMAT
Defines the disk format and thus also the volume set mode with respect to PAM key use.

FORMAT=*K
The PAM key is made available. The volume set consists of K disks.
K files with BLKCTRL=PAMKEY are permitted.

FORMAT=*NK(...)
The disks of the volume set are formatted without a PAM key (NK disks). Only NK files with BLKCTRL=DATA / NO are possible.

PHYSICAL-BLOCK-SIZE
Defines the physical block size on the NK disks of the volume set.

PHYSICAL-BLOCK-SIZE=*2K(...)
NK2 disk (default value)
The minimum transfer unit between disk and memory is 2 Kb. This means a write or read job always moves 2 Kb (or a multiple thereof).

ALLOCATION-UNIT=6 / 8 / 64
Defines the unit size of the space allocated for files by DMS on the volume set disks. 6 Kb is the default for NK2 disks.

PHYSICAL-BLOCK-SIZE=*4K(...)
NK4 disk The minimum transfer unit is 4 Kb.

ALLOCATION-UNIT=8 / 64
Defines the unit size of the space allocated for files by DMS on the volume set disks. 8 Kb is the default value for NK4 disks.

AVAILABILITY
Defines the availability of the volume set.

AVAILABILITY=*NORMAL
No DRV disks are created (usual availability).

AVAILABILITY=*DRV
A volume set is set up with DUAL-RECORDING.

VOLUME-SET-ATTR=*PARAMETERS(...)
The logical attributes of the volume set are defined via parameters.

USAGE
Determines how the volume set is to be used as a file storage location.

USAGE=*STD
The volume set is to be used as the storage location for default files.

USAGE=*WORK
The volume set is to be used as the storage location for work files.

USAGE=*HSMS-CONTROLLED
The volume set is used by the HSMS subsystem to implement the S1 storage level and the backup volume set of an SM pubset.

AVAILABILITY
Determines the degree of file availability.

AVAILABILITY=*STD
The volume set offers no increased reliability.

AVAILABILITY=*HIGH
The volume set offers higher reliability.

PERFORMANCE-ATTR
Defines the performance attributes for the volume set.

PERFORMANCE-ATTR=*STD
The volume set has the standard performance attributes.

PERFORMANCE-ATTR=*PARAMETERS(...)
The performance attributes of the volume set are defined via parameters.

PERFORMANCE=list-poss(3):
More than one performance attribute (maximum: 3) can be specified.

PERFORMANCE=*STD
The volume set does not offer higher I/O performance.

PERFORMANCE=*HIGH
The volume set offers higher I/O performance.

PERFORMANCE=*VERY-HIGH
The volume set offers very high I/O performance.

WRITE-CONSISTENCY
Defines the consistency time for write accesses to a disk of the volume set.

WRITE-CONSISTENCY=*BY-CLOSE
The consistency of write accesses is not set up until the file is closed.

WRITE-CONSISTENCY=*IMMEDIATE
The consistency of write accesses is set up immediately.

ACTION=*ADD(...)
A “free” volume set, i.e. one that is empty and does not yet belong to a pubset, is to be included as it is in an SM pubset.

The operands of the substructures for ACTION=*ADD are described in the operand ACTION= *INSTALL.

ACTION=*EXTEND
The volume set specified by the volume set ID is to be extended.

13.6.4.2 COPY - Copy files

This statement is used to copy files between privated disks and pubsets. The description of the individual operands indicates whether the operand requires the TSOS privilege.

```
COPY

FILE=name(...)

    name(...)

        | NEW= *SAME / name / *PREFIX(...)

        |      *PREFIX(...)

        |          |  ADD= *NO / pfx

        |          |  ,REMOVE= *NO / pfx

,FROM= *PUBSET / *DEVICE(...)

    *DEVICE(...)

        | VOLUME=vsn

        | ,DEVICE-TYPE=device-type / STDDISK

,TO= *PUBSET(...) / vsn(...)

    *PUBSET(...)

        | DISK-NUMBER= *ANY / <integer x..y>

    vsn(...)

        | DEVICE-TYPE=device-type / STDDISK

,REPLACE-OLD-FILES= *DIALOGUE / *YES /
*NO
```

FILE
Specifies the files to be copied.

FILE=name(...)
Specifies the files that are to be transcribed by means of a partially or fully qualified name. If no pubset ID is specified, the following applies:
The pubset ID is taken from the DECLARE-PUBSET statement if issued, or from the home pubset.

NEW
Specifies the new names to be given to the transferred files.

NEW=*SAME
The files are cataloged under the same name as specified in the FILE operand. *SAME must not be specified together with FROM=*PUBSET.

Example

If the file sequence A.B. is to be copied from the current BS2000 pubset to the target pubset (pubset ID from the DECLARE-PUBSET statement), the following statement must be used: `COPY FILE=:<source-pubset>:A.B.(NEW=A.B.)`

NEW=name
Fully/partially qualified name under which the file specified by FILE=name is to be cataloged. The name specified in FILE must also be fully/partially qualified.

NEW=*PREFIX(...)
Controls the addition or removal of prefixes when forming target file names.

ADD=*NO
No prefix is added to the target file name.

ADD=pfx
The prefix pfx is added to the name of the target file:
Source file: ABC, prefix: V21.0 becomes
Target file: V21.0.ABC

REMOVE=*NO
An existing prefix is not removed when forming the name of a target file.

REMOVE=pfx
An existing prefix is removed when forming the name of a target file.

Source file: ZN.ABC, prefix: LG becomes

Target file: ZN.ABC

Source file: ZN.ABC, prefix: ZN becomes

Target file: ABC

pfx can be up to 10 characters long. Neither the catalog ID nor the user ID may be used for pfx.

If both ADD and REMOVE are specified in the same statement, REMOVE is processed before ADD:

```
PREFIX ADD=V21.0,REMOVE=ZN
```

Source file: ZN.ABC becomes

Target file: V21.0.ABC

FROM
Specifies from where the files are to be transcribed.

FROM=*PUBSET
The files are to be transcribed from the pubsets that have just been imported.

FROM=*DEVICE(...)

VOLUME
The files are to be transcribed from private volumes.

VOLUME=vsn
Volume serial number of the private volume.

DEVICE-TYPE=device-type / STDDISK
When VOLUME is specified, the device type or the standard device type (STDDISK) must also be defined.

TO
Specifies to where the files are to be transcribed.

TO=*PUBSET(...)
If DECLARE-PUBSET has been specified, the files are transcribed to this pubset. Otherwise they are transcribed to the default pubset of the user.

DISK-NUMBER
This operand may only be specified by users with the TSOS privilege.

Specifies the disk on the SF pubset to which the files are to be transcribed. The operand is ignored for disks in an SM pubset.

DISK-NUMBER=*ANY
The files are to be transcribed to any disks within the SF pubset.

DISK-NUMBER=n
The files are to be transcribed to the disk denoted by n. (for “number of the disk in the pubset”, see the CREATE-VOLUME statement).

TO=vsn(...)
Volume serial number of the private disk to which the files are to be transcribed.

DEVICE-TYPE=device-type
Device type of the private disk.

DEVICE-TYPE=STDDISK
Default device type of the private disk.

REPLACE-OLD-FILES
Specifies the procedure to be followed if a file has already been cataloged under the same name.

REPLACE-OLD-FILES=*DIALOGUE
In interactive mode, the user receives a message and can decide whether or not the file is to be overwritten. In execution modes where there is no dialog with the user (e.g. batch processing), specifying *DIALOGUE has the same effect as *NO.

REPLACE-OLD-FILES=*YES
An already existing file is to be overwritten (at the same location if the FCB type is the same).

REPLACE-OLD-FILES=*NO
An already existing file is not to be overwritten.

Notes

1. All the files required for the installation must be transcribed by means of the COPY statement.
2. Job variables and file generation groups are skipped.
3. The file attributes are taken from the source file.
4. In the event of disk errors, processing of the current file for transcription is aborted; where possible, the system will then go on to the next file.
5. A conversion of the file format is not possible.
6. If, while copying, a file is to be read into a different ID (only possible under TSOS), this must be specified via the FILE operand NEW=name.

13.6.4.3 CREATE-CATALOG - Define file catalog

This statement is used to define the proportion of the file catalog of a pubset or volume set that is to be created on the specified disk or pubset area.
The size of the TSOSCAT of the pubset or volume set is the sum of the areas defined for the individual disks.

```
CREATE-CATALOG

  DISK-NUMBER=list-poss(32): n / *RANGE(...)

        *RANGE(...)

              |       FROM=n

              |       ,TO=m

  ,FILE-SIZE=2000 / <integer 60..64016>
```

DISK-NUMBER
Specifies the disks on which a catalog section is to be created. Depending on the format of the pubset or volume set identifier, the number of the disk in the pubset or volume set must be specified.

DISK-NUMBER=n
One-, two- or three-digit number of the disk in the pubset or volume set (see "CREATE-VOLUME - Initialize disks"). Specification of a list with up to 32 elements is possible.

DISK-NUMBER=*RANGE
A number of disks within a range are to be processed. The range is delimited by means of the FROM and TO operands.

FROM=n
Start of the range.

TO=m
End of the range.

A catalog section is created on each disk in the area; the size of the section is determined by the FILE-SIZE operand.
n must be smaller than m.

FILE-SIZE=<integer 60..64016>
Specifies the size (in PAM pages) of the catalog for each disk.
The size is rounded up internally to a full even UNIT number.

FILE-SIZE=2000
The default value for the catalog size is 2000 PAM pages.

Notes

1. If system administration does not specify a CREATE-CATALOG statement for the system disk of the pubset /volume set (Pubres/Volres), a catalog section of 2000 PAM pages is created there by default.
2. If, for DECLARE-PUBSET ACTION=*INSTALL, the sum of the values for the FILE-SIZE operand on each disk is greater than the maximum permissible size for the catalog, a number of catalog extents are set up successively until the maximum size is reached. If required, the size of the last catalog extent is reduced.
3. Only the EXTRA-LARGE TSOSCAT type is supported.
4. SIR aborts with an error message if the specified catalog size is larger than the disk capacity.

13.6.4.4 CREATE-IPL-VOLUME - Convert disk to IPL disk

This statement makes a disk bootstrappable, i.e. converts it into an IPL disk. The IPL capability of disks is described in detail on "SF pubsets".

CREATE-**IPL-VOL**UME

DISK-NUMBER=n / ***PRIV**ATE(...)

***PRIV**ATE(...)

| **DISK**=vsn

| ,**DEV**ICE-**TYPE**= **STDDISK** / device-type

,**HSI-TYPE**= ***BY-OWN-HSI** / ***390** / ***X86** [1)]

[1)] The *7500 operand value can still be specified for reasons of compatibility.

DISK-NUMBER
Specifies the disk that is to be made into an IPL disk. Depending on the format of the pubset ID (see the DECLARE-PUBSET statement), the number of the disk on the pubset must be specified.

DISK-NUMBER=n
One- to three-digit number of the IPL disk in the pubset (see "CREATE-VOLUME - Initialize disks").

DISK-NUMBER=*PRIVATE(...)
Specifies a private disk that is to be made into an IPL disk.

DISK=vsn
VSN of the private disk

DEVICE-TYPE=STDDISK
Default device type of the private disk.

DEVICE-TYPE=device-type
Device type of the private disk.

HSI-TYPE
Defines the HSI for which IPL capability is to be established.

HSI-TYPE=*BY-OWN-HSI
IPL capability is to be established for the HSI where SIR is running.

HSI-TYPE=*390
IPL capability is established for the /390 HSI (/390 servers).

HSI-TYPE=*X86
IPL capability is to be established for the X86 HSI (x86 servers).

Notes

1. The IPL files must be cataloged in the pubset if DISK-NUMBER=n is specified. IPL files include the IPL module (SYSPRG.IPL.<ver> with HSI=*390 and SKMPRG.IPL.<ver> with HSI=*X86), as well as the REP files for IPL and SLED (SYSREP.IPL.<ver> and SYSREP.SLED.<ver>), if these are available.
2. As explained above, for IPL conversion to be possible, the data must be present on the disk. CREATE-IPL-VOLUME fulfils this condition by copying the files to a file copy.
3. In the case of DISK-NUMBER=*PRIVATE, the IPL files must already be in the home pubset and in a format that is compatible with the private disk. A DECLARE-PUBSET statement must not be issued if during this SIR run one or more private disks are to be turned into IPL disks.
   If the REP files are not available for IPL and SLED, a message is output and processing continued.
4. A disk can only ever be bootstrappable for one HSI. This means that if IPL capability is established for HSI-TYPE=*X86 using CREATE-IPL-VOLUME, any IPL capability which previously existed for a /390 HSI is lost.
5. When establishing the IPL capability of a disk, the following backup files are created so as to record both memory data which is overwritten when BOOT and/or SLED is loaded and startup configuration data:
   - BOOT-file.<x>.SAVE
   - SLED-file.<x>.SAVE
   - SYSDAT.IPL-CONF.<x> (24 pages of 2 Kb each)

   where BOOT-file: file name of the BOOT load module

   SLED-file: file name of the SLED load module

   <x>: DSKxxx or vsn

   The files are created with BLOCKSIZE=(STD,2).

   The procedure is as follows if files with these names already exist:

   - The correct size file is on the correct disk.
     No further action is necessary.
   - The incorrect size file is on the correct disk.
     The file size is corrected.
   - The file is on another disk within the pubset. The file is created on the correct disk and deleted from the other disk.

Example

Standard IPL	SYSPRG.IPL.<ver>
Standard IPL Rep	SYSREP.IPL.<ver>
Standard SLED Rep	SYSREP.SLED.<ver>

`CREATE-IPL-VOLUME DISK-NUMBER=*PRIVATE(PRIV00),HSI-TYPE=*390` generates the following files on the IPL disk:

IPL	SYSPRG.IPL.PRIV00
BOOTSAVE	SYSPRG.BOOT.PRIV00.SAVE
SLEDSAVE	SYSPRG.SLED.PRIV00.SAVE
IPL-CONF	SYSDAT.IPL-CONF.PRIV00
IPL-REP	SYSREP.IPL.PRIV00
SLED-REP	SYSREP.SLED.PRIV00

6. If at least one disk of a pubset is converted to an IPL disk, SIR creates the file SYS.NSI.SAVEREP in order to save system corrections during system initialization.
7. All anchored files are given the file attributes BACKUP=E, DESTROY=*YES, ACCESS=*READ and MIGRATION=*FORBIDDEN.
8. SIR does not support file protection with BASIC-ACL for IPL files. If the user has protected one of these files with BASIC-ACL, the protection must be canceled to enable SIR to modify the IPL files.
9. The file names for storing the IPL files are formed implicitly. It is important to ensure that the IPL files are stored under the default name in the pubset. Otherwise a message is output and the statement ignored.
10. The HSI-TYPE operand should only be specified if IPL disk capability is not to beestablished for the HSI on which SIR is currently running.
11. The statement is rejected for disks of pubsets, which cannot be used as home pubsets:
    - pubsets with `SYS-ID=*STD`
    - pubsets with `PHYSICAL-BLOCK-SIZE=*4K`
    - SM pubsets

13.6.4.5 CREATE-PAGING-FILE - Create paging file

This statement is used to create the paging files.

```
CREATE-PAGING-
FILE

 DISK-NUMBER=list-poss(32): n / *RANGE(...)

        *RANGE(...)

                |       FROM=n

                |       ,TO=m

 ,FILE-SIZE=<integer 1..32767>
```

DISK-NUMBER
Specifies the disk of the pubset on which a paging file is to be created.
Depending on the format of the pubset ID (see the DECLARE-PUBSET statement), the number of the disk on the pubset must be specified.

DISK-NUMBER=n
One- to three-digit number of the IPL disk in the pubset (see "CREATE-VOLUME - Initialize disks"). A list containing up to 32 elements is permitted.

DISK-NUMBER=*RANGE(...)
A number of disks within a range are to be processed. The range is delimited by the FROM and TO operands.

FROM=n
Beginning of the area in the pubset.

TO=m
End of the area in the pubset.

A paging file is created on each disk. Its size is determined by the value of the FILE-SIZE operand.
n must be smaller than m.

FILE-SIZE=<integer 1..32767>
Defines the size of the paging file per disk in Mbytes.
If the disk capacity is less than that specified in the FILE-SIZE operand, a paging file is created that is the size of the maximum disk value.

Notes

1. The system is unable to operate without a paging file, but this need not necessarily be on the home pubset. It is therefore possible to set up pubsets without a paging file.
2. If no CREATE-PAGING-FILE statement was specified when setting up a pubset (ACTION=*INSTALL), a message is issued.
3. Paging files are given the same file attributes as are assigned with the /CREATE-PAGING-FILE command.
4. Further information on paging files can be found in the “Performance Handbook” [9 (Related publications)].

13.6.4.6 CREATE-SNAP-FILE - Create snapshot file

This statement serves to create a snapshot file. Only one snapshot file is permitted per pubset. It is cataloged under the file name $TSOS.SNAPFILE.

CREATE-**SNAP-** **FILE**
DISK-NUMBER=n ,**F**ILE-**SIZE**=<integer 16..1024>

DISK-NUMBER
Specifies the disk of the pubset on which the snapshot file is to be created.
Depending on the format of the pubset ID (see the DECLARE-PUBSET statement), the number of the disk on the pubset must be specified.

DISK-NUMBER=n
One- to three-digit number of the IPL disk in the pubset (see "CREATE-VOLUME - Initialize disks").

FILE-SIZE=<integer 16..1024>
Specifies the minimum size of the snapshot file in Mbytes.
The value is rounded up to the size of the allocation unit.
It must be possible to create the entire file as one extent.
If the disk capacity is less than the value of the FILE-SIZE operand, the snapshot file is created with the size of the maximum disk value.
Information on the size of the snapshot file is provided in the “Diagnostics Handbook” [2 (Related publications)].

Notes

1. If a paging file is created on the same disk, the snapshot file is created from the end of the disk and the paging file is centered. If the areas overlap on the disk, the snapshot file is not created.
2. Creation of a snapshot file is recommended when the SNAP dump is already to be available after system initialization. See the SNAP-ACTIVE-SWITCH parameter in the startup parameter service, “Introduction to System Administration” [5]. When setting up a pubset with `DECLARE-PUBSET PUBSET-TYPE=*S-F (ACTION=*INSTALL)`, a warning is output if no CREATE-SNAP-FILE statement is specified.
3. A snapshot file can also be created or modified retroactively using the ACTIVATE-SNAPSHOT command when the SNAP dump is activated.

13.6.4.7 CREATE-VOLUME - Initialize disks

This statement is used to initialize disks or ranges of a pubset or volume set. This corresponds to the VOLIN function.

CREATE-**VOL**UME

DISK-NUMBER=list-poss(32): n(...) / ***RANGE**(...)

n(...)

| **OVERWR**ITE-**DISK**=***ANY** / vsn

| ,**UNIT**=***NO** / mn(...)

| mn(...)

| | **SUBUNIT**=***NO** / mn

***RANGE**(...)

| **FROM**=n

| ,**TO**=m

| ,**OVERWR**ITE-**DISK**=***ANY** / list-poss(32): vsn

| ,**UNIT**=***NO** / list-poss(32): mn(...)

| mn(...)

| | **SUBUNIT**=***NO** / mn

,**DEV**ICE-**TYPE**=device-type / ***BY-UNIT** [1)]

,**FORM**AT=***NO** / ***YES**

[1)] Specification of the DEVICE-TYPE operand is no longer recommended. The DEVICE-TYPE should be determined implicitly by the UNIT. The operand is no longer displayed in guided dialog.

DISK-NUMBER
Specifies which disk or which range of a pubset or volume set is to be initialized.

DISK-NUMBER=n(...)
Number of the disk within the pubset or volume set. The permissible range depends on the selected format of the pubset or volume set ID.

pubset-id1, volumeset-id1: (the identifier is 1 character long)

The number n of the disk in the pubset must consist of two digits. it is added to the new VSN of the disk to be processed as characters 5 and 6.
Value: 00 n 31

The VSN of the disk thus has the following format: **PUBpnn** where **p** is the singlecharacter pubset or volume set identifier.

pubset-id2, volumeset-id2: (the identifier is 2 to 4 characters long).

The number of the disk in the pubset depends on the allocation unit:

ALLOCATION-UNIT = 6:

The number of the disk has one character.
Permissible value range for z: 0 - 9, A - V

The VSN of the disk can thus have the following format:

pp.00z or **ppp.0z** or **pppp.z**
where **p** is the two- to four-digit pubset or volume set identifier.

ALLOCATION-UNIT > 6:

The number of the disk may have up to three characters.
Permissible value range for x,y,z: 0 - 9, A - Z, where:

1) 0< 1 <...< 9 < A < B...< Z < 10

2) 10<11...<99<AA<BB...<ZZ<100

3) 100<101...<999<AAA<BBB...<ZZZ

If less than three digits are specified, the disk number is padded with zeros from the left.

The VSN of the disk can thus have the following format:

1) **pppp.z** or 2) **ppp.yz** or 3) **pp.xyz**
where **p** is the two- to four-digit pubset or volume set identifier.

A list is permitted. It may contain up to 32 elements. The OVERWRITE-DISK operand always refers to the associated ‘n’ specification.

OVERWRITE-DISK
Specifies whether the VSN is to be checked prior to initialization.

OVERWRITE-DISK=*ANY
The initialization takes place without a prior VSN check by SIR. In this case, VOLIN will later issue the NVL0024 question. I.E., the VSN check is conducted manually, in interactive mode by the caller, in batch mode by the operator.

OVERWRITE-DISK=vsn
VSN of the disk to be mounted. VOLIN initializes this disk only when the value `vsn` coincides with the VSN of the disk.

UNIT
Introduces a list of disk devices.

UNIT=*NO
Specifies that no disk mnemonics are to be predefined.

UNIT=mn(...)
Mnemonic name of the first disk (MAINUNIT disk) of a disk pair or, in the case of non-DRV pubsets, the only disk to be used.

SUBUNIT=*NO
No second disk is specified.

SUBUNIT=mn
Mnemonic name of the second disk (SUBUNIT disk) of a disk pair. This may only be specified with DRV.

DISK-NUMBER=*RANGE(...)
A number of disks within a range are to be processed. The range is delimited by means of the FROM and TO operands.

FROM=n
Start of the range.

TO=m
End of the range.

> n and m are governed by the same rules as for the n specification in the `DISK-NUMBER` operand. n must be smaller than m.
>
> Irrespective of the notation, the value range for both n and m is 0..9, A-Z. This means, for example, that the specification `FROM=1,TO=14` covers a total of 40 disks (1-9, A-Z, 10-14). So if you wish to create a pubset containing 14 disks (1-14), you will need two statements:
>
> ```
> //CREATE-VOLUME DISK-NUMBER=*RANGE(FROM=1,TO=9,UNIT=(...))
> ```
>
> ```
> //CREATE-VOLUME DISK-NUMBER=*RANGE(FROM=10,TO=14,UNIT=(...))
> ```

OVERWRITE-DISK
Specifies whether the VSN is to be checked prior to initialization.

OVERWRITE-DISK=*ANY
The initialization takes place without a prior VSN check by SIR. In this case, VOLIN will later issue the NVL0024 question. I.E., the VSN check is conducted manually, in interactive mode by the caller, in batch mode by the operator.

OVERWRITE-DISK=list-poss(32): vsn
A list of VSNs is only admissible if several disks within an area are to be processed with the *RANGE operand. The number of list elements must match the number of disks in the range. The statement is rejected if this is not the case. If a DRV pubset/volume set is formed, each VSN specified must be present twice in the system.

UNIT
Introduces a list of disk devices.

UNIT=*NO
Specifies that no disk mnemonics are to be predefined.

UNIT=mn(...)
Mnemonic name of the first disk of a disk pair or, in the case of non-DRV pubsets, the only disk to be used.

SUBUNIT=*NO
No second disk is specified.

SUBUNIT=mn
Mnemonic name of the second disk of a disk pair. This may only be specified with DRV.

FORMAT
Specifies the type of processing.

FORMAT=*YES

The disk is overwritten with a specific bit pattern during the initialization run (default value), i.e. all files are deleted.

FORMAT=*NO

The disk is not overwritten with a bit pattern, instead it is only initialized. A change in the operating mode with regard to the user of PAM keys must apply to the whole pubset/volume set.

Notes

1. During initialization, every disk of the pubset to be processed must be specified in a CREATE-VOLUME statement, since they must all be anchored with VSN and device type in the SVL of the system disk of the pubset (Pubres).
2. In the case of a pubset extension (operand ACTION=*EXTEND in the DECLARE-PUBSET statement), the disks specified in this CREATE-VOLUME statement are added to the pubset. If a disk that already belongs to the pubset is specified, this statement is ignored.
3. In contrast to the VOLIN statements or to the INITIALIZE-PRIVATE/PUBLIC-VOLUME statements, it is possible to process a number of disks with a single CREATE-VOLUME statement in this case. However, as a precondition for this, all disks listed in a statement must either have the same device type or be specified via UNIT and be handled the same way by the FORMAT parameter.
4. When OVERWRITE-DISK=*ANY is specified, the mounted disk is only overwritten if system administration agrees to this in a message. In this message, the VSN of the mounted disk should be specified.
5. During creation/extension of a pubset with dual-recording disks (“mirror” disks), two disks are always initialized together. If the operand OVERWRITE-DISK=vsn is specified, there must in this case be two disks with the specified VSN.
6. If there is more than one disk to be initialized with a specific bit pattern, they can, due to time constraints, be initialized in parallel SIR tasks using the statement INITIALIZE-PUBLIC-VOLUME since initializations with a specific bit pattern are processed sequentially when CREATE-VOLUME with FORMAT=*YES is used.

13.6.4.8 DECLARE-PUBSET - Define pubset and type of processing

This statement is used to define the pubset to be processed, as well as the type of SIR processing.

The definition covers the following elements:

- the pubset type
  This defines whether an SF pubset (single-feature pubset) or an SM pubset (systemmanaged pubset) is to be set up.
- the pubset identifier
  This identifies the pubset. All the pubsets of a particular system are recorded in the MRS catalog. In the case of SF pubsets, the pubset identifier is a syntactic component of the VSN of all disks belonging to that pubset.
- the system identifier
  This distinguishes the system from other systems when the SF pubset is being used as the home pubset.
- the operating mode of the SF pubset
  This determines whether PAM keys are used on the disks belonging to the SF pubset.
- the allocation unit
  This is the unit with which DMS allocates space for files on the disks of the SF pubset.
- the physical block size on all disks belonging to the SF pubset.
- whether disks and files with a capacity in excess of 32 Gbytes are permitted.
- the identifier of the control volume set of an SM pubset which contains administration information.

DECLARE-**PUB**SET

```
PUBSET-TYPE=*SYSTEM-MANAGED(...) / *SINGLE-FEATURE(...)
    *SYSTEM-MANAGED(...)
        | PUBSET=pubset-id
        | ,ACTION= *INST ALL(...) / *EXTEND(...)
        |     *INSTALL(...)
        |         | CONTROL-VOLUME-SET=catid
        |         | ,LARGE-DISKS-ALLOWED= *NO / *YES(...)
        |         |     *YES(...)
        |         |         | LARGE-FILES-ALLOWED= *NO / *YES
        |     *EXTEND(...)
        |         | CONTROL-VOLUME-SET=catid
    *SINGLE-FEATURE(...)
        | PUBSET=pubset-id
        | ,ACTION= *INST ALL(...) / *EXTEND / *IPL
        |     *INSTALL(...)
        |         | FORMAT= *K / *NK(...)
        |         |     *NK(...)
        |         |         | PHYSICAL-BLOCK-SIZE= *2K (...) / *4K(...)
        |         |         |     *2K(...)
        |         |         |         | ALLOCATION-UNIT= 6 / 8 / 64
        |         |         |     *4K(...)
        |         |         |         | ALLOCATION-UNIT= 8 / 64
        |         | ,AVAILABILITY= *NORM AL / *DRV
        |         | ,LARGE-DISKS-ALLOWED= *NO / *YES(...)
        |         |     *YES(...)
        |         |         | LARGE-FILES-ALLOWED= *NO / *YES
        | ,SYS-ID=*STD / *CAT-ID / sys-id
```

PUBSET-TYPE
Defines the pubset type to be set up.

PUBSET-TYPE=*SYSTEM-MANAGED(...)
An SM pubset is to be set up.

PUBSET=pubset-id
Identifier of the pubset (1 to 4 alphanumeric characters, must not begin with the string PUB).

ACTION
Specifies the SIR operating mode.

ACTION=*INSTALL(...)
The pubset specified by pubset ID is to be created.

CONTROL-VOLUME-SET=catid
Specifies the catalog ID of the volume set to be created as the control volume set of the SM pubset. catid must be a valid catalog ID.

LARGE-DISKS-ALLOWED=*NO / *YES(...)
Specifies whether disks with a capacity in excess of 32 Gbytes may be added to a volume set of this pubset (*YES(...)) or not (*NO).

LARGE-FILES-ALLOWED=*NO / *YES
Specifies whether the files of a pubset with large disks, which exceed 32 Gbytes in length are permitted (*YES) or not (*NO).

ACTION=*EXTEND(...)
The pubset identified by the pubset ID is to be extended.

CONTROL-VOLUME-SET=catid
Specifies the catalog ID of the control volume set.
The control volume set can only have a valid catalog ID as its name.

PUBSET-TYPE=*SINGLE-FEATURE (...)
An SF pubset is to be set up.

PUBSET=pubset-id
Identifier of the pubset. The pubset identifier has two formats. For SF pubsets, this is an intrinsic part of the format of the VSN of all disks belonging to the pubset.

pubset-id1:
The identifier is 1 (alphanumeric) character long. If is the fourth character in the VSN of every disk belonging to the SF pubset. Format of the VSN:
PUB**p**xx where:

p	= pubset-id1
xx	= number of the disk in the pubset. Value range: 00 - 31

pubset-id2:
The identifier is 2 to 4 (alphanumeric) characters long. It is located at the beginning of the VSN. Format of the VSN:
pp.xyz or **ppp**.yz or **pppp**.z where:

p	= character from pubset-id2
x,y,z	= number of the disk in the pubset

Value range for allocation unit > 6

x,y,z:	0 - 9, A - Z.

Value range for allocation unit = 6

x,y:	The positions for x and y contain 0;
z:	0 - 9, A - V.

The values W through Z are not permitted for z, since a pubset is not allowed to contain more than 32 disks.

pubset-id2 must not begin with the string PUB.

ACTION
Specifies the SIR operating mode.

ACTION=*INSTALL(...)
The pubset specified by pubset ID is to be created.

FORMAT
Defines the disk format and thus the operating mode of the pubset with respect to the use of PAM keys.

FORMAT=*K
The PAM key is made available. The pubset consists of K disks.
K files with BLKCTRL=PAMKEY are permitted.

FORMAT=*NK(...)
The disks of the pubset are formatted without PAM keys (NK disks).
Only NK files with BLKCTRL=DATA / NO are possible.

PHYSICAL-BLOCK-SIZE
Specifies the physical block size on the NK disks of the pubset.

PHYSICAL-BLOCK-SIZE=*2K(...)

NK2 disk (default value)
The minimum transfer unit between disk and memory is 2 Kb. This means a write or read job always moves 2 Kb (or a multiple thereof).

PHYSICAL-BLOCK-SIZE=*4K(...)
NK4 disk The minimum transfer unit is 4 Kb.

ALLOCATION-UNIT
Specifies the size of the unit used by DMS for allocation of space for files on disks of the pubset.

ALLOCATION-UNIT=6
The unit (size of the smallest file) is 6 Kb. This may be specified only for pubsets with NK2 disks and is the default value for these disks.

ALLOCATION-UNIT=8 / 64
The unit (size of the smallest file) should be 8 or 64 Kb in size. This may be specified for pubsets with NK2 or NK4 disks. 8 Kb is the default value for NK4 disks.

AVAILABILITY
Defines the availability of the pubset.

AVAILABILITY=*NORMAL
No DRV disks are created (usual availability).

AVAILABILITY=*DRV
A pubset with DUAL-RECORDING is created.

LARGE-DISKS-ALLOWED=*NO / *YES(...)
Specifies whether disks with a capacity in excess of 32 Gbytes may be added to the pubset (*YES(...)) or not (*NO).

LARGE-FILES-ALLOWED=*NO / *YES
Specifies whether the files of a pubset with large disks, which exceed 32 Gbytes in length are permitted (*YES) or not (*NO).

ACTION=*EXTEND
The pubset identified by the pubset ID is to be extended.

ACTION=*IPL
IPL files are to be anchored in the SVL.

SYS-ID
System identifier used when a pubset is used as the home pubset of the system in order to distinguish it from other systems in connection with the allocation of shareable devices (MTC).

SYS-ID must be specified if ACTION=*INSTALL is specified. If ACTION=*EXTEND / *IPL applies, a corresponding message is output and SYS-ID is ignored.

SYS-ID=*STD
The default value is 250.
A pubset with the default value cannot be made bootstrappable.

SYS-ID=*CAT-ID
If the pubset-id1 format of the pubset identifier is used, SYS-ID=*CAT-ID must be specified. No other predefinition is possible here.

SYS-ID=sys-id
Numeric value in the range 65 - 192.

Notes

1. If only a private disk is to be turned into an IPL disk with SIR, no DECLARE-PUBSET statement may be specified.
2. Requirements that must be met if increased availability with DUAL-RECORDING (`AVAILABILITY=*DRV`) is to be achieved:

   In order to obtain mirror disks with identical contents during creation, the subsystem DRV must first be loaded and after this, the command
   `/SET-DRV PUBSET=pubset-id,RECORDING-MODE=*DRV` must be used to set DRV mode for the pubset.

   During initialization of the disks, two successive disks are in each case processed exclusively for the same VSN. The operator must respond with care to the MOUNT messages, in order to avoid confusion. After this, the disks are allocated only as shareable and all operations are executed in dual-recording mode.

Example

```
DECLARE-PUBSET PUBSET-TYPE=*S-F(030,*IPL)

DECLARE-PUBSET PUBSET-TYPE=*S-F(030,*INSTALL(*NK(*4K(64))),*STD)
```

13.6.4.9 DELETE-IPL-FACILITY - Delete IPL capability and IPL files of disk

The DELETE-IPL-FACILITY statement is used to delete the IPL capability and IPL files of a given disk. This disk can then no longer be used to initialize the system.

DELETE-IPL-FACILITY
DISK=***PUB**LIC(...) / ***PRIV**ATE(...) ***PUB**LIC(...) | **DISK**=vsn ***PRIV**ATE(...) | **DISK**=vsn | ,**DEV**ICE**-TYPE**=**STDDISK** / device-type

DISK
Defines the disk to be processed.

DISK=*PUBLIC(...)
Specifies that the IPL capability of a public disk is to be deleted.

DISK=vsn
VSN of the public disk

DISK=*PRIVATE(...)
Specifies that the IPL capability of a private disk is to be deleted.

DISK=vsn
VSN of the private disk

DEVICE-TYPE=STDDISK
Default device type of the private disk.

DEVICE-TYPE=device-type
Device type of the private disk.

Notes

1. This function can also be used on disks belonging to an SM pubset. Even though IPL capability cannot be configured for disks belonging to an SM pubset, IPL files can be transferred to a volume set by means of migration from an SF pubset.
   If the disk belongs to a volume set, this volume set must be part of an SM pubset.
2. The pubset containing the disk whose IPL capability is to be deleted must be addressable (imported).
3. SIR cannot delete IPL files with additional file protection. A message to this effect is output.
4. IPL files on private disks cannot be deleted if files with the same name, but a different location have been imported into the default pubset. A message to this effect is output.

13.6.4.10 END - Terminate SIR

The END statement is used to terminate SIR. In procedure mode, the specified statements are still processed as if the statement START-EXECUTION had been specified.

END

13.6.4.11 END-VOLUME-SET-DECLARATION - Terminate statement sequence for volume set

The END-VOLUME-SET-DECLARATION statement terminates the statement sequence for a volume set. Together with the BEGIN-VOLUME-SET-DECLARATION, which initiates the declaration, it embraces a statement block which can be repeated for each volume set to be set up.

END-VOLUME**-SET-DECL**ARATION

13.6.4.12 INITIALIZE-PRIVATE-VOLUME - Initialize private disks

The INITIALIZE-PRIVATE-VOLUME statement is used to initialize a private disk. The statement is executed immediately rather than being transferred to the current sequence of statements.

INITIALIZE**-PRIV**ATE**-VOL**UME
VOLUME=vsn ,**UNIT**=mn ,**OVERWR**ITE**-DISK**=***ANY** / vsn **,FORM**AT=***UNCHA**NGED / ***K** / ***NK** ,**F1-LAB**EL**-SIZE**=1000 / <integer 1..32766> ,**CAPACITY**=***STD** / <integer 1000..2 147 483 647> ,**FUNC**TION=***CRE**ATE**-LAB**ELS**-ONLY** / ***FORM**AT**-DISK**

VOLUME=vsn
Specifies the VSN of the disk to be initialized.

UNIT=mn
Two- or four-digit mnemonic of the device on which the disk is mounted.

OVERWRITE-DISK
Specifies whether the disk is to be checked before being used.

OVERWRITE-DISK=*ANY
This disk is initialized without the existing (old) VSNs being checked.

OVERWRITE-DISK=vsn
This disk is only initialized if the existing VSN tallies with that specified by VOLUME=vsn.

FORMAT
Defines the disk format of the disk to be initialized.

FORMAT=*UNCHANGED
This disk is initialized with its previous format; it is initialized in format K if it is unformatted.

FORMAT=*K
The disk is initialized with format K.

FORMAT=*NK
The disk is initialized with format NK2, allocation unit 6.

F1-LABEL-SIZE=1000 / <integer 1..32766>
Specifies the size of the F1 label in PAM blocks. 1000 PAM blocks are reserved by default. The permitted size of the F1 label ranges from 1 to 32,766 PAM blocks. The specified value is rounded up to allocation unit size.

CAPACITY=
Specifies the maximum disk capacity. This operand is evaluated only for disk type `A5` (D3435). Disks of other types are always initialized with their maximum capacity. This operand is mainly used for disk migration.

CAPACITY=*STD
The disk is made available with its maximum capacity.

CAPACITY=<integer 1000..2 147 483 647>
The disk is made available with the capacity specified (in PAM pages). The specification may not exceed the maximum capacity of the disk.

FUNCTION
Specifies whether the disk is to be initialized with a specific bit pattern.

FUNCTION=*CREATE-LABELS-ONLY
Only the labels are generated

FUNCTION=*FORMAT-DISK
The disk is initialized with a specific bit pattern

13.6.4.13 INITIALIZE-PUBLIC-VOLUME - Initialize public disks

The statement INITIALIZE-PUBLIC-VOLUME is used to initialize a public disk. The statement is executed immediately rather than being transferred to the current sequence of statements.

```
INITIALIZE-PUBLIC-VOLUME

 VOLUME=vsn
 ,UNIT=mn
 ,OVERWRITE-DISK=*ANY / vsn
 ,FORMAT=*UNCHANGED / *K / *NK(...)
     *NK(...)
         |   PHYSICAL-BLOCK-SIZE=*2K(...) / *4K(...)
         |       *2K(...)
         |           |   ALLOCATION-UNIT=6 / 8 / 64
         |       *4K(...)
         |           |   ALLOCATION-UNIT=8 / 64
 ,CAPACITY=*STD / <integer 1000..2 147 483 647>
 ,FUNCTION=*CREATE-LABELS-ONLY / *FORMAT-DISK
```

VOLUME=vsn
Specifies the six-digit VSN of the disk to be initialized.

UNIT=mn
Two- or four-digit mnemonic of the device on which the disk is mounted.

OVERWRITE-DISK
Specifies whether the disk is to be checked before being used.

OVERWRITE-DISK=*ANY
This disk is initialized without the existing (old) VSNs being checked.

OVERWRITE-DISK=vsn
This disk is only initialized if the existing VSN tallies with that specified by VOLUME=vsn.

FORMAT
Defines the disk format of the disk to be initialized.

FORMAT=*UNCHANGED
This disk is initialized with its previous format; it is initialized in format K if it is unformatted.

FORMAT=*K
The disk is initialized with format K.

FORMAT=*NK(...)
The disk is initialized with format NK.

PHYSICAL-BLOCK-SIZE
Defines the physical block size on the NK disk.

PHYSICAL-BLOCK-SIZE=*2K(...)
NK2 disk (default value)
The minimum transfer unit between disk and memory is 2 Kb.

ALLOCATION-UNIT=6 / 8 / 64
The unit (size of the smallest file) should be 6 Kb (default value), 8 or 64 Kb in size.

PHYSICAL-BLOCK-SIZE=*4K(...)
NK4 disk The minimum transfer unit is 4 Kb.

ALLOCATION-UNIT=8 / 64
The unit (size of the smallest file) should be 8 or 64 Kb in size.

CAPACITY=
Specifies the maximum disk capacity. This operand is evaluated only for disk type `A5` (D3435). Disks of other types are always initialized with their maximum capacity. This operand is mainly used for disk migration.

CAPACITY=*STD
The disk is made available with its maximum capacity.

CAPACITY=<integer 1000..2 147 483 647>
The disk is made available with the capacity specified (in PAM pages). The specification may not exceed the maximum capacity of the disk.

FUNCTION
Specifies whether the disk is to be initialized with a specific bit pattern.

FUNCTION=*CREATE-LABELS-ONLY
Only the labels are generated

FUNCTION=*FORMAT-DISK
The disk is initialized with a specific bit pattern

13.6.4.14 MODIFY-IPL-VOLUME - Modifying an IPL disk

This statement can be used to modify an IPL disk.

MODIFY-**IPL-VOL**UME

DISK-NUMBER=n / ***PRIV**ATE(...)

***PRIV**ATE(...)

| **DISK**=vsn

| ,**DEV**ICE-**TYPE**=**STDDISK** / device-type

,**IPL-FILE**=***UNCHA**NGED / ***STD** / name

,**IPL-REP-FILE**=***UNCHA**NGED / ***STD** / name

,**SLED-REP-FILE**=***UNCHA**NGED / ***STD** / name

,**HSI-TYPE**=***BY-OWN-HSI** / ***390** / ***X86** [1)]

[1)] The *7500 operand value can still be specified for reasons of compatibility.

DISK-NUMBER
Specifies the IPL disk that is to be modified. Depending on the format of the pubset ID (see the DECLARE-PUBSET statement), the number of the disk in the pubset must be specified (see the CREATE-VOLUME statement).

DISK-NUMBER=n
One- to three-digit number of the IPL disk in the pubset (see "CREATE-VOLUME - Initialize disks").

DISK-NUMBER=*PRIVATE(...)
Specifies a private IPL disk that is to be modified.

DISK=vsn
Volume serial number of the private IPL disk.

DEVICE-TYPE=STDDISK
Default device type for the private IPL disk.

DEVICE-TYPE=device-type
Device type of the private IPL disk.

IPL-FILE
Defines the IPL load module to be anchored in the SVL.

IPL-FILE=*UNCHANGED
The load module remains unchanged. It has already been anchored.

IPL-FILE=*STD
Default file to be anchored.

IPL-FILE=name
Name of the file to be anchored (see notes).

IPL-REP-FILE
Defines the IPL REP file to be anchored in the SVL.

IPL-REP-FILE=*UNCHANGED
This file remains unchanged. It has already been anchored.

IPL-REP-FILE=*STD
Default file to be anchored.

IPL-REP-FILE=name
Name of the file to be anchored (see notes).

SLED-REP-FILE
Defines the SLED REP file to be anchored in the SVL.

SLED-REP-FILE=*UNCHANGED
This file remains unchanged. It has already been anchored.

SLED-REP-FILE=*STD
Default file to be anchored.

SLED-REP-FILE=name
Name of the file to be anchored (see notes).

HSI-TYPE
Specifies the HSI for which IPL capability is to be established (see notes).

HSI-TYPE=*BY-OWN-HSI
IPL capability is established for the HSI on which SIR is running.

HSI-TYPE=*390
IPL capability is established for the /390 HSI (/390 servers).

HSI-TYPE=*X86
IPL capability is to be established for the X86 HSI (x86 servers).

Notes

1. The IPL capability of disks is described in detail on "SF pubsets".
2. If *STD is specified for a file, it must either be installed using IMON or be in the pubset to be processed.
3. If DISK-NUMBER=*PRIVATE is specified, the IPL files must be in the home pubset in a compatible format.
4. The files specified by “name” do not have to be cataloged in the pubset. They are copied to the disk to be processed, where they are cataloged anew with the extension <.DSKxxx> or <.vsn> for private disks.
5. If at least one disk of the pubset is converted to an IPL disk, SIR creates the file SYS.NSI.SAVEREP in order to save system corrections during system initialization.
6. The user can specify several, noncontradictory statements for the same disk by specifying the operands for the file names individually:

   ```
   MODIFY-IPL-VOLUME DISK-NUMBER=03,IPL-FILE=IPLDATEI
   ```

   ```
   MODIFY-IPL-VOLUME DISK-NUMBER=03,IPL-REP-FILE=IPLREP
   ```

   `IPLREP` and `IPLDATEI` must be in the pubset to be processed.
7. The user can create an IPL disk with the aid of MODIFY-IPL-VOLUME by specifying at least one IPL file. If the IPL file is neither anchored nor specified, the processing is discontinued.
8. If the REP files for IPL and SLED are neither anchored nor specified, a suitable message is issued and processing is continued. The disk is then IPL capable.
9. In order to verify the consistency of the IPL file specified in the statement, the version of the included load module will be checked. The user is responsible for ensuring consistency between REP files and the IPL file.
10. All anchored files are given the file attributes BACKUP=E, DESTROY=*YES, *ACCESS=*READ and MIGRATE=*FORBIDDEN.
11. SIR does not support file protection with BASIC-ACL for IPL files. If the user has protected one of these files with BASIC-ACL, the protection must be canceled to enable SIR to modify the IPL files.
12. A disk can only ever be bootstrappable for one HSI. This means that if IPL capability is established for HSI-TYPE=*X86 using MODIFY-IPL-VOLUME, any IPL capability which previously existed for a /390 HSI is lost.
13. The HSI-TYPE operand should only be specified if IPL disk capability is not to be established for the HSI on which SIR is currently running.
14. The statement is rejected for disks of pubsets, which cannot be used as home pubsets:
    - pubsets with `SYS-ID=*STD`
    - pubsets with `PHYSICAL-BLOCK-SIZE=*4K`
    - SM pubsets

14 SMPGEN Conversion of SF pubsets to SM pubsets

Version: **SMPGEN V21.0A**

Privileges: **STD-PROCESSING** (for nonprivileged functions)
TSOS (for privileged functions)

SMPGEN V21.0 runs as of BS2000 V21.0 and requires TSOS privilege (except for the restricted check function).

SMPGEN offers the following functions:

- Creation of new SM pubsets from existing SF pubsets
- Supplementing existing SM subsets by one or more SF pubsets
- Checking the prerequisites for conversion (check function)

The utility routine SMPGEN (System-Managed Pubset GENeration) convert one or more single-feature pubsets (SF pubsets) to a new or an already existing SM pubset. The files, file entries and metadata on the pubset are retained in the process or are transferred to the SM pubset in a valid form.

SMPGEN also provides a check function which systems support can use up front to check whether the specified pubsets satisfy the requirements for successful conversion. This check function can also be called in restricted form for nonprivileged users, albeit only for use under the ID of the caller. In this way, the user is able to obtain information on any preparatory work which he or she must carry out.

The utility routine SMPGEN consists of the nonprivileged subsystem SMPGEN-U (SMPGEN utility) and the privileged subsystem SMPGEN-S (SMPGEN system) which is called internally.

Please refer to the manual entitled “System-Managed Storage” [8 (Related publications)] for a description of the structure of SF and SM pubsets. Details on the important attributes and behavior of SM pubsets and volume sets that are required in order to gain an understanding of SMPGEN can be found in the section “Attributes of SM pubsets and volume sets”.

14.1 Typical scenario

When converting SF pubsets to SM pubsets, it is generally not sufficient just to call the SMPGEN function CREATE- or MODIFY-SM-PUBSET. Rather, systems support has the task of making sure that the conversion is as easy as possible for the user. After the SMPGEN run, various work must be performed before the SM pubset can be operated as required.
In particular, the pubsets generated by SMPGEN are not yet supported by HSMS. in order to be able to process migrated files, it is first necessary to set up an HSMS environment. Depending on the complexity of the existing HSMS environments, there are several ways of modifying the HSMS environment; these require special preparation; for details please refer to the “System-Managed Storage” manual [8 (Related publications)].

The actions which can be necessary and useful when converting to SM pubsets are described below:

- If several SF pubsets are to be combined, it is useful to perform a consistency check first. This determines whether or not conversion is possible.
- If conversion is possible, the users are informed that references to the old pubset IDs in all BS2000 procedures must be replaced by references to the new pubset ID.

  Normally, the users are not really affected greatly by this, since the files are addressed via the standard catalog ID in the user catalog.
  Data inventory systems are not really affected either, since they use a specific addressing logic to hide the location of the file from the user.
  If applications that have to work with particular pubsets are present, they must in future specify, along with the pubset ID of the SM pubset, the relevant volume set ID or a storage class that references the desired volume set when storing the files. However, it is not possible to specify this ID unless the relevant user ID possesses the physical allocation right in the pubset. This right must be assigned explicitly following conversion.

  If only one SF pubset is to be converted, it is useful to first rename this pubset with PVSREN and to specify the old pubset ID as the pubset ID of the SM pubset. This ensures that the catalog ID in the file name is not changed and that the JCL of the user does not need to be adapted. Similarly, the catalog ID may be retained as the standard catalog ID in the user catalog.

  The procedure is analogous when several SF pubsets are to be combined, but one of them has to play a special role in the system; it can be expedient to rename this pubset and specify its old pubset ID as the pubset ID of the SM pubset.

  The control volume set should be as reliable as possible and should therefore also not be too large. It must, however, provide enough space for the new catalogs to be created (see section “Creating new SM pubsets”).
- If a consistency check has already been carried out and conflicts have been detected in the file names or in the guards, systems support must request the user to resolve these conflicts.
- Systems support is responsible for resolving name conflicts in system files. System files anchored in the SVL (IPL, SLEDSAVE etc.) should not be deleted; they should either be renamed or be deleted by using the SIR statement DELTE-IPL-FACILITY.
  To resolve name conflicts in (HSMS) migration directories, rename the directories and adapt the associated archive definitions to match the new directory names.
- If necessary, paging files must be deactivated using `/REDUCE-PAGING-AREA`.

- Systems support checks whether snapsets exist for the SF pubsets. If this is the case, they delete the snapsets.
- Systems support prepares the inputs for the SMPGEN run and possibly a procedure for importing the SM pubset.
- If the SM pubset is to be operated with HSMS support (always necessary if several migrated files are present), systems support prepares the conversion or adaption of the HSMS environment.
  (The catalog entries of the migrated files are retained after conversion; however, it is not possible to recall these files, until HSMS support has been established. Similarly, there is no backup environment. Direct access to previous backups is not possible.)
- It may be useful to perform a further consistency check in offline mode. The users should, if necessary, be informed of any inconsistencies that have to be resolved.
- When all inconsistencies have been resolved, conversion to the SM pubset may be started at a time agreed with the users, at which the SF pubsets are not being used.

  Some final preparatory work may be required for conversion of the HSMS environment, e.g. relocation of the migrated files on the S1 level to S2 to ensure that the conversion will be successful.
  First, a full logical backup and an FDDRL backup of the pubsets is made. Then the pubsets are converted using the SMPGEN statement CREATE- or MODIFY-SYSTEM-MANAGED-PUBSET.

  If there are still inconsistencies at this point, or if new one are encountered, they are listed; conversion is not performed, and the pubsets can still be imported as SF pubsets.
- Following successful conversion, the standard catalog IDs that correspond to the volume set IDs should be replaced by the pubset ID of the SM pubset in the user catalogs of all systems in which the pubset is to be imported.
- The program name must be modified in the guards of other pubsets which reference a program in the new SM pubset. Systems support can provide the users with a corresponding SDF-P procedure.
- Now the new SM pubset can be imported normally, for which ACTUAL-JOIN=*STD must apply.
  An extended SM pubset must be reimported in order to perform outstanding recovery measures after the SMPGEN run. Here, too, ACTUAL-JOIN=*STD (default) must be specified. SIZE-TOLERANCE=*YES must, if required, be entered previously in the MRSCAT entry of the SM pubset (with `/MODIFY-PUBSET-CACHE-ATTRIBUTES`); it then applies to all volume sets.
- If the SM pubset is to contain volume sets with USAGE=*WORK, these must be added to the imported pubset using `/MODIFY-PUBSET-DEFINITION-FILE` and put into operation with `/MODIFY-PUBSET-PROCESSING`.
- Now, if desired, HSMS support and the HSMS environment can be set up or adapted.
- The quotas and pubset-wide authorizations can be adapted to the special conditions, e.g. by restricting the work quotas.
  Groups can be generated or the group constellation completed. If desired, the groupspecific quotas may be changed.
- If desired, the standard file format (FILE FORMAT) may be modified.
- If guard condition faults are notified, the guard owners must check the relevant guard entries and adapt the new pubset ID as required.
- If software was installed on one of the converted SF pubset using IMON, and the catalog ID of the installed files has changed, the software configuration inventory (SCI) must be modified with `/MODIFY-IMON-SCI`. The name of the SF pubset prior to conversion must be specified for OLD-NAME, and the name of the SM pubset specified for NEW-NAME.
- After successful conversion of the pubset, systems support should inform the user.

14.2 Requirements for operation of SMPGEN

Requirements of the operating system and the job task

- If a guard catalog exists in at least one SF pubset (i.e. the file $TSOS.SYSCAT.GUARDS), the GUARDS subsystem must be loaded.
- Output to an S variable is supported only by the product SDF-P, which must be purchased separately. The VAS subsystem must also be loaded.

In addition to these requirements, the following conditions must also be observed for the MODIFY- and CREATE-SYSTEM-MANAGED-PUBSET statements, depending on the function called:

- The $TSOS.SYSACL.FILE file may not be present on any of the pubsets as the SRPMFACL (FACS) subsystem has not been avaialble since BS2000/OSD-BC V5.0.
- An MRSCAT entry must be present for each pubset.
- The pubsets must not be imported in the local system nor in another system. There must be no import of export (`/IMPORT-PUBSET` or `/EXPORT-PUBSET`) running for them.
- No files on Net-Storage volumes may be listed in the pubset's catalog. These catalog entries must be deleted before a SMPGEN run (`/EXPORT-FILE` or, see next paragraph, `/REMOVE-NET-STORAGE-VOLUME`, operand `FILES-ON-VOLUME=*EXPORT`).
- All Net-Storage volumes must be removed from the pubsets (`/REMOVE-NET-STORAGE-VOLUME`). On the other hand, Net-Storage volumes may be assigned to the existing SM pubset.
- When the function is called by a nonprivileged user (for which the value CHECK-NAME-CONSISTENCY (PUBSET-STATE=*IMPORTED) is assumed implicitly), the SF pubsets must have the state “local accessible”, i. e. imported locally. If USER-ID=*ALL is specified, either the pubsets must be imported as system-exclusive or the local system must be the pubset master.

Requirements for the original pubsets (SF pubsets)

The following conditions apply to SF pubsets to be converted to an SM pubset:

- They must be capable of being imported into the current version of the operating system with ACTUAL-JOIN=*STD, i.e. without regeneration of the user catalog. This means that they have must already have been imported at least once.
- There may be no more that 8190 different user IDs in the SF pubset (= maximal number of users in an SM pubset).
- No more than 255 SF pubsets can be converted to an SM pubset.
- The paging files on the SF pubsets may not be active (i.e. they may need to be deactivated beforehand).
- No snapsets may exist for the SF pubsets (i.e. system administration may need to delete these beforehand). Empty snapset catalogs are deleted by SMPGEN if necessary.
- If the caller does not specify the operand S1-MIGRATED-FILES=*ALLOWED, the following also applies: there must be no migrated files cataloged on the S1 level. (i.e. any present must be migrated to S2 beforehand).

The first three conditions stated also apply to the check function.

Furthermore, it must be possible to set up new system files without encountering any name conflicts. The precise requirements are listed in the description of the statements.

Notes for Large-Objects pubsets

- If one of the original pubsets is a Large-Objects pubset, the SM pubset created will also be a Large-Objects pubset. The displays for this are located in both the MRSCAT of the SM pubset created and in the DMS entry of the control volume set.
- For security reasons, the displays are also stored in the DMS entry of the other volume sets.

Starting SMPGEN

The SMPGEN functionality is made available by the nonprivileged subsystem SMPGEN-U. SMPGEN-U should be loaded beforehand for performance reasons by systems support with `/START-SUBSYSTEM SUBSYSTEM-NAME=SMPGEN-U`.

SMPGEN is started during the session with `/START-SMPGEN`. The SMPGEN-S subsystem is called internally by SMPGEN-U. Both subsystems cannot be replaced by means of DSSM. Parallel product versions are not supported.

14.3 Check function (consistency check)

The check function (also called consistency check) is used to see whether the SF pubsets can be converted into an SM pubset that needs to be created or already exists.

The SMPGEN check function can be called with the following statements:

```
//MODIFY-SYSTEM-MANAGED-PUBSET
//CREATE-SYSTEM-MANAGED-PUBSET
```

Depending on whether the user is privileged (has the TSOS privilege) or not the check covers the following pubsets:

- nonprivileged users:
  Only pubsets on which the user's user ID is entered are checked. The caller is informed of name conflicts and may be able to resolve these.
  Pubsets which the caller cannot access because his/her user ID is not entered there are excluded from the consistency check. SMPGEN issues a message to this effect. Further processing (creation or extension of SM pubsets) is not possible.
- privileged users (TSOS user ID, TSOS privilege):
  The privileged user can start the checks on the pubsets (by specifying the OPERATIONAL-MODE=*CHECK-NAME-CONSISTENCY operand) either only for his/her user ID or for all user IDs (suboperand USER-ID=*OWN /*ALL).

The statement is executed either in the current task or in a new batch task that is created. In the latter case, the user can continue to work with SMPGEN in parallel or terminate the program (Operand EXECUTION-MODE=*SYNCHRONOUS/*ASYNCHRONOUS).

When the consistency check is performed also depends on the privilege:

- nonprivileged users:
  For the nonprivileged user, the checks are always carried out during the current pubset session. The result therefore contains random components, since it may happen that names are constantly being cataloged or deleted. There is no time synchronization for information that is to be compared.
- privileged users (TSOS user ID, TSOS privilege):
  The privileged user has the choice of performing the checks in the current pubset session or with exclusive access by SMPGEN (PUBSET-STATE=*IMPORTED/*NOT-IMPORTED operand).

  If the second variant (exclusive access) is selected, the public disks are allocated system-exclusively by SMPGEN. A flag will be set in the MRSCAT entries of the pubset to indicate that they are currently being processed by SMPGEN. These pubsets can no longer be imported during the checks. Their MRSCAT entries cannot be deleted during processing. The pubset status displayed by `/SHOW-PUBSET-ATTRIBUTES` is "inaccessible".

The consistency check also always runs (again) implicitly before SF pubsets are actually converted to SM pubsets.

14.3.1 What is checked?

The following checks are made:

- Check of the file and job variable names
- Check of the guard names
- Further checks (system files, migrated files, etc.)

Check of the file and job variable names

1. There must be no file or job variable names in the pubsets to be converted which would be longer than 54 characters when the new catalog ID is added (generally the future SM pubset ID).
   Similarly, there must be no file generation group names that would be longer than 47 characters when the new catalog ID is added (or no name of an individual file generation- with the generation number added - that would be longer than 54 characters).
2. Guard conditions that refer to the program name are checked to determine whether or not they would be valid when converted to an SM pubset; however, they are not automatically adapted. This can cause the following:
   - The catalog ID of the SM pubset is longer than that of the SF pubset and the implicit adaptation of the path name would cause the maximum permissible size of the condition range to be exceeded. In this case, the entire condition range remains unchanged and must be adapted by the owner of the guard.
   - The catalog ID of the program name contains wildcard symbols. In this case, both catalog IDs (that of the SF pubset and that of the SM pubset) are checked for wildcard catalog IDs. If at least one of the old catalog IDs satisfies the wildcard conditions, but the new SM pubset catalog ID does not, a conflict is reported. The program name containing the wildcard symbol is retained in each case.

   This point is an exception in that the pubset conversion can still be carried out; the guard entries can be adapted by the user at a later stage.
3. The SF pubsets to be converted must not contain two path names or job variable names that differ only with respect to the catalog ID.

   This also applies to system files, for example $TSOS.SYSTEM.MRSCAT. Systems support must ensure that these exist on only one of the pubsets.
   Exceptions to this are system files that are deleted or renamed automatically after conversion to an SM pubset.

   These are the files:

   $TSOS.SYS.PUBSET.CONFIG
   $TSOS.SYS.PVS.SHARER.CONTROL
   $TSOS.SYSCAT.GUARDS
   $TSOS.SYSEAM
   $TSOS.SYSPBN
   $TSOS.SYSSRPM
   $TSOS.SYSSRPM.BACKUP
   $TSOS.TSOSCAT
   $TSOS.TSOSCAT.#M nn (mit nn=00..99)
   $TSOS.TSOSCAT.#P nn (mit nn=00..99)
   $TSOS.TSOSCAT.#J nn (mit nn=00..99)
   $TSOS.TSOSCAT.$PFI
   $TSOS.TSOSCAT.$NLO
   $TSOS.TSOSCAT.$NLC
   $TSOS.TSOSJOIN

Check of the guard names

The pubset must not contain two guard names that differ only with respect to the catalog ID.

Further checks

The following names must not be present in any of the pubsets:

```
$TSOS.SYSCAT.STORCLS
$TSOS.SYSCAT.VSETLST
$SYSHSMS.SYSCAT.HSMS
```

None of the file names may start with $TSOS.SYSWRK.SMPGEN.
$TSOS.SYS.PUBSET.CONFIG files created by the user, or the files whose names commence with $TSOS.
TSOSCAT., must not be present in any of the pubsets.

If a certain number of name conflicts specified by the user is exceeded, the consistency check is aborted. (A name conflict here means the occurrence of file of job variable names that are too long, or the presence of files, job variables or guards in several pubsets.)

If the user is privileged and S1-MIGRATED-FILES=*ALLOWED is not specified, it is also checked that no files are migrated to S1 level. (This is not permissible; they must be migrated to S2 as required; failure counts as 1 conflict per pubset.)

> ¡ It is not checked whether there is enough space for the new SM pubset management files.

Notes

By default, not only the number of errors but also the detailed information is output in the event of an error; this information includes the file names, job variables and guards occur more than once and on which pubsets they are present.

Furthermore, all file and job variable names that are too long and all guard condition conflicts (see "Check of the file and job variable names", item 2) are output. The error lists are sorted according to user ID or pubset/catalog ID.
The information is output in list form by default. If output is to be made to a terminal, browsing on screen is possible. Alternatively, or in addition, the information can be stored in an S variable (operand value STRUCTURED-OUTPUT).

SMPGEN requires a correct SYSFILE environment. If the function is to be executed in a new batch task that is to be created, the default pubset of the runtime user ID (or the SPOOL pubset defined in ACS) must be imported. If SYSOUT or SYSLST outputs fails repeatedly, the SMPGEN function is terminated with an error.

14.3.2 Outputs

Error messages and the message relating to the start of the function, as well as the concluding success message are routed to the same output destination as the information on inconsistencies, i.e. to the output destination(s) specified by TEXT-OUTPUT.
In interactive mode, the message relating to the start of the function and the success message are also output to SYSOUT if TEXT-OUTPUT=*SYSLST is specified.

If the information on inconsistencies is not output in text form (TEXT-OUTPUT=*NONE), the error messages, the message relating to the start of the function, and the concluding success message are output to SYSOUT.

The following information is output:

1. Table of affected pubsets along with disk format, allocation unit (DRV, DUALCOPY), maximum I/O length and flag indicating whether a cache is defined. (For further information on the cache definition, use `/SHOW-PUBSET-PARAMETERS`.)

   Refer to mask 1 in section "Outputs to screen masks" for format details. This mask provides the user with the option of canceling the function by making an appropriate entry in the footer.
2. Logging messages:
   - Result message (consistency check successful or not).
     The result message is always output to SYSOUT in interactive mode, independent of the TEXT-OUTPUT operand.
   - In special cases, or in the event of termination on error, further messages may be issued first.
   - If the user is privileged, the size of the old pubset administration files is output in order to be able to estimate the space required for the new control volume set; and for specification of the control volume set ID also the free space required on the pubset to be used.
3. In the event of an error, or if warnings are issued, before the result message is output the following takes place (If STRUCTURE-OUTPUT was specified, the information is stored in S variables, regardless of the required scope of information).

   for **ERROR-INFOMATION=*SUMMARY**:
   - Number of files, job variables and guard name conflicts

     In the case of file generation groups, only the conflicts of the group names, not those in the individual file generations, are counted.
   - Number of file names and job variable names that are too long

     For file generation groups, only the generation group, not the individual file generations, are counted.
   - Number of guard conditions that could not be automatically adapted to the new pubset ID (warnings)

   for **ERROR-INFORMATION=*DETAIL**, detailed error information is output:
   - Lists of name conflicts

     For file generation groups, only the generation group is given; the conflicts for the individual file generations are not reported and not counted. See masks 3-8, starting on "Outputs to screen masks" for details of the list formats.
   - Lists of the file and job variable names that are too long

     For file generation groups, only the generation group is given (with addition: FGG); the conflicts for the individual file generations are not given and not counted. See masks 9-10, starting on "Outputs to screen masks" for details of the list formats.

- List of guard conditions that cannot be matched to the new pubset ID automatically. See mask 11 in section "Outputs to screen masks" for details of the list format.

Regardless of the required scope of information, the following error information is output for the privileged user (before all other error information):

- If one of the pubsets contains files migrated to S1 and S1-MIGRATED-FILES=*ALLOWED was not specified, an error message is returned (one message for each pubset affected).
- List of illegal files

  This refers to the files mentioned in the previous statement descriptions, which must not be cataloged in the pubsets because SMPGEN creates work files or new administration files under this name. See mask 2 in section "Outputs to screen masks" for details of the list format.

The following table explains the names and contents of the S variables. For further information on S variables, refer to the “Commands” manual [1 (Related publications)], volume 1.

Output information	Name of the S variable	T	Contents	Condition
Multiple occurrence of file name without catalog ID and user ID	var(*LIST).F.CONFL(*LIST).F	S	"<filename>	INF=*DETAIL
Number of checked pubsets on which the multiple file exists	var(*LIST).F.CONFL(*LIST). NUM-OF- PUBSET	I	"<integer>	INF=*DETAIL
Catalog ID of relevant pubset	var(*LIST).F.CONFL(*LIST). PUBSET(*LIST). CAT-ID	S	"<catid>	INF=*DETAIL
User ID of multiple file name	var(*LIST).F.CONFL(*LIST). USER-ID	S	"<userid>	INF=*DETAIL
Catalog ID of file name that is too long	var(*LIST).F-LEN.ERROR (*LIST).CAT-ID	S	"<catid>	INF=*DETAIL
File name that is too long without catalog ID and user ID	var(*LIST).F-LEN.ERROR (*LIST).F	S	"<filename>	INF=*DETAIL

Output information	Name of the S variable	T	Contents	Condition
Flag to indicate whether or not it is a file generation group (Y) or (N)	var(*LIST),F-LEN.ERROR(*LIST).FGG-IND	S	Y N	INF=*DETAIL
Number of characters in the file name that exceed the maximum of 54 (with catalog ID)	var(*LIST).F-LEN.ERROR(*LIST).NUM-OF-CHAR	I	" <integer>	INF=*DETAIL
User ID of the file name that is too long	var(*LIST).F-LEN.ERROR(*LIST).USER-ID		" <userid>	INF=*DETAIL
Number of file names that are too long	var(*LIST).F-LEN.NUM-OF-ERRORS	I	<integer>	INF=*DETAIL/*SUMMARY
Number of file name conflicts	var(*LIST).F.NUM-OF-CONFL	I	<integer>	INF=*DETAIL/*SUMMARY
Number of cases in which a guard condition is not automatically matched to the new pubset ID	var(*LIST).GUARD-COND.NUM-OF-WARN	I	<integer>	INF=*DETAIL/*SUMMARY
Catalog ID of the guard where guard condition does not match	var(*LIST).GUARD-COND.WARN(*LIST).CAT-ID	S	" <catid>	INF=*DETAIL
Name of the guard where the guard condition does not match, without catalog ID and user ID	var(*LIST).GUARD-COND.WARN(*LIST).GUARD	S	" <guard-name>	INF=*DETAIL
Reason for not automatically matching the guard condition	var(*LIST).GUARD-COND.WARN(*LIST).REASON	S	" <c-string>	INF=*DETAIL
User ID of the relevant guard	var(*LIST).GUARD-COND.WARN(*LIST).USER-ID	S	" <userid>	INF=*DETAIL
Name of the relevant guard which occurs more than once without catalog ID and user ID	var(*LIST).GUARD.CONFL(*LIST).GUARD	S	" <guard-name>	INF=*DETAIL
Number of checked pubsets on which multiple guards exist	var(*LIST).GUARD.CONFL(*LIST).NUM-OF-PUBSET	I	" <integer>	INF=*DETAIL

Catalog ID of the relevant pubset	var(*LIST).GUARD.CONFL(*LIST).PUBSET(*LIST).CAT-ID	S	"<catid>	INF=*DETAIL
User ID of the relevant guard	var(*LIST).GUARD.CONFL(*LIST).USER-ID	S	"<userid>	INF=*DETAIL
Number of guard name conflicts	var(*LIST).GUARD.NUM-OF-CONFL	I	<integer>	INF=*DETAIL/*SUMMARY
Path name of the impermissible file	var(*LIST).INADMIS.F(*LIST)	S	"<filename>	INF=*DETAIL/*SUMMARY
Number of file name conflicts (0 if check carried out by a nonprivileged user)	var(*LIST).INADMIS.NUM-OF-ERRORS	I	<integer>	INF=*DETAIL/*SUMMARY
Name of job variable that occurs more than once without catalog ID and user ID	var(*LIST).JV.CONFL(*LIST).JV	S	"<filename>	INF=*DETAIL
Number of checked pubsets on which multiple job variables exist	var(*LIST).JV.CONFL(*LIST).NUM-OF-PUBSET	I	"<integer>	INF=*DETAIL
Catalog ID of the relevant pubset	var(*LIST).JV.CONFL(*LIST).PUBSET(*LIST).CAT-ID	S	"<catid>	INF=*DETAIL
User ID of the job variable that occurs more than once	var(*LIST).JV.CONFL(*LIST).USER-ID	S	"<userid>	INF=*DETAIL
Catalog ID of the JV name that is too long	var(*LIST).JV-LEN.ERROR(*LIST).CAT-ID	S	"<catid>	INF=*DETAIL
Name of the JV that is too long without catalog ID and user ID	var(*LIST).JV-LEN.ERROR(*LIST).JV	S	"<filename>	INF=*DETAIL
Number of characters in the JV name that is too long which exceed the maximum of 54 (with catalog ID)	var(*LIST).JV-LEN.ERROR(*LIST).NUM-OF-CHAR	I	"<integer>	INF=*DETAIL
User ID of the JV name that is too long	var(*LIST).JV-LEN.ERROR(*LIST).USER-ID		"<userid>	INF=*DETAIL
Number of JV names that are too long	var(*LIST).JV-LEN.NUM-OF-ERRORS	I	<integer>	INF=*DETAIL/*SUMMARY

Number of JV name conflicts	var(*LIST).JV.NUM-OF-CONFL	I	<integer>	INF=*DETAIL/*SUMMARY

14.4 Creating new SM pubsets

The CREATE-SYSTEM-MANAGED-PUBSET statement can be used by the privileged user to transfer up to 255 SF pubsets to an SM pubset, each SF pubset becoming a volume set of the SM pubset. Here the user data and the associated management and protection information is, with a few exceptions (see "Restrictions"), retained.

When new SM pubsets are created from existing SF pubsets the following steps are performed within the system:

1. Syntactic check of the statement and preview of the result

   If the task in running in interactive mode, the structure of the SM pubset to be created is listed in screen mask 12 and again in screen mask 13 after successful import of the pubset, following the syntactic check of the CREATE-SYSTEM-MANAGED-PUBSET statement; in both masks, the user can also abort the function.

2. Implicit consistency check

   Before the disks are modified, implicitly a consistency check is run over the pubsets involved, see also section “Check function (consistency check)”. If inconsistencies are detected, the function is aborted with an error; detailed error information is output (see also the outputs of the check function in section “Outputs”).

   If guard conditions cannot be adapted (item 2, "What is checked?") messages are output during the consistency check. The consistency check is deemed terminated without error (but with warnings).
   The possible consequences of an abort following a consistency check are discussed in section “Error behavior”.

3. Exclusive reservation of the disks by SMPGEN

   Both during the consistency check and during subsequent processing, SMPGEN accesses public disks exclusively. A flag is set in the MRSCAT entries of the pubset to indicate that they are currently being processed by SMPGEN.

   The pubsets cannot be imported during the entire procedure and their MRSCAT entries can neither be deleted nor modified.

   An MRSCAT entry is also created for the new SM pubset; this entry cannot be deleted or modified during the procedure and the pubset is marked in the entry as “being created”.

4. Conversion of the pubsets

   The disk SVLs are modified in the course of conversion so that the old SF pubsets can no longer be imported as such. Their association with the SM pubset and the control volume set is permanently anchored in the SVL.

   BS2000/OSD-BC V6.0 is stored in the SVL as the lowest operating system version that supports the pubset.

5. Deletion of all MRSCAT entries of the converted pubsets

   Following successful conversion to an SM pubset, the MRSCAT entries of the pubsets that are to be converted to volume sets are deleted.
   The MRSCAT entry of the new SM pubset to be created is retained.
   The values to be set in the MRSCAT entry of the pubset, e.g. Cache-SIZE-TOLERANCE, are set to the default values.

Simulation mode

If the OPERATIONAL-MODE=*SIMULATION operand is specified, the function is not actually performed. Only a syntax check is carried out and the specified volume sets are listed. For more details, refer to the operand descriptions.

Check mode

If the OPERATIONAL-MODE=*CHECK-NAME-CONSISTENCY operand is specified, only the check function is performed. For more information please see "Check function (consistency check)".

Notes

The files that reside on the pubsets remain in their original physical locations.

The contents of the associated catalog entries remain unchanged, except for the file on a pubset that becomes the volume set with the attribute AVAILABLITY=*HIGH (where the file is not a temporary file):
In this case, the file attribute in the catalog entry is set to AVAILABILITY=*HIGH, which means that the file cannot be migrated to a volume set with a lower availability.

The catalog entries of files that have no disk assignment to the SF pubsets (e.g. migrated files, private disk or volume disks) retain their original contents, just like the job variables.

Whenever possible, BS2000 OSD/BC from V11.0 onwards only supports the EXTRA LARGE catalog format, see the “Introduction to System Administration” [5 (Related publications)]. The NORMAL and LARGE catalog formats are now obsolete. The following aspects must be taken into consideration:

1. Catalogs of a preceding version in the NORMAL or LARGE format are automatically converted into the EXTRA LARGE format as soon as the pubset is imported exclusively or as master. At the same time, the special catalogs are renamed as well. These converted catalogs can be used in a preceding version as well.
2. The SM is created with the catalog format EXTRA LARGE. However, the conversion of the catalogs does not take place during the SMPGEN run, but during the first import after the run, see previous point. It is thus recommended that you import the SM pubset immediately after the SMPGEN run.
3. When a new SM pubset is generated, only one special catalog of the types TSOSCAT.#M00, TSOSCAT.#J00 and TSOSCAT.#P00 each is created. If these catalogs are not large enough to contain all special catalog entries, it is advisable first to generate an SM pubset comprising the future control volume set and then to import this pubset. Additional special catalogs can subsequently be created with `/ADD-CATALOG-FILE` and the SM pubset can be extended by the remaining SF pubsets in a second SMPGEN run.

If GUARDS is used, the message `PRO6009  ERROR WHEN CLOSING GUARDS CATALOG` is output for each affected volume set when the volume sets are exported after successful pubset conversion; this message can be ignored.

The contents of the guards also remain unchanged. For guards that reference a program that is cataloged in the pubsets to be converted, the catalog ID in the path name is replaced by the new pubset ID (this applies only to guard entries within the new SM pubset, however; guard entries in other pubsets are not adapted; further, it is not possible to have automatically adaptable guards, see section “Restrictions”).

The user entries are transferred in the valid format for SM pubset, whereby group structures could be lost (see section “Restrictions”).
If there is a user ID on the pubset but not on the pubset that is to be used for the logon information, the user ID is marked “locked” in the SM pubset (USER-LOCKED=*YES), since no accounting information is available. This is not relevant for using the SM pubset as a data pubset.
If there is a user ID on more than one of the pubsets that are to be combined to form an SM pubset, this ID is treated as though it were the same user.

SMPGEN requires a correct SYSFILE environment. If the function is to be executed in a new batch task that is to be created, the default pubset of the runtime user ID (or in ACS defined Spool-Pubset) must be imported. If SYSOUT or SYSLST outputs fails repeatedly, the SMPGEN function is terminated with an error.

The (physical) format of the control volume set is entered as DEFAULT FILE FORMAT.

If several pubsets with different formats are combined, the DEFAULT FILE FORMAT created by SMPGEN may have to be modified.

Example

The format of the control volume set is K; all other volume sets are keyless (NK); in this case, the files are created on the control volume set per default and this becomes too full.

WARNING!

- CREATE-SYSTEM-MANAGED-PUBSET OPERATIONAL-MODE= *OPERATION should never be called without a previous FDDRL backup of the pubset to be converted, since an abort due to an error would destroy all the pubsets and render them inaccessible.
- A logical backup using HSMS is also recommended. Here, a new backup archive can be created which can be used for the creation of the HSMS environment for the SM pubset. For details of how to adapt the HSMS environment, refer to the description of SMS migration in the “System-Managed Storage” manual [8 (Related publications)].
- SPOOL jobs for files that are cataloged in the pubsets to be converted should no longer be present, since they can no longer be carried out after renaming the catalog ID.
- When the operating system is operated without SECOS at the time of pubset conversion, SECOS-specific user information (e.g. group information, privileges) is completely lost even if KEEP-USER-ATTRIBUTES=*ALL is specified.

14.4.1 Determining quotas and other attributes

The number of file entries, job variables and allocated pages is added, as are the catalog entry (CE) and public space limits. The quotas (limits) are set as follows:

- FILE-NUMBER-LIMIT = Sum(FILE-NUMBER-LIMIT), totalled over all SF pubsets
- JV-NUMBER-LIMIT = Sum(JV-NUMBER-LIMIT), totalled over all SF pubsets
- TOTAL-SPACE-LIMIT(PERM) = MAXIMUM
- S0-LEVEL-SPACE-LIMIT(PERM) = Sum(PUBLIC-SPACE-LIMIT), totalled over all SF pubsets
- HIGH-PERF-SPACE-LIMIT(PERM) = MAXIMUM
- VERY-HIGH-PERF-SPACE-LIMIT(PERM) = MAXIMUM
- HIGH-AVAIL-SPACE-LIMIT(PERM) = Sum(PUBLIC-SPACE-LIMIT), totalled over all SF pubsets that become HIGH-AVAIL volume set
- TOTAL-SPACE-LIMIT(TEMP) = Sum(TEMP-SPACE-LIMIT), totalled over all SF pubsets
- HIGH-PERF-SPACE-LIMIT(TEMP) = MAXIMUM
- VERY-HIGH-PERF-SPACE-LIMIT(TEMP) = MAXIMUM
- TOTAL-SPACE-LIMIT(WORK) = MAXIMUM
- HIGH-PERF-SPACE-LIMIT(WORK) = MAXIMUM
- VERY-HIGH-PERF-SPACE-LIMIT(WORK) = MAXIMUM

MAXIMUM here means that no separate limit exists for the particular subquota, but that it is matched dynamically to the more comprehensive top quota. Thus the space available for VERY-HIGH-PERF files for example, can be more than the entire space available for HIGH-PERF files. For the top quota, MAXIMUM means the maximum assignable value (i.e. 2147483647).

The values for PUBLIC-SPACE-EXCESS, DMS-TUNING-RESOURCES and PHYSICAL-ALLOCATION are set to ensure that the least restrictive condition is taken over by all the SF pubsets concerned. The ascending order of restriction is as follows:

- PUBLIC-SPACE-EXCESS: ALLOWED, TEMPORARILY-ALLOWED, NO
- DMS-TUNING-RESOURCES: EXCLUSIVE-USE, CONCURRENT-USE, NONE
- PHYSICAL-ALLOCATION: ALLOWED, NOT-ALLOWED

When determining the SPACE-USED values, all files of a user are considered in the assignment counters according to their position on the background levels, their attributes temporary/permanent, their performance and availability attributes. The files that reside on a HIGH-AVAILABILITY volume set are included in the HIGH-AVAIL quota because they receive the file attribute HIGH-AVAIL from SMPGEN.
If group structures are adopted from one of the pubsets, the following applies:

- PUBLIC-SPACE-LIMIT: MAXIMUM
- TEMP-SPACE-LIMIT: MAXIMUM
- WORK-SPACE-LIMIT: MAXIMUM
- FILE-NUMBER-LIMIT: MAXIMUM
- JV-NUMBER-LIMIT: MAXIMUM

The values PUBLIC-SPACE-EXCESS and DMS-TUNING-RESOURCES are adopted from the pubset marked as ATTRIBUTE-ORIGIN.

The (physical) format of the control volume set is entered as DEFAULT FILE FORMAT.

The allocation defaults (for primary and secondary allocation) are taken from the SF pubset that becomes the control volume set.
The saturation thresholds remain the same in each volume set as those selected for the converted SF pubset.

The following files required for administration of the SM pubset are created on the control volume set:

- $TSOS.SYS.PUBSET.CONFIG: Configuration description
- $TSOS.TSOSCAT.#M00: Catalog of the migrated and nospace files
- $TSOS.TSOSCAT.#P00: Catalog of the private disks and volume disks
- $TSOS.TSOSCAT.#J00: Catalog of the job variables
- $TSOS.TSOSCAT.$PFI: Index of all catalog entries
- $TSOS.TSOSCAT.$NLO: List of all file names
- $TSOS.TSOSCAT.$NLC: Copy of the file name list

Catalog entries not held in special catalogs in the SM pubsets (i.e. all those that take up space in the pubset, except for files) are migrated to this new special catalog.

The $TSOS.TSOSCAT files are renamed to ones that are volume-set-specific

The files $TSOS.SYSCAT.GUARDS and $TSOS.SYSSRPM are deleted on all pubsets that do not become a control volume set. The guard and user entries are transferred to the corresponding file on the control volume set.

If EAM is to be operated on the SM pubset, a new EAM file must be set up. The following files are deleted:

- $TSOS.SYSEAM
- $TSOS.SYSPBN
- $TSOS.TSOSJOIN
- $TSOS.SYSSRPM.BACKUP

The cache configuration is taken from the MRSCAT entries of the pubsets to be converted in the pubset configuration file of the SM pubset. Exception: the pubset-related cache size tolerance, which must be reset for the new pubset, if desired.

The shareability value anchored in the SVL of the pubset converted to the control volume set defines the shareability of the newly created SM pubset.

The SYSIDs in the SVLs are generally retained; only in the case of the control volume set is it replaced in the event of an SM pubset ID that is 1 byte long.

14.4.2 Restrictions

The user data and the associated administration and protection information are retained except for the following restrictions:

- If one or more of the SF pubsets transferred to the volume set was usable as the home pubset, this option no longer possible; an SM pubset can no longer be used as the home pubset.
- Only the contents of the guard entries in the pubsets specified for conversion are adapted and remain valid (if they can be adapted; if not, the warnings mentioned on "Outputs" and following are output; the relevant guard entries must be adapted by the guard owner.)

  Guard entries in other pubsets can be rendered invalid by conversion. This happens when the access condition contains the name of a program that was cataloged in the SF pubset. Since the path name of the program has changed during the conversion, the program that has the access right is no longer found. The owner of the guard must therefore modify the guard.
- The SYSEAM files are lost.
- The privileges and authorizations relevant only to the home pubset are either lost or taken over by exactly one of the SF pubsets, depending on the setting of the KEEP-USER-ATTRIBUTES parameter.
- The logon password are either lost or taken over by exactly one of the SF pubsets, depending on the setting of the KEEP-USER-ATTRIBUTES parameter.
  (In the latter case, the logon passwords are lost for users who do not have a user entry on the specified SF pubset. However, as the system does not check passwords which are stored on SM pubsets, this is not important.)
- The group structures are either lost or taken over by exactly one of the SF pubsets, depending on the setting of the KEEP-USER-ATTRIBUTES parameter.

  The user IDs for which no group structure is adopted are assigned to the group *UNIVERSAL.
  (This is in the one case all user IDs, and in the other the user IDs that are not entered on the specified pubset or have not been assigned to a group there).
- The SYSSRPM.BACKUP file is lost. This means that the pubset conversion and switchover to another user catalog cannot be carried out simultaneously.
- SPOOL jobs are lost if they refer to files that have the catalog ID of a converted SF pubset, since the catalog ID is not adapted in the SPOOL job queue. Any print jobs from an earlier session that it still contains must therefore be restarted and the associated temporary files must then be deleted explicitly by systems support.
- The HSMS-EXCEPT-FILE files loses its effect. To restore the protection against unwanted file displacement, systems support can create a pubset-specific EXCEPT-FILE on the SM pubset and assign it to the pubset when setting up the HSMS environment, or assign the attribute MIGRATE=*FORBIDDEN to the files affected.
- Pubset configuration files that do not originate from an aborted reconfiguration job or from an aborted cache mode change are deleted implicitly, and a warning is issued. It is recommended not to carry out pubset conversion if the reconfiguration is incomplete, or if the cache mode has been changed; instead, clean up the pubset beforehand using `/RESUME-PUBSET-RECONFIGURATION`.

14.4.3 Outputs

Error messages and logging messages, as well as the concluding success messages are routed to the same output destination as the information on inconsistencies, i.e. to the destination output(s) specified by TEXT-OUTPUT.
In interactive mode, the success message and the logging messages are also output to SYSOUT if TEXT-OUTPUT=*SYSLST.

If the information on inconsistencies is not output in text form (TEXT-OUTPUT=*NONE), the error messages, the message relating to the start of the function, and the concluding success message are output to SYSOUT.

1. In interactive mode: Table of the volume sets with the specified logical attributes. This table is output to SYSOUT regardless of the TEXT-OUTPUT operand.

   Refer to mask 12 in section "Outputs to screen masks" for format details.
   This mask provides the user with the option of canceling the function by making an appropriate entry in the footer.
2. Table of the volume sets with specification of the disk format and specified logical attributes.

   Refer to mask 13 in section "Outputs to screen masks" for format details.
   This mask provides the user with the option of canceling the function by making an appropriate entry in the footer.
3. Logging messages:
   - Message indicating that the function or task executing the function has been started
   - Progress log: message that the GUARDS/FACS/SRPM/CMS conversion has been started
   - Result message of the consistency check
     The result message is always output to SYSOUT in interactive mode, independent of the TEXT-OUTPUT operand.
   - In special cases, or in the event of termination on error, further messages may be issued first.
4. The following takes place in the event of an error:
   - The illegal files, name conflicts, over-long names, guard condition errors are listed as for CHECK-NAME-CONSISTENCY (see section “Outputs”).

     Refer to masks 2-11 in section "Outputs to screen masks" for format details.
   - If there are files in one of the pubsets that have been converted to S1 and S1-MIGRATED-FILES=*ALLOWED is not specified, an error message is issued. (One message for each pubset affected).

Outputs in simulation mode

1. Result message after syntax check
2. Table of the volume sets with the specified logical attributes.

   Refer to mask 12 in section "Outputs to screen masks" for format details.

14.5 Extending an existing SM pubset

The MODIFY-SYSTEM-MANAGED-PUBSET statement is used by the privileged user to extend an existing SM pubset by the addition of SF pubsets, each SF pubset becoming a volume set of the SM pubset. When the statement is executed a check is made to see that a maximum of 255 volume sets are contained in the extended SM pubset.

The user data and the associated management and protection information of the SF pubsets to be added and of the existing SM pubset are, with a few exceptions (see section “Restrictions”), retained.

When new SM pubsets are extended the following steps are performed within the system:

1. Syntactic check of the statement and preview of the result

   If the task in running in interactive mode, the structure of the SM pubset to be extended is listed in screen mask 12 and again in screen mask 13 after successful import of the pubset, following the syntactic check of the MODIFY-SYSTEM-MANAGED-PUBSET statement; in both masks, the user can also abort the function.

   With MODIFY-SYSTEM-MANAGED-PUBSET only the newly added volume sets are listed in screen masks 12 and 13. The existing volume sets of the SM pubset are not displayed in the list.

2. Implicit consistency check

   Before the disks are modified, implicitly a consistency check is run over the pubsets involved, see also section “Check function (consistency check)”. If inconsistencies are detected, the function is aborted with an error; detailed error information is output (see also the outputs of the check function in section “Outputs”).

   If guard conditions cannot be adapted (item 2, "What is checked?") messages are output during the consistency check. The consistency check is deemed terminated without error (but with warnings).
   The possible consequences of an abort following a consistency check are discussed in section “Error behavior” ff.

3. Exclusive reservation of the disks by SMPGEN

   Both during the consistency check and during subsequent processing, SMPGEN accesses public disks exclusively. A flag is set in the MRSCAT entries of the pubset to indicate that they are currently being processed by SMPGEN.

   The pubsets cannot be imported during the entire procedure and their MRSCAT entries can neither be deleted nor modified. The MRSCAT entry of the SM pubset to be extended is marked as “being generated”.

4. Conversion of the pubsets

   The disk SVLs of the SF pubsets are modified in the course of conversion so that the old SF pubsets can no longer be imported as such. Their association with the SM pubset and the control volume set is permanently anchored in the SVL.

   BS2000/OSD-BC V6.0 is stored in the SVL as the lowest operating system version that supports the pubset.

5. Deletion of all MRSCAT entries of the converted pubsets

   Following successful conversion to an SM pubset, the MRSCAT entries of the SF pubsets that are to be converted to volume sets are deleted.

6. Reworking

   After the SMPGEN statement for extending an SM pubset has been executed this does not mean that extension of the SM pubset has been completed. Nor is a renewed extension of the SM pubset possible immediately after the SMPGEN statement for extending an SM pubset has been executed. Extension of an SM pubset can be completed only after the SM pubset has been successfully imported and the next extension of the SM pubset with SMPGEN is permitted after this.

   When the next import - which must be initiated by the user - takes place the CMS catalogs of the SM pubset are cleared of superfluous job variable entries.

Simulation mode

If the OPERATIONAL-MODE=*SIMULATION operand is specified, the function is not actually performed. Only a syntax check is carried out and the specified volume sets are listed. For more details, refer to the operand descriptions.

Check mode

If the OPERATIONAL-MODE=*CHECK-NAME-CONSISTENCY operand is specified, only the check function is performed. For more information please see "Check function (consistency check)".

Notes

If GUARDS is used, the message `PRO6009  ERROR WHEN CLOSING GUARDS CATALOG` is output for each affected volume set when the volume sets are exported after successful pubset conversion; this message can be ignored.

Whenever possible, BS2000 from OSD/BC V11.0 onwards only supports the EXTRA LARGE catalog format, see the “Introduction to System Administration” [5 (Related publications)]. The NORMAL and LARGE catalog formats are now obsolete. When SM pubsets are extended in conjunction with SF pubsets, the following special aspects must be taken into consideration:

1. Catalogs of a preceding version in the NORMAL or LARGE format are automatically converted into the EXTRA LARGE format as soon as the pubset is imported exclusively or as master. At the same time, the special catalogs are renamed as well. These converted catalogs can be used in a preceding version as well.
2. While an SM pubset is being extended, only the special catalogs are available that already exist. However, if necessary, these will be extended to their maximum size during the SMPGEN run (32008 4K blocks).

SMPGEN requires a correct SYSFILE environment. If the function is to be executed in a new batch task that is to be created, the default pubset of the runtime user ID (or the SPOOL pubset defined in ACS) must be imported. If SYSOUT or SYSLST outputs fails repeatedly, the SMPGEN function is terminated with an error.

If several pubsets with different formats are combined, the DEFAULT FILE FORMAT created by SMPGEN may have to be modified.

Example

The format of the control volume set is K; all other volume sets are keyless (NK); in this case, the files are created on the control volume set per default and this becomes too full.

WARNING!

- MODIFY-SYSTEM-MANAGED-PUBSET OPERATIONAL-MODE= *OPERATION should never be called without a previous FDDRL backup of the SF pubsets to be converted, since an abort due to an error would destroy all the SF pubsets and render them inaccessible.
- A logical backup using HSMS is also recommended. Here, a new backup archive can be created which can be used for the creation of the HSMS environment for the extended SM pubset. For details of how to adapt the HSMS environment, refer to the description of SMS migration in the “System-Managed Storage” manual [8 (Related publications)].
- SPOOL jobs for files that are cataloged in the SF pubsets to be converted should no longer be present, since they can no longer be carried out after renaming the catid.

14.5.1 Restrictions

The user data and the associated administration and protection information are retained except for the following restrictions:

- The SYSEAM files on the SF pubsets are lost.
- The privileges and authorizations on the SF pubsets which are only relevant to the home pubset are lost.
- The logon passwords on the SF pubsets are totally lost.
- The group structures on the SF pubsets to be added are totally lost.
- The SYSSRPM.BACKUP file on the SF pubsets to be added is lost. This means that the pubset conversion and switchover to another user catalog cannot be carried out simultaneously.
- Pubset configuration files on the SF pubsets to be added which originate from an aborted reconfiguration job or an aborted cache mode change are deleted implicitly, and a warning is issued.
  It is recommended that you should not carry out pubset conversion if the reconfiguration is incomplete, or if the cache mode has been changed; instead, clean up the pubset beforehand using `/RESUME-PUBSET-RECONFIGURATION`.

The statements made regarding the conversion of SF pubsets to new SM pubsets also apply for the location of the files on the SF pubsets, the catalog entries of files and job variables on the SF pubsets and the guards on the SF pubsets.

- The user entries on the SF pubsets are converted to the format valid for SM pubsets, groups structures being lost in the process.

  If a user ID exists on one of the SF pubsets but not on the pubset from which the logon information is to be taken over, the user ID is marked as “locked” (USER-LOCKED=*YES) in the SM pubset since no accounting information is available.

  This is not relevant for using the SM pubset as a data pubset. If a user ID exists on several of the SF pubsets which are to be added to the SM pubset or on the SM pubset, it is handled as though it were an identical user. The numbers of file entries, JVs and reserved pages in the SF pubsets to be included and the SM pubset are totaled, as are the CE and public space limits.

The statements made regarding the conversion of SF pubsets to new SM pubsets also apply for updating or creating user-specific quotas, SPACE-USED values and settings, with a sum total being obtained for all the SF pubsets to be added and the SM pubset.

- The values for PUBLIC-SPACE-EXCESS, DMS-TUNING-RESOURCES and PHYSICAL-ALLOCATION are set to ensure that the least restrictive condition is taken over by all the SF pubsets concerned. The ascending order of restriction is as follows:
  - PUBLIC-SPACE-EXCESS: ALLOWED, TEMPORARILY-ALLOWED, NO
  - DMS-TUNING-RESOURCES: EXCLUSIVE-USE, CONCURRENT-USE, NONE
  - PHYSICAL-ALLOCATION: ALLOWED, NOT-ALLOWED
- The DEFAULT FILE FORMAT and the allocation defaults (standard for primary and secondary allocation) of the SM pubset remain unchanged. The saturation thresholds remain the same in each volume set as those selected for the converted SF pubset.

The following files required for administration of the SM pubset are updated on the control volume set:

- $TSOS.SYS.PUBSET.CONFIG: Configuration description
- $TSOS.TSOSCAT.#Mnn (nn=00..99): Catalog(s) of the migrated and nospace files
- $TSOS.TSOSCAT.#Pnn (nn=00..99): Catalog(s) of the private disks and volume disks
- $TSOS.TSOSCAT.#Jnn (nn=00..99): Catalog(s) of the job variables
- $TSOS.TSOSCAT.$PFI: Index of all catalog entries
- $TSOS.TSOSCAT.$NLO: List of all file names
- $TSOS.TSOSCAT.$NLC: Copy of the file name list

Catalog entries from the SF pubsets that are to be added which are held in special catalogs in the SM pubsets (i.e. all those that take up space in the pubset, except for files) are migrated to these existing special catalogs.

The $TSOS.TSOSCAT files are renamed to ones that are volume-set-specific

The following files are deleted on the SF pubsets that are to be added:

- $TSOS.SYSCAT.GUARDS
- $TSOS.SYSEAM
- $TSOS.SYSPBN
- $TSOS.SYSSRPM
- $TSOS.SYSSRPM.BACKUP
- $TSOS.TSOSJOIN

The guard and user entries from the $TSOS.SYSCAT.GUARDS and $TSOS.SYSSRPM files are transferred to the relevant file on the control volume set.

The cache configuration is transferred from the MRSCAT entries of the SF pubsets that are to be converted to the pubset configuration file of the SM pubset. Exception: the systemrelated or pubset-related cache size tolerance, which must be reset for the new pubset, if required.

The shareability value anchored in the SVL of the control volume set remains unchanged.

The SYSIDs in the SVLs are generally retained.

14.5.2 Outputs

Error messages and logging messages, as well as the concluding success messages are routed to the same output destination as the information on inconsistencies, i.e. to the destination output(s) specified by TEXT-OUTPUT.
In interactive mode, the success message and the logging messages are also output to SYSOUT if TEXT-OUTPUT=*SYSLST.

If the information on inconsistencies is not output in text form (TEXT-OUTPUT=*NONE), the error messages, the message relating to the start of the function, and the concluding success message are output to SYSOUT.

1. In interactive mode: Table of the SF pubsets to be converted to volume sets from the ADD-VOLUME-SET operand with the specified logical attributes.
   This table is output to SYSOUT regardless of the TEXT-OUTPUT operand.

   Refer to mask 12 in section "Outputs to screen masks" for format details.
   This mask provides the user with the option of canceling the function by making an appropriate entry in the footer.
2. Table of the SF pubsets to be converted to volume sets from the ADD-VOLUME-SET operand with specification of the disk format and specified logical attributes.

   Refer to mask 13 in section "Outputs to screen masks" for format details.
   This mask provides the user with the option of canceling the function by making an appropriate entry in the footer.
3. Logging messages:
   - Message indicating that the function or task executing the function has been started
   - Progress log: message that the GUARDS/FACS/SRPM/CMS conversion has been started
   - Result message of the consistency check: The result message is always output to SYSOUT in interactive mode, independent of the TEXT-OUTPUT operand.
   - In special cases, or in the event of termination on error, further messages may be issued first.
4. The following takes place in the event of an error:
   - The illegal files, name conflicts, over-long names, guard condition errors are listed as for CHECK-NAME-CONSISTENCY (see section “Outputs”).

     Refer to masks 2-11 in section "Outputs to screen masks" for format details.
   - If there are files in one of the pubsets that have been converted to S1 and S1-MIGRATED-FILES=*ALLOWED is not specified, an error message is issued. (One message for each pubset affected).

Outputs in simulation mode

1. Result message after syntax check
2. Table of the SF pubsets to be converted to volume sets from the ADD-VOLUME-SET operand with the specified logical attributes.
   Refer to mask 12 in section "Outputs to screen masks" for format details.

14.6 Starting and stopping SMPGEN

The program is started with `/START-SMPGEN`.

START-SMPGEN	Alias: **SMPGEN**
VERSION = ***STD** / <product-version> **,MONJV** = ***NONE** / <filename 1..54 without-gen-vers> **,CPU-LIM**IT = ***JOB-RE**ST / <integer 1..32767 *seconds*>	

The END statement is used to terminate SMPGEN.

Format

END

The statement has no operands.

14.7 Statements

- Overview of SMPGEN statements
- Description of the statements
  - CREATE-SYSTEM-MANAGED-PUBSET - Convert SF pubsets to SM pubset
  - MODIFY-SYSTEM-MANAGED-PUBSET - Extending an existing SM pubset

14.7.1 Overview of SMPGEN statements

Statement	Function
CREATE-SYSTEM-MANAGED-PUBSET	Convert one or more SF pubsets to an SM pubset. • In simulation mode: check the statement and list the pubset to be generated • In check mode: check whether the specified SF pubsets can be converted and can be combined to form an SM pubset
MODIFY-SYSTEM-MANAGED-PUBSET	Convert one or more SF pubsets and combine them with an existing SM pubset. • In simulation mode: check the statement and list the pubset to be generated • In check mode: check whether the specified SF pubsets can be converted and can be combined with an existing SM pubset
CHECK-NAME-CONSISTENCY [1)]	Check whether the specified SF pubsets can be converted to an SM pubset

[1)] The statement CHECK-NAME-CONSISTENCY is only supported for reasons of compatibility and is no longer visible in a guided dialog. The extended functions of the CREATE- and MODIFY-SYSTEM-MANAGED-PUBSET statements are available.

In addition, SMPGEN supports execution of the SDF standard statements (see the “SDF Dialog Interface” manual [20 (Related publications)]).

14.7.2 Description of the statements

- CREATE-SYSTEM-MANAGED-PUBSET - Convert SF pubsets to SM pubset
- MODIFY-SYSTEM-MANAGED-PUBSET - Extending an existing SM pubset

14.7.2.1 CREATE-SYSTEM-MANAGED-PUBSET - Convert SF pubsets to SM pubset

- For privileged users

  The CREATE-SYSTEM-MANAGED-PUBSET statement either creates a new SM pubset or checks whether specified SF pubsets can be included in a new SM pubset without conflict. When a new SM pubset is created a maximum of 255 SF pubsets are converted to a new SM pubset, each SF pubset becoming a volume set of the SM pubset.The check function performs a consistency check which provides information on system-wide name conflicts in the pubsets concerned.

- For nonprivileged users

  The CREATE-SYSTEM-MANAGED-PUBSET statement only provides the check function for nonprivileged users. Here a consistency check decides whether the specified SF pubsets can be included in a new SM pubset. The checks relate only to the caller's user ID. Callers can thus obtain information on which name conflicts they have caused and how they can resolve these.

Outputs

The outputs of this statement are described in the section "Outputs (Creating new SM pubsets)" or in the section "Outputs (Check function)"; the mask descriptions start on "Outputs to screen masks".

Operation

How this statement functions, its restrictions and the quota management are described in the section "Creating new SM pubsets".

> **WARNING!**
>
> - CREATE-SYSTEM-MANAGED-PUBSET OPERATIONAL-MODE= *OPERATION should never be called without a previous FDDRL backup of the pubset to be converted, since an abort due to an error would destroy all the pubsets and render them inaccessible.
> - A logical backup using HSMS is also recommended. Here, a new backup archive can be created which can be used for the creation of the HSMS environment for the SM pubset. For details of how to adapt the HSMS environment, refer to the description of SMS migration in the "System-Managed Storage" manual [8 (Related publications)].
> - SPOOL jobs for files that are cataloged in the pubsets to be converted should no longer be present, since they can no longer be carried out after renaming the catalog ID.

Simulation and check mode

If the operand OPERATIONAL-MODE=*SIMULATION is specified, the function described is not really executed. Only a syntax check is carried out and the specified volume sets are listed.

If the OPERATIONAL-MODE=*CHECK-NAME-CONSISTENCY operand is specified, only the check function is performed.

For more details, refer to the operand descriptions.

Format for privileged users

```
CREATE-SYSTEM-MANAGED-PUBSET
```

```
PUBSET=<catid>
,CONTROL-VOLUME-SET=<catid>(...)
    <catid>(..)
        |   AVAILABILITY=*STD/*HIGH
        |   ,PERFORMANCE-ATTR=*STD/*PARAMETERS(...)
        |       *PARAMETERS(...)
        |           |   PERFORMANCE=list-poss(3):*STD/*HIGH/ *VERY-HIGH
        |           |   ,WRITE-CONSISTENCY=*BY-CLOSE/*IMMEDIATE
,ADD-VOLUME-SET=*NONE/list-poss(254):<catid>(...)
    <catid>(...)
        |   AVAILABILITY=*STD/*HIGH
        |   ,USAGE = *STD / *HSMS-CONTROLLED
        |   ,PERFORMANCE-ATTR=*STD/*PARAMETERS(...)
        |       *PARAMETERS(...)
        |           |   PERFORMANCE=list-poss(3):*STD/*HIGH/ *VERY-HIGH
        |           |   ,WRITE-CONSISTENCY=*BY-CLOSE/*IMMEDIATE
,KEEP-USER-ATTRIBUTES=*ALL(...)/*BASIC
    *ALL(...)
        |   ATTRIBUTE-ORIGIN=*CONTROL-VOLUME-SET/<catid>
,S1-MIGRATED-FILES=*NOT-ALLOWED/*ALLOWED
,MAX-ERRORS=32767/<integer0..32767>
,MONJV=*NONE/ <filename 1..54 without-gen-vers>
,ERROR-INFORMATION=*DETAIL/*SUMMARY
,EXECUTION-MODE=*SYNCHRONOUS(...)/*ASYNCHRONOUS(...)
    *SYNCHRONOUS(...)
        |   TEXT-OUTPUT=*STD/*SYSOUT/*SYSLST/*NONE
        |   ,STRUCTURE-OUTPUT=*SYSINF/*NONE/<composed-name1..100>
    *ASYNCHRONOUS(...)
        |   TEXT-OUTPUT=*SYSLST/<filename 1..54 without-gen-vers>
```

```
,OPERATIONAL-MODE=*OPERATION / *SIMULATION / *CHECK-NAME-CONSISTENCY(...)

    *CHECK-NAME-CONSISTENCY(...)

        |    PUBSET-STATE=*IMPORTED/*NOT-IMPORTED

        |    ,USER-ID=*OWN/*ALL
```

Operands

PUBSET = <catid>
ID of the SM pubset to be created or which is to be checked to see whether it can be created.

The ID must differ from the IDs of the SF pubsets to be converted.

If not just a check is to take place (OPERATIONAL-MODE= *CHECK-NAME-CONSISTENCY operand) but also a conversion, no identical MRSCAT entry may already exist.
The specified ID is included in the check of the file name length and of whether guard conditions can be modified automatically.

CONTROL-VOLUME-SET = <catid>(...)
ID of the SF pubset that is to become the control volume set of the new SM pubset.

It must differ from the ID of the SM pubset to be created.
As for all SF pubsets to be converted, there must also be an MRSCAT entry with the correct device type for this pubset.

If not just a check is to take place (OPERATIONAL-MODE= *CHECK-NAME-CONSISTENCY operand) but also a conversion, the pubset may not be imported in either the local or any other system.
The control volume set must satisfy the requirements described in the section “Requirements for operation of SMPGEN”.

If just the test function is performed (OPERATIONAL-MODE=*CHECK-NAME-CONSISTENCY), the specification for the subsequent volume set attributes is ignored.

AVAILABILITY =
Defines the degree of availability of files with respect to the failsafe capacity of the volume set.
The availability is a logical attribute of the volume set that is used as a selection criterion when choosing storage space for archiving files. Changing the volume set attribute is a job that must be commissioned explicitly by systems support.

AVAILABILITY = *STD
Default value: the volume set does not offer enhanced availability.

AVAILABILITY = *HIGHThe volume set offers high availability.
It is the responsibility of systems support to check that failsafe performance is actually guaranteed physically and, for example, to activate DRV or DUALCOPY mode before starting conversion.

PERFORMANCE-ATTR =
Defines the performance attributes of the volume set for file access. These attributes are logical attributes which are used as a selection criterion when choosing the storage space. The available hardware is not verified (e.g. cache).

PERFORMANCE-ATTR = *STD
Default value: no enhanced performance attributes for file access are to be defined for the volume set.

PERFORMANCE-ATTR = *PARAMETERS(...)
The performance attributes that apply to the volume set are defined via the following substructure:

PERFORMANCE =
Describes the I/O performance profile of the volume set. It is possible to specify a list of up to three values.

PERFORMANCE = *STD
Default value: the volume set provides no enhanced I/O performance.

PERFORMANCE = *HIGH
The volume set provides enhanced I/O performance.

PERFORMANCE = *VERY-HIGH
The volume set provides very high I/O performance.

WRITE-CONSISTENCY =
Defines whether the hardware used to increase the I/O performance guarantees data consistency even in the event of a system crash.

WRITE-CONSISTENCY = *BY-CLOSE
Default value: write I/Os with enhanced performance are possible only in trade-off against immediate crash-proof data storage; until the file is closed the data is not physically stored on a medium which guarantees data retention even if the system crashes.

WRITE-CONSISTENCY = *IMMEDIATE
Also for write I/Os with enhanced performance, the data is immediately stored on a medium which guarantees data retention even in the event of a system crash.

ADD-VOLUME-SET =
Specifies whether further SF pubsets are to be added to the SM pubset, in addition to the pubset specified as the control volume set.

ADD-VOLUME-SET = *NONE
Default value: no further SF pubsets are to be added.

ADD-VOLUME-SET = <catid>(...)
IDs of further SF pubsets to be added to the SM pubset as a volume set. The pubset ID becomes the volume set ID.

All volume set IDs must be different from the ID of the control volume set and the selected SM pubset ID.
If not just a check is to take place (OPERATIONAL-MODE= *CHECK-NAME-CONSISTENCY operand) but also a conversion, the SF pubsets may not be imported in either the local or any other system; however, MRSCAT entries with the correct device type must already exist for this purpose.
The volume sets must satisfy the requirements described in the section “Requirements foroperation of SMPGEN”.

The volume set attributes, except USAGE, that can be specified as the substructure are described under the CONTROL-VOLUME-SET operand.
If just the test function is performed (OPERATIONAL-MODE=*CHECK-NAME-CONSISTENCY), the specification for the subsequent volume set attributes is ignored.

USAGE =
Specifies the usage of the volume set.

USAGE = *STD
The volume set is to be used as the storage location for the default files. Work files (WORK file attribute) may not be stored there.

USAGE = *HSMS-CONTROLLED
The volume set is used by the HSMS subsystem to implement storage level S1 and the backup volume set of an SM pubset. Users may not store any files there.

KEEP-USER-ATTRIBUTES =
Specifies the procedure for the login IDs and the user group structure, as well as the user information relevant only to the home pubset (privileges, rights, operator roles).

If just the test function is performed (OPERATIONAL-MODE=*CHECK-NAME-CONSISTENCY), the specification for this operand is ignored.

KEEP-USER-ATTRIBUTES = *ALL(...)
Default value: the information is to be taken from one of the SF pubsets to be combined. The information on all other original pubsets is ignored.

ATTRIBUTE-ORIGIN = *CONTROL-VOLUME-SET
The information is to be taken from the pubset that is to become the control volume set.

ATTRIBUTE-ORIGIN = <catid>
Specifies the ID of the pubset from which the information is to be taken. It must be one of the pubsets to be converted.

KEEP-USER-ATTRIBUTES = *BASIC
Only the important user information relevant to the data pubset is to be taken. The information is lost during pubset conversion (even if only one pubset is converted).

S1-MIGRATED-FILES =
Specifies whether if one of the pubsets to be checked contains cataloged files which are currently migrated to S1 level this is to be defined as a conflict.

S1-MIGRATED-FILES = *NOT-ALLOWED
Default value: the case described above is deemed to be a conflict.

S1-MIGRATED-FILES = *ALLOWED
The case described above is not deemed to be a conflict.
The S1-migrated files are no longer accessible following conversion, but must be restored from a backup file. Their catalog entries are retained so that they can be selectively moved to the S1 level of the SM pubset after creation of the HSMS environment with REPAIR-CATALOG-BY-RESTORE.

MAX-ERRORS = 32767 / <integer 0..32767>
The consistency check is aborted when the specified number of errors have occurred. The conflicts encountered before the function is aborted are output, provided ERROR-INFORMATION = *DETAIL was specified. 32767 is the default value.

MONJV =
Specifies a job variable used for monitoring.

MONJV = *NONE
Default value: no monitoring job variable is specified.

MONJV = <filename 1..54 without-gen-vers>
Specifies the name of the monitoring job variable. It must already be cataloged in an accessible pubset and must not be password-protected.

The job variables' status display is set as follows during conversion (OPERATIONAL-MODE=*OPERATION):

$I at the start of processing

$R on successful completion of processing

$A on termination with error

The job variables' status display is set as follows during the consistency check (OPERATIONAL-MODE=*CHECK-NAME- CONSISTENCY):

$I at the start of the consistency check

$R at the end of the consistency check if consistency exists

$E at the end of the consistency check if consistency does not exist

$A in the event of an incomplete consistency check

There is no point in specifying this operand unless you are using the software product JV.

> If the job variable cannot be supplied with values (e.g. because a password is set), the consistency check is performed anyway.
> If the job variable is to receive defined contents when execution of the function is started, the user must initialize this variable him/herself. If the function is aborted right at the start of execution, e.g. because a semantic error is detected, SMPGEN does not provide a value for the job variable.
>
> If the job variable is contained on one of the pubsets that are to be combined and OPERATIONAL-MODE=*CHECK-NAME-CONSISTENCY(PUBSET-STATE= *IMPORTED) is not specified, the SMPGEN job is rejected.

ERROR-INFORMATION =
Controls the scope of the error information output in text and/or variable form.

ERROR-INFORMATION = *DETAIL
Default value: detailed error information is to be output. Not only is the number of name conflicts output, but also all names that occur more than once or names that are too long are listed.
In interactive mode, screen masks are output to the terminal that offer the option of suppressing follow-up screens or detail screens, or to switch to SYSLST (see section"Outputs to screen masks").
In batch mode and to SYSLST, the output is made in the same format, but here there is no possibility of intervening.

ERROR-INFORMATION = *SUMMARY

No detailed error information is to be output, i.e. no lists of name conflicts but only the number of conflicts in the various name classes.

EXECUTION-MODE =
Specifies whether the function is to be executed in the current task or in a new batch task to be created.

EXECUTION-MODE = *SYNCHRONOUS(...)
Default value: the function is to be executed in the current task.

TEXT-OUTPUT =
Controls the output destination for data output in text format, i.e. lists, logging and error messages.

TEXT-OUTPUT = *STD
Default value: the output is to be carried out according to default setting. In interactive mode, output is to SYSOUT and to SYSLST. In batch mode, output is to SYSLST.

TEXT-OUTPUT = *SYSOUT
Output is directed to SYSOUT. This specification may only be made in interactive mode. SYSOUT may not have been reassigned to a file. If SYSOUT output is desired, screen masks are output and these must be confirmed by pressing the send key. If numerous errors occur, it is possible to apply or suppress a detailed error log for SYSLST later.

TEXT-OUTPUT = *SYSLST
Output is directed to SYSLST.

TEXT-OUTPUT = *NONE
No outputs in the form of lists or messages are desired. It makes sense to specify this value when the STRUCTURE-OUTPUT operand is used instead.
Error messages and messages reporting the start and success of the function cannot be suppressed; they are output to SYSOUT even if TEXT-OUTPUT=*NONE is specified.

STRUCTURE-OUTPUT =
Directs output of error information to S variables.

STRUCTURE-OUTPUT = *SYSINF
Default: the error information is to be output to an S variable to which the user has assigned the variable stream SYSINF.
If this variable stream is not active, no output is made to an S variable.

STRUCTURE-OUTPUT = *NONE
No output is made to an S variable.

STRUCTURE-OUTPUT = <composed-name 1..100>
Specifies the S variables in which the error information is to be stored.
The user must declare the variable as a dynamically defined structure (`/DECLARE-VARIABLE ..., MULTIPLE-ELEMENTS=*LIST,TYPE=*STRUCTURE).`

EXECUTION-MODE = *ASYNCHRONOUS(...)
The function is to be executed in a new batch task that is to be created. In this case, no information output takes place to an S variable.

TEXT-OUTPUT =
Controls the output of lists, logging and error messages.

TEXT-OUTPUT = *SYSLST
Output is made to a spoolout file that is printed at the end of the job and then deleted. The name of the spoolout file corresponds to the default setting of the SYSLST file for ENTER tasks.

TEXT-OUTPUT = <filename 1..54 without-gen-vers>
The text is to be output to the specified file. If the file already exists, the user must hold the appropriate write access right. The file must not be password-protected. It must be accessible locally and must not be a temporary file. The file is neither printed out nor deleted at the end of the job.

OPERATIONAL-MODE =
Specifies whether the function “Create an SM pubset” is really executed or only simulated or whether only the check function (consistency check) is to be performed.

OPERATIONAL-MODE = *OPERATION
Default value: the function is really executed.

OPERATIONAL-MODE = *SIMULATION
The function is only to be simulated.
The syntax of the operands specified is checked.
However, the operands MAX-ERRORS, MONJV, ERROR-INFORMATION and EXECUTION-MODE (with suboperands) do not have the meanings given in the operand descriptions.
Execution always takes place in the current task: if this is an interactive task, output is to SYSOUT and SYSLST; if it is a batch task, output is made only to SYSLST.

When the syntax check has been performed successfully, a table of volume sets with the specified logical attributes is output.

The original pubsets need not be accessible for this function; no disk operations are performed. Consequently, no checks that require disk availability are carried out, i.e. no name consistency check. The MRSCAT entries are not checked either.

OPERATIONAL-MODE = *CHECK-NAME-CONSISTENCY(...)
The check function (consistency check) is performed by SMPGEN.
Here the SF pubsets which were specified with the CONTROL-VOLUME-SET and ADD-VOLUME-SET operands are checked to see whether they can be transferred to the SM pubset that is to be created or can be converted.

PUBSET-STATE =
Specifies whether the consistency check is to performed for the current pubset session, i.e. possibly in parallel with other user accesses to the pubsets.

PUBSET-STATE = *IMPORTED
Default value: the consistency check is to be performed for the current pubset session. The pubsets have already been imported locally. The consistency check provides only a rough estimate which changes may since have rendered inaccurate.

PUBSET-STATE = *NOT-IMPORTED
SMPGEN is to allocate the pubsets exclusively for the consistency check. The pubsets are neither local nor imported from another system. During the consistency check they are locked against other accesses. This means that the result is correct at output time.

USER-ID =
Specifies the user ID(s) for which a pubset consistency check is to be performed.

USER-ID = *OWN
Default value: the consistency checks are to cover only the file names, job variables and guards under the user ID of the caller.
The system administration specific general checks referred to in the description of the statement (illegal files, S1 migration) are also performed when this value is specified.

USER-ID = *ALL

The consistency checks are to cover all user IDs. This specification is not allowed if one or more of the pubsets is/are imported in “shared slave” mode.

Format for nonprivileged users

```
CREATE-SYSTEM-MANAGED-PUBSET

 PUBSET=<catid>

 ,CONTROL-VOLUME-SET=<catid>

 ,ADD-VOLUME-SET=*NONE/list-poss(254):<catid>

 ,MAX-ERRORS=32767/<integer0..32767>

 ,MONJV=*NONE/ <filename 1..54 without-gen-vers>

 ,ERROR-INFORMATION=*DETAIL/*SUMMARY

 ,EXECUTION-MODE=*SYNCHRONOUS(...)/*ASYNCHRONOUS(...)

     *SYNCHRONOUS(...)

         |   TEXT-OUTPUT=*STD/*SYSOUT/*SYSLST/*NONE

         |   ,STRUCTURE-OUTPUT=*SYSINF/*NONE/<composed-name1..100>

     *ASYNCHRONOUS(...)

         |   TEXT-OUTPUT=*SYSLST/<filename 1..54 without-gen-vers>
```

Operands

PUBSET = <catid>
ID of an SM pubset which is to be checked to see whether it can be created.
The ID must differ from the IDs of the SF pubsets to be combined.
An identical MRSCAT entry may exist: such a pubset is not accessed (for example for checking purposes).
The specified ID is included in the check of the file name length and of whether guard conditions can be modified automatically.

CONTROL-VOLUME-SET = <catid>
ID of the SF pubset that is to become the control volume set of the new SM pubset. The ID becomes the volume set ID of the control volume set. It must differ from the ID of the SM pubset to be created. The SF pubset must have been imported into the local system.

ADD-VOLUME-SET =
Specifies whether, in addition to the pubset specified as the control volume set, further SF pubsets are to be checked for inclusion in the SM pubset.

ADD-VOLUME-SET = *NONE
Default value: no further SF pubsets are to be checked.

ADD-VOLUME-SET = <catid>
Specifies the IDs of further SF pubsets to be checked.
The pubset ID becomes the volume set ID.
All volume set IDs must be different from the ID of the control volume set and the selected SM pubset ID.
The SF pubsets must have been imported into the local system.

The volume set attributes, except USAGE, that can be specified as the substructure are described under the CONTROL-VOLUME-SET operand.

MAX-ERRORS = 32767 / <integer 0..32767>
The consistency check is aborted when the specified number of errors have occurred. The conflicts encountered before the function is aborted are output, provided ERROR-INFORMATION = *DETAIL was specified. 32767 is the default value.

MONJV =
Specifies a job variable used for monitoring.

MONJV = *NONE
Default value: no monitoring job variable is specified.

MONJV = <filename 1..54 without-gen-vers>
Specifies the name of the monitoring job variable. It must already be cataloged in an accessible pubset and must not be password-protected.

The job variables' status display is set as follows during the consistency check:

$I	at the start of the consistency check
$R	at the end of the consistency check if consistency exists
$E	at the end of the consistency check if consistency does not exist
$A	in the event of an incomplete consistency check

There is no point in specifying this operand unless you are using the software product JV.

> If the job variable cannot be supplied with values (e.g. because a password is set), the consistency check is performed anyway.
> If the job variable is to receive defined contents when execution of the function is started, the user must initialize this variable him/herself. If the function is aborted right at the start of execution, e.g. because a semantic error is detected, SMPGEN does not provide a value for the job variable.

ERROR-INFORMATION =
Controls the scope of the error information output in text and/or variable form.

ERROR-INFORMATION = *DETAIL
Default value: detailed error information is to be output. Not only is the number of name conflicts output, but also all names that occur more than once or names that are too long are listed.
In interactive mode, screen masks are output to the terminal that offer the option of suppressing follow-up screens or detail screens, or to switch to SYSLST (see section“Outputs to screen masks”).
In batch mode and to SYSLST, the output is made in the same format, but here there is no possibility of intervening.

ERROR-INFORMATION = *SUMMARY
No detailed error information is to be output, i.e. no lists of name conflicts but only the number of conflicts in the various name classes.

EXECUTION-MODE =
Specifies whether the function is to be executed in the current task or in a new batch task to be created.

EXECUTION-MODE = *SYNCHRONOUS(...)
Default value: the function is to be executed in the current task.

TEXT-OUTPUT =
Controls the output destination for data output in text format, i.e. lists, logging and error messages.

TEXT-OUTPUT = *STD
Default value: the output is to be carried out according to default setting. In interactive mode, output is to SYSOUT and to SYSLST. In batch mode, output is to SYSLST.

TEXT-OUTPUT = *SYSOUT
Output is directed to SYSOUT. This specification may only be made in interactive mode. SYSOUT may not have been reassigned to a file.
If SYSOUT output is desired, screen masks are output and these must be confirmed by pressing the send key.
If numerous errors occur, it is possible to apply or suppress a detailed error log for SYSLST later.

TEXT-OUTPUT = *SYSLST
Output is directed to SYSLST.

TEXT-OUTPUT = *NONE
No outputs in the form of lists or messages are desired. It makes sense to specify this value when the STRUCTURE-OUTPUT operand is used instead.
Error messages and messages reporting the start and success of the function cannot be suppressed; they are output to SYSOUT even if TEXT-OUTPUT=*NONE is specified.

STRUCTURE-OUTPUT =
Directs output of error information to S variables.

STRUCTURE-OUTPUT = *SYSINF
Default: the error information is to be output to an S variable to which the user has assigned the variable stream SYSINF.
If this variable stream is not active, no output is made to an S variable.

STRUCTURE-OUTPUT = *NONENo output is made to an S variable.

STRUCTURE-OUTPUT = <composed-name 1..100>
Specifies the S variables in which the error information is to be stored.
The user must declare the variable as a dynamically defined structure (`/DECLARE-VARIABLE ..., MULTIPLE-ELEMENTS=*LIST,TYPE=*STRUCTURE`).

EXECUTION-MODE = *ASYNCHRONOUS(...)
The function is to be executed in a new batch task that is to be created. In this case, information output to an S variable is not supported.

TEXT-OUTPUT =
Controls the output of lists, logging and error messages.

TEXT-OUTPUT = *SYSLST

Output is made to a spoolout file that is printed at the end of the job and then deleted. The name of the spoolout file corresponds to the default setting of the SYSLST file for ENTER tasks.

TEXT-OUTPUT = <filename 1..54 without-gen-vers>

The text is to be output to the specified file. If the file already exists, the user must hold the appropriate write access right. The file must not be password-protected. It must be accessible locally and must not be a temporary file. The file is neither printed out nor deleted at the end of the job.

14.7.2.2 MODIFY-SYSTEM-MANAGED-PUBSET - Extending an existing SM pubset

- For privileged users

  The MODIFY-SYSTEM-MANAGED-PUBSET statement either extends an existing SM pubset or checks whether specified SF pubsets can be included in an existing SM pubset without conflict.

  The extension of an SM pubset is not yet fully completed after the MODIFY-SYSTEM-MANAGED-PUBSET statement has been executed, but only after a normal import of an SM pubset which follows the SMPGEN run. As a result a renewed extension is only possible after the SM pubset has been imported.

- For nonprivileged users

  The MODIFY-SYSTEM-MANAGED-PUBSET statement provides only the check function for nonprivileged users. Here a consistency check decides whether the specified SF pubsets can be included in an existing SM pubset. The checks relate only to the caller's user ID. Callers can thus obtain information on which name conflicts they have caused and how they can resolve these.

Outputs

The outputs of this statement are described in the section "Outputs (Creating new SM pubsets)" or in the section "Outputs (Check function)"; the mask descriptions start on "Outputs to screen masks".

Operation

How this statement functions is described in the section "Extending an existing SM pubset".

> **WARNING**!
>
> - MODIFY-SYSTEM-MANAGED-PUBSET OPERATIONAL-MODE= *OPERATION should never be called without a previous FDDRL backup of the SF pubsets to be converted, since an abort due to an error would destroy all the SF pubsets and render them inaccessible.
> - A logical backup using HSMS is also recommended. Here, a new backup archive can be created which can be used for the creation of the HSMS environment for the extended SM pubset. For details of how to adapt the HSMS environment, refer to the description of SMS migration in the "System-Managed Storage" manual [8 (Related publications)].
> - SPOOL jobs for files that are cataloged in the SF pubsets to be converted should no longer be present, since they can no longer be carried out after renaming the catid.

Simulation and check mode

If the operand OPERATIONAL-MODE=*SIMULATION is specified, the function described is not really executed. Only a syntax check is carried out and the specified volume sets are listed.

If the OPERATIONAL-MODE=*CHECK-NAME-CONSISTENCY operand is specified, only the check function is performed.

For more details, refer to the operand descriptions.

Format for privileged users

```
MODIFY-SYSTEM-MANAGED-PUBSET

 PUBSET=<catid>

 ,ADD-VOLUME-SET=list-poss(254):<catid>(...)

      <catid>(...)

           |    AVAILABILITY=*STD/*HIGH

           |    ,USAGE = *STD / *HSMS-CONTROLLED

           |    ,PERFORMANCE-ATTR=*STD/*PARAMETERS(...)

           |         *PARAMETERS(...)

           |              |    PERFORMANCE=list-poss(3):*STD/*HIGH/ *VERY-HIGH

           |              |    ,WRITE-CONSISTENCY=*BY-CLOSE/*IMMEDIATE

 ,S1-MIGRATED-FILES=*NOT-ALLOWED/*ALLOWED

 ,MAX-ERRORS=32767/<integer0..32767>

 ,MONJV=*NONE/ <filename 1..54 without-gen-vers>

 ,ERROR-INFORMATION=*DETAIL/*SUMMARY

 ,EXECUTION-MODE=*SYNCHRONOUS(...)/*ASYNCHRONOUS(...)

      *SYNCHRONOUS(...)

           |    TEXT-OUTPUT=*STD/*SYSOUT/*SYSLST/*NONE

           |    ,STRUCTURE-OUTPUT=*SYSINF/*NONE/<composed-name1..100>

      *ASYNCHRONOUS(...)

           |    TEXT-OUTPUT=*SYSLST/<filename 1..54 without-gen-vers>

 ,OPERATIONAL-MODE=*OPERATION/*SIMULATION / *CHECK-NAME-CONSISTENCY(...)

      *CHECK-NAME-CONSISTENCY(...)

           |    PUBSET-STATE=*IMPORTED/*NOT-IMPORTED

           |    ,USER-ID=*OWN/*ALL
```

Operands

PUBSET = <catid>

ID of the existing SM pubset that is to be extended or checked to see whether it can be extended. The ID must differ from the IDs of the SF pubsets to be combined.

The specified ID is included in the check of the file name length and of whether guard conditions can be modified automatically.

ADD-VOLUME-SET =
Specifies which further SF pubsets are to be added to the SM pubset.

ADD-VOLUME-SET = <catid>(...)
IDs of further SF pubsets to be added to the SM pubset as a volume set. The pubset ID becomes the volume set ID. All the volume set IDs must be different from each other and from the selected SM pubset ID.

The volume sets must satisfy the requirements described in the section “Requirements foroperation of SMPGEN”. If not just a check is to take place (OPERATIONAL-MODE= *CHECK-NAME-CONSISTENCY operand) but also a conversion, the SF pubsets may not be imported in either the local or any other system; however, MRSCAT entries with the correct device type must already exist for this purpose.

If just the test function is performed (OPERATIONAL-MODE=*CHECK-NAME-CONSISTENCY), the specification for the subsequent volume set attributes is ignored.

AVAILABILITY =
Defines the degree of availability of files with respect to the failsafe capacity of the volume set.
The availability is a logical attribute of the volume set that is used as a selection criterion when choosing storage space for archiving files. Changing the volume set attribute is a job that must be commissioned explicitly by systems support.

AVAILABILITY = *STD
Default value: the volume set does not offer enhanced availability.

AVAILABILITY = *HIGH
The volume set offers high availability.
It is the responsibility of systems support to check that failsafe performance is actually guaranteed physically and, for example, to activate DRV or DUALCOPY mode before starting conversion.

USAGE =
Specifies the usage of the volume set.

USAGE = *STD
The volume set is to be used as the storage location for the default files. Work files (WORK file attribute) may not be stored there.

USAGE = *HSMS-CONTROLLED
The volume set is used by the HSMS subsystem to implement storage level S1 and the backup volume set of an SM pubset. Users may not store any files there.

PERFORMANCE-ATTR =
Defines the performance attributes of the volume set for file access. These attributes are logical attributes which are used as a selection criterion when choosing the storage space. The available hardware (e.g. cache, SSD) is not verified.

PERFORMANCE-ATTR = *STD
Default value: no enhanced performance attributes for file access are to be defined for the volume set.

PERFORMANCE-ATTR = *PARAMETERS(...)
The performance attributes that apply to the volume set are defined via the following substructure:

PERFORMANCE =
Describes the I/O performance profile of the volume set. It is possible to specify a list of up to three values.

PERFORMANCE = *STD
Default value: the volume set provides no enhanced I/O performance.

PERFORMANCE = *HIGH
The volume set provides enhanced I/O performance.

PERFORMANCE = *VERY-HIGH
The volume set provides very high I/O performance.

WRITE-CONSISTENCY =
Defines whether the hardware used to increase the I/O performance guarantees data consistency even in the event of a system crash.

WRITE-CONSISTENCY = *BY-CLOSE
Default value: write I/Os with enhanced performance are possible only in trade-off against immediate crash-proof data storage; until the file is closed the data is not physically stored on a medium which guarantees data retention even if the system crashes.

WRITE-CONSISTENCY = *IMMEDIATE
Also for write I/Os with enhanced performance, the data is immediately stored on a medium which guarantees data retention even in the event of a system crash.

S1-MIGRATED-FILES =
Specifies whether if one of the pubsets to be checked contains cataloged files which are currently migrated to S1 level this is to be defined as a conflict.

S1-MIGRATED-FILES = *NOT-ALLOWED
Default value: the case described above is deemed to be a conflict.

S1-MIGRATED-FILES = *ALLOWED

The case described above is not deemed to be a conflict.
The S1-migrated files are no longer accessible following conversion, but must be restored from a backup file. Their catalog entries are retained so that they can be selectively moved to the S1 level of the SM pubset after creation of the HSMS environment with REPAIR-CATALOG-BY-RESTORE.

MAX-ERRORS = 32767 / <integer 0..32767>
The consistency check is aborted when the specified number of errors have occurred. The conflicts encountered before the function is aborted are output, provided ERROR-INFORMATION = *DETAIL was specified. 32767 is the default value.

MONJV =
Specifies a job variable used for monitoring.

MONJV = *NONE
Default value: no monitoring job variable is specified.

MONJV = <filename 1..54 without-gen-vers>
Specifies the name of the monitoring job variable. It must already be cataloged in an accessible pubset and must not be password-protected.

The job variables' status display is set as follows during conversion (OPERATIONAL-MODE=*OPERATION):

$I at the start of processing

$R on successful completion of processing

$A on termination with error

The job variables' status display is set as follows during the consistency check (OPERATIONAL-MODE=*CHECK-NAME- CONSISTENCY):

$I at the start of the consistency check

$R at the end of the consistency check if consistency exists

$E at the end of the consistency check if consistency does not exist

$A in the event of an incomplete consistency check

There is no point in specifying this operand unless you are using the software product JV.

> If the job variable cannot be supplied with values (e.g. because a password is set), the consistency check is performed anyway. If the job variable is to receive defined contents when execution of the function is started, the user must initialize this variable him/herself. If the function is aborted right at the start of execution, e.g. because a semantic error is detected, SMPGEN does not provide a value for the job variable.If the job variable is contained on one of the pubsets that are to be combined and OPERATIONAL-MODE=*CHECK-NAME-CONSISTENCY(PUBSET-STATE= *IMPORTED) is not specified, the SMPGEN job is rejected.

ERROR-INFORMATION =
Controls the scope of the error information output in text and/or variable form.

ERROR-INFORMATION = *DETAIL
Default value: detailed error information is to be output. Not only is the number of name conflicts output, but also all names that occur more than once or names that are too long are listed.
In interactive mode, screen masks are output to the terminal that offer the option of suppressing follow-up screens or detail screens, or to switch to SYSLST (see section"Outputs to screen masks").
In batch mode and to SYSLST, the output is made in the same format, but here there is no possibility of intervening.

ERROR-INFORMATION = *SUMMARY
No detailed error information is to be output, i.e. no lists of name conflicts but only the number of conflicts in the various name classes.

EXECUTION-MODE =
Specifies whether the function is to be executed in the current task or in a new batch task to be created.

EXECUTION-MODE = *SYNCHRONOUS(...)
Default value: the function is to be executed in the current task.

TEXT-OUTPUT =
Controls the output destination for data output in text format, i.e. lists, logging and error messages.

TEXT-OUTPUT = *STD
Default value: the output is to be carried out according to default setting. In interactive mode, output is to SYSOUT and to SYSLST. In batch mode, output is to SYSLST.

TEXT-OUTPUT = *SYSOUT
Output is directed to SYSOUT.
This specification may only be made in interactive mode. SYSOUT may not have been reassigned to a file.
If SYSOUT output is desired, screen masks are output and these must be confirmed by pressing the send key.
If numerous errors occur, it is possible to apply or suppress a detailed error log for SYSLST later.

TEXT-OUTPUT = *SYSLST
Output is directed to SYSLST.

TEXT-OUTPUT = *NONE
No outputs in the form of lists or messages are desired. It makes sense to specify this value when the STRUCTURE-OUTPUT operand is used instead.

Error messages and messages reporting the start and success of the function cannot be suppressed; they are output to SYSOUT even if TEXT-OUTPUT=*NONE is specified.

STRUCTURE-OUTPUT =
Directs output of error information to S variables.

STRUCTURE-OUTPUT = *SYSINF
Default: the error information is to be output to an S variable to which the user has assigned the variable stream SYSINF.
If this variable stream is not active, no output is made to an S variable.

STRUCTURE-OUTPUT = *NONE
No output is made to an S variable.

STRUCTURE-OUTPUT = <composed-name 1..100>
Specifies the S variables in which the error information is to be stored.
The user must declare the variable as a dynamically defined structure (`/DECLARE-VARIABLE ..., MULTIPLE-ELEMENTS=*LIST, TYPE=*STRUCTURE`).

EXECUTION-MODE = *ASYNCHRONOUS(...)
The function is to be executed in a new batch task that is to be created.
In this case, information output to an S variable is not supported.

TEXT-OUTPUT =
Controls the output of lists, logging and error messages.

TEXT-OUTPUT = *SYSLST
Output is made to a spoolout file that is printed at the end of the job and then deleted. The name of the spoolout file corresponds to the default setting of the SYSLST file for ENTER tasks.

TEXT-OUTPUT = <filename 1..54 without-gen-vers>
The text is to be output to the specified file. If the file already exists, the user must hold the appropriate write access right. The file must not be password-protected. It must be accessible locally and must not be a temporary file. The file is neither printed out nor deleted at the end of the job.

OPERATIONAL-MODE =
Specifies whether the function “Extend an SM pubset” is really executed or only simulated or whether only the check function (consistency check) is to be performed.

OPERATIONAL-MODE = *OPERATION
Default value: the function is really executed.

OPERATIONAL-MODE = *SIMULATION
The function is only to be simulated.
The syntax of the operands specified is checked.
However, the operands MAX-ERRORS, MONJV, ERROR-INFORMATION and EXECUTION-MODE (with suboperands) do not have the meanings given in the operand descriptions.
Execution always takes place in the current task: if this is an interactive task, output is to SYSOUT and SYSLST; if it is a batch task, output is made only to SYSLST.

When the syntax check has been performed successfully, a table of volume sets with the specified logical attributes is output.

The original pubsets need not be accessible for this function; no disk operations are performed. Consequently, no checks that require disk availability are carried out, i.e. no name consistency check. The MRSCAT entries are not checked either.

OPERATIONAL-MODE = *CHECK-NAME-CONSISTENCY(...)
The check function (consistency check) is performed by SMPGEN.
Here the SF pubsets which were specified with the CONTROL-VOLUME-SET and ADD-VOLUME-SET operands are checked to see whether they can be combined with the specified SM pubset or converted.

PUBSET-STATE =
Specifies whether the consistency check is to performed for the current pubset session, i.e. possibly in parallel with other user accesses to the pubsets.

PUBSET-STATE = *IMPORTED
Default value: the consistency check is to be performed for the current pubset session. The pubsets have already been imported locally. The consistency check provides only a rough estimate which changes may since have rendered inaccurate.

PUBSET-STATE = *NOT-IMPORTED
SMPGEN is to allocate the pubsets exclusively for the consistency check. The pubsets are neither local nor imported from another system. During the consistency check they are locked against other accesses. This means that the result is correct at output time.

USER-ID =
Specifies the user ID(s) for which a pubset consistency check is to be performed.

USER-ID = *OWN

Default value: the consistency checks are to cover only the file names, job variables and guards under the user ID of the caller.

The general checks referred to in the description of the statement for privileged users (illegal files, S1 migration) are also performed when this value is specified.

USER-ID = *ALL

The consistency checks are to cover all user IDs. This specification is not allowed if one or more of the pubsets is /are imported in “shared slave” mode.

Format for nonprivileged users

```
MODIFY-SYSTEM-MANAGED-PUBSET

 PUBSET=<catid>

 ,ADD-VOLUME-SET=list-poss(254):<catid>

 ,MAX-ERRORS=32767/<integer0..32767>

 ,MONJV=*NONE/ <filename 1..54 without-gen-vers>

 ,ERROR-INFORMATION=*DETAIL/*SUMMARY

 ,EXECUTION-MODE=*SYNCHRONOUS(...)/*ASYNCHRONOUS(...)

     *SYNCHRONOUS(...)

         |    TEXT-OUTPUT=*STD/*SYSOUT/*SYSLST/*NONE

         |    ,STRUCTURE-OUTPUT=*SYSINF/*NONE/<composed-name1..100>

     *ASYNCHRONOUS(...)

         |    TEXT-OUTPUT=*SYSLST/<filename 1..54 without-gen-vers>
```

Operands

PUBSET = <catid>
ID of the existing SM pubset which is to be checked to see whether it can be extended. The ID must differ from the IDs of the SF pubsets to be combined.

The specified ID is included in the check of the file name length and of whether guard conditions can be modified automatically.

ADD-VOLUME-SET =
Specifies which further SF pubsets are to be added to the SM pubset.

ADD-VOLUME-SET = <catid>
IDs of further SF pubsets to be added to the SM pubset as a volume set. The pubset ID becomes the volume set ID. All the volume set IDs must be different from each other and from the selected SM pubset ID.

The volume sets must satisfy the requirements described in the section “Requirements foroperation of SMPGEN”. If not just a check is to take place (OPERATIONAL-MODE= *CHECK-NAME-CONSISTENCY operand) but also a conversion, the SF pubsets may not be imported in either the local or any other system; however, MRSCAT entries with the correct device type must already exist for this purpose.

MAX-ERRORS = 32767 / <integer 0..32767>
The consistency check is aborted when the specified number of errors have occurred. The conflicts encountered before the function is aborted are output, provided ERROR-INFORMATION = *DETAIL was specified. 32767 is the default value.

MONJV =
Specifies a job variable used for monitoring.

MONJV = *NONE
Default value: no monitoring job variable is specified.

MONJV = <filename 1..54 without-gen-vers>
Specifies the name of the monitoring job variable. It must already be cataloged in an accessible pubset and must not be password-protected.

The job variables' status display is set as follows during the consistency check:

$I	at the start of the consistency check
$R	at the end of the consistency check if consistency exists
$E	at the end of the consistency check if consistency does not exist
$A	in the event of an incomplete consistency check

There is no point in specifying this operand unless you are using the software product JV.

> If the job variable cannot be supplied with values (e.g. because a password is set), the consistency check is performed anyway. If the job variable is to receive defined contents when execution of the function is started, the user must initialize this variable him/herself. If the function is aborted right at the start of execution, e.g. because a semantic error is detected, SMPGEN does not provide a value for the job variable.

ERROR-INFORMATION =
Controls the scope of the error information output in text and/or variable form.

ERROR-INFORMATION = *DETAIL
Default value: detailed error information is to be output. Not only is the number of name conflicts output, but also all names that occur more than once or names that are too long are listed.
In interactive mode, screen masks are output to the terminal that offer the option of suppressing follow-up screens or detail screens, or to switch to SYSLST (see section"Outputs to screen masks").
In batch mode and to SYSLST, the output is made in the same format, but here there is no possibility of intervening.

ERROR-INFORMATION = *SUMMARY
No detailed error information is to be output, i.e. no lists of name conflicts but only the number of conflicts in the various name classes.

EXECUTION-MODE =
Specifies whether the function is to be executed in the current task or in a new batch task to be created.

EXECUTION-MODE = *SYNCHRONOUS(...)
Default value: the function is to be executed in the current task.

TEXT-OUTPUT =
Controls the output destination for data output in text format, i.e. lists, logging and error messages.

TEXT-OUTPUT = *STD
Default value: the output is to be carried out according to default setting. In interactive mode, output is to SYSOUT and to SYSLST. In batch mode, output is to SYSLST.

TEXT-OUTPUT = *SYSOUT
Output is directed to SYSOUT. This specification may only be made in interactive mode. SYSOUT may not have been reassigned to a file.
If SYSOUT output is desired, screen masks are output and these must be confirmed by pressing the send key.
If numerous errors occur, it is possible to apply or suppress a detailed error log for SYSLST later.

TEXT-OUTPUT = *SYSLST
Output is directed to SYSLST.

TEXT-OUTPUT = *NONE
No outputs in the form of lists or messages are desired. It makes sense to specify this value when the STRUCTURE-OUTPUT operand is used instead.
Error messages and messages reporting the start and success of the function cannot be suppressed; they are output to SYSOUT even if TEXT-OUTPUT=*NONE is specified.

STRUCTURE-OUTPUT =
Directs output of error information to S variables.

STRUCTURE-OUTPUT = *SYSINF
Default: the error information is to be output to an S variable to which the user has assigned the variable stream SYSINF.
If this variable stream is not active, no output is made to an S variable.

STRUCTURE-OUTPUT = *NONE
No output is made to an S variable.

STRUCTURE-OUTPUT = <composed-name 1..100>
Specifies the S variables in which the error information is to be stored.
The user must declare the variable as a dynamically defined structure (`/DECLARE-VARIABLE ..., MULTIPLE-ELEMENTS=*LIST, TYPE=*STRUCTURE`).

EXECUTION-MODE = *ASYNCHRONOUS(...)
The function is to be executed in a new batch task that is to be created.
In this case, information output to an S variable is not supported.

TEXT-OUTPUT =
Controls the output of lists, logging and error messages.

TEXT-OUTPUT = *SYSLST
Output is made to a spoolout file that is printed at the end of the job and then deleted. The name of the spoolout file corresponds to the default setting of the SYSLST file for ENTER tasks.

TEXT-OUTPUT = <filename 1..54 without-gen-vers>
The text is to be output to the specified file. If the file already exists, the user must hold the appropriate write access right. The file must not be password-protected. It must be accessible locally and must not be a temporary file. The file is neither printed out nor deleted at the end of the job.

14.8 Error behavior

SMPGEN creates several WORK files when processing statements; these normally exist only during the SMPGEN session.

In the event of a system failure or recovery errors, such files can of course exist beyond the SMPGEN session. They can be recognized by the prefix SYSWRK.SMPGEN or S.SMPGE. SYSWRK.SMPGEN... files must be deleted by systems support before starting a new SMPGEN session.

Special points when creating or extending SM pubsets

SMPGEN attempts to rule out situations that prevent formation of an SM pubset as early as possible by performing a consistency check. Even after a successful check, the pubset structures are not changed initially, only the new management files are created.
In the event of an extension the SM pubset's management data is saved by copying it. During conversion, the old administration files are also retained in the main, so that SMPGEN can restore the old environment in the event that conversion fails or that the task is aborted. The pubsets involved are then returned to their old states.
Only the SYSPBN and EAM files could be deleted.

Only in the last phase (after all checks have been completed successfully and disk space requirements are satisfied) is it no longer possible to return to the old state of the SF pubsets. A fatal error in this phase (just like a system crash in the middle phase of conversion) would mean that the pubsets could no longer be imported; they have to be reconstructed from the previous FDDRL backup.
During the critical phase of conversion, the pubsets and volume sets are marked as not importable.

If an import is not possible following a system crash, either as an SF or an SM pubset, it can be assumed that all relevant information is retained and a consistent disk status achieved.

If the system halts while an SM pubset is being imported but the pubset can still be imported as an SM pubset, it can be assumed that a consistent disk status has been achieved. However, depending on when the system halted the SM pubset has either its initial status or the correct status which has been extended by SF pubsets.

If, in the event of a conversion failure, the pubsets were converted back to SF pubset and it was not possible to restore the old GUARDS environment, the pubsets are still marked as importable. Following import as SF pubsets, the objects that are protected by guards cannot be addressed. The old GUARDS environment can be restored by systems support, however, by using `/CHANGE-GUARD-FILE` to assign the guards catalogs renamed by SMPGEN. The catalogs reside under the name `SYSWRK.SMPGEN.SYSCAT.GUARDS.<pubsetid>`.

The new or extended CMS management files are deleted during reconversion to SF pubsets or placed in the initial status.
If they cannot be deleted during reconversion for whatever reason, this does not prevent a new SMPGEN run.
If the same volume set is to become the control volume set, any new CMS administration files still present on it are used again. If another pubset is to become the control volume set, the CMS implicitly deletes the CMS administration files that still reside on the “simple” volume set as leftovers from the previous SMPGEN run.

When exporting the pubsets, the following message may be output for each pubset involved, if guards are used:

```
PRO6009   ERROR WHEN CLOSING GUARDS CATALOG
```

This message can be ignored.

Special aspects regarding SMPGEN statements with which all the pubsets involved are imported by SMPGEN itself

(e.g. CREATE-SYSTEM-MANAGED-PUBSET mit OPERATIONAL-MODE=*OPERATION or with OPERATIONAL-MODE=*CHECK-NAME-CONSISTENCY(PUBSET-STATE= *NOT-IMPORTED))

In the event of termination due to an error, or abnormal program/task termination, the MRSCAT entries are cleaned up. Only if the recovery routine aborts do the MRSCAT entries retain the status “inaccessible/SMPGEN processing in progress/cannot be deleted”. The MRSCAT entries can then not be deleted. This means that the pubset can no longer be imported in this session and that the SMPGEN session cannot be started a second time.

STEP termination

If a syntactically or semantically incorrect SMPGEN statement is input, or if the statement cannot be fully processed, the program moves to the next STEP statement, if the error occurs within the program and not in the asynchronously generated task.

If an error occurs that poses problems for the overall program execution (e.g. no information output possible, end of file on SYSDTA), the program ends with STEP termination, i.e. all statements up to the next `SET-JOB-STEP` or `EXIT-JOB` are skipped within a procedure.

14.9 Outputs to screen masks

The screen masks are made up of 80 columns and 24 lines, and have the following structure:

- Header with title
- Data section for output of object attributes (output fields)
  The output fields are supplied with the appropriate values by SMPGEN. They cannot be overwritten by the user. Exceptions are the fields “MORE” in masks 3, 5 and 7, for which inputs are possible.
- Footer with input option (e.g. change the output medium)
  When outputting to SYSLST, or when executing batch tasks, the footer is omitted. In this case, the output is always complete (equivalent to the input MORE=Y for output to the terminal).

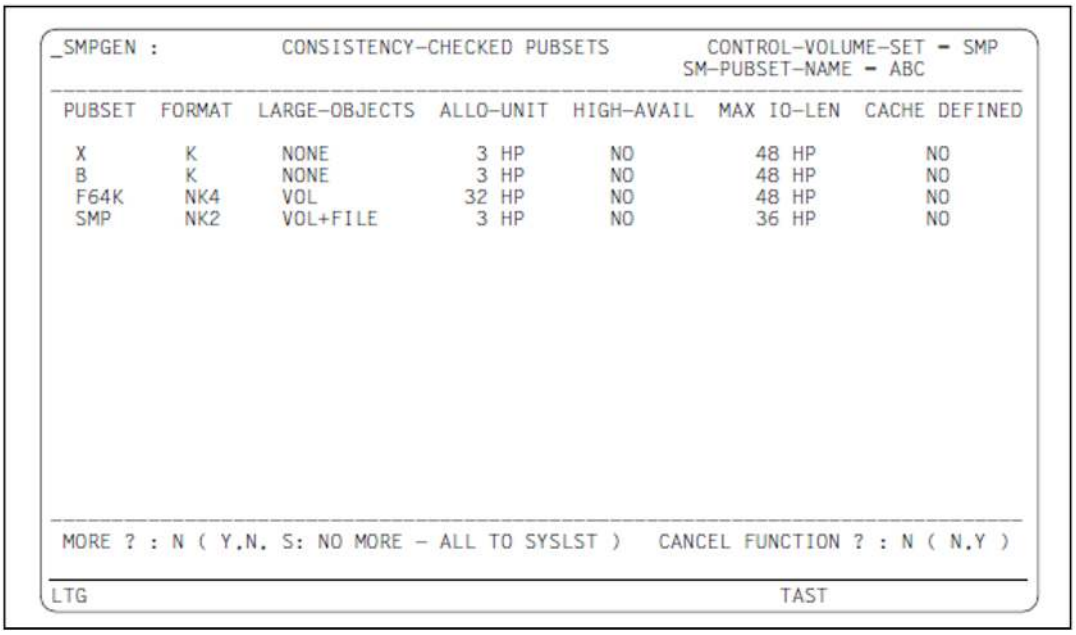
The character in line 1, column 1 (support of the local hardcopy function) can be overwritten in order to position the cursor on it.
To continue the program run, press the DUE key on the terminal. The K1, K2 and K3 keys are ignored.

The following masks are available:

1. CONSISTENCY-CHECKED PUBSETS
2. INADMISSIBLE FILES
3. FILE-NAME CONFLICTS
4. MULTIPLE FILE-NAME CONFLICTS
5. JV-NAME CONFLICTS
6. MULTIPLE JV-NAME CONFLICTS
7. GUARD-NAME CONFLICTS
8. MULTIPLE GUARD-NAME CONFLICTS
9. FILE-NAME LENGTH ERRORS
10. JV-NAME LENGTH ERRORS
11. GUARD CONDITION MODIFICATION FAILURES
12. VOLUME-SET TABLE 1
13. VOLUME-SET TABLE

MASK 1: CONSISTENCY-CHECKED PUBSETS

This mask is generated with the CREATE- and MODIFY-SYSTEM-MANAGED-PUBSET statements when performing only the check function or when the CHECK-NAME-CONSISTENCY statement is executed.



HEADER

CONTROL-VOLUME-SET

Pubset ID input for the operand of the same name;
*NONE is shown as *NON

SM-PUBSET-NAME

SM pubset ID entered via an operand

DATA

PUBSET

Pubset ID of the checked pubset

FORMAT

Disk format determined: K / NK2 / NK4

LARGE-OBJECTS

Indicates whether volumes or files that are larger than 32 Gbytes are permitted:

VOL+FILE:	Volumes and files that are larger than 32 Gbytes are permitted
VOL:	Only volumes that are larger than 32 Gbytes are permitted

NONE: Neither volumes nor files that are larger than 32 Gbytes are permitted

ALLOCATION-UNIT

Allocation unit in half pages (2KB each)

HIGH-AVAIL

Enhanced availability: YES (DRV operation, RAID etc.) / NO

MAXIMAL IO-LENGTH

Maximum transfer length for disk input/output

CACHE DEFINED

Has a cache medium been defined?: YES / NO

FOOTER

MORE

This field is preset to “Y” if more pubsets than specified in the current statement are checked than are shown on the current and previous screen pages.
Overwriting this field with “N” prevents output of further pubsets at the terminal.Overwriting this field with “S” means that no further pubsets are output to the terminal but the complete pubset information is output to SYSLST independent of the OUTPUT parameters.

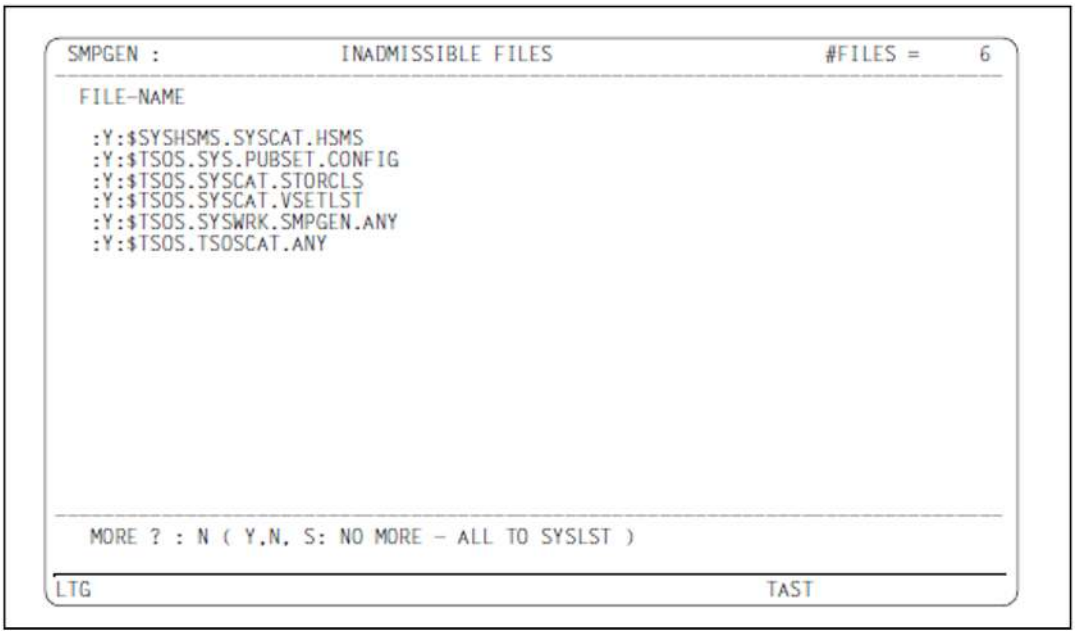
CANCEL FUNCTION

This field is preset to “N”. If overwritten with another value, the function is aborted.

MASK 2: INADMISSIBLE FILES

This mask is generated for privileged users of SMPGEN when illegal files are present in the pubsets to be checked.

These are the files referred to in the description of the check function, which cannot be used in the SM pubset. They must not be cataloged in the pubsets, because SMPGEN sets up work files or new administration files under these names, or because the files cannot be used as paging files in the SM pubset.



HEADER

#FILES

Number of illegal files

DATA

FILE-NAME

Path name

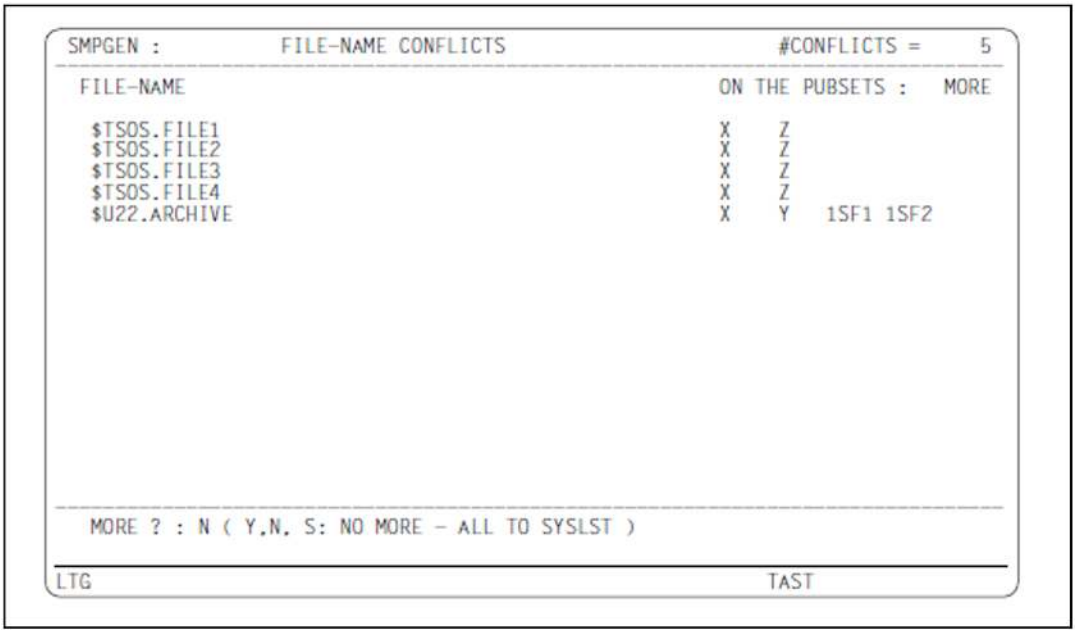
FOOTER

MORE

This field is preset to “Y” if more files are present than are shown on the current and previous screen pages. Overwriting this field with “N” prevents output of further pubsets at the terminal. Overwriting this field with “S” means that no further file names are output to the terminal but that all the illegal files are listed, independent of the TEXT-OUTPUT operand. Other outputs are continued as normal.

MASK 3: FILE-NAME CONFLICTS

This mask is created if file name conflicts arise. If file names occur on more that four pubsets, a full pubset screen is output for each of the file names immediately after this screen (mask 4), unless otherwise defined by the user in the overwrite field on the right margin.
The next screen to output file name conflicts, mask 3, is not displayed until this operation has been completed.



HEADER

#CONFLICTS

Number of files that are cataloged on several of the checked pubsets

DATA

FILE-NAME

File name without catalog ID

ON THE PUBSETS

Catalog ID of relevant pubset

MORE

This field is preset to “Y” if the name exists on more that four pubsets, otherwise blank. Overwriting this field with “N” or blanks prevents mask 4 being output at the terminal. Overwriting this field with “S” means that mask 4 is no longer output to the terminal, but to SYSLST, regardless of what is specified for the TEXT-OUTPUT operand.

FOOTER

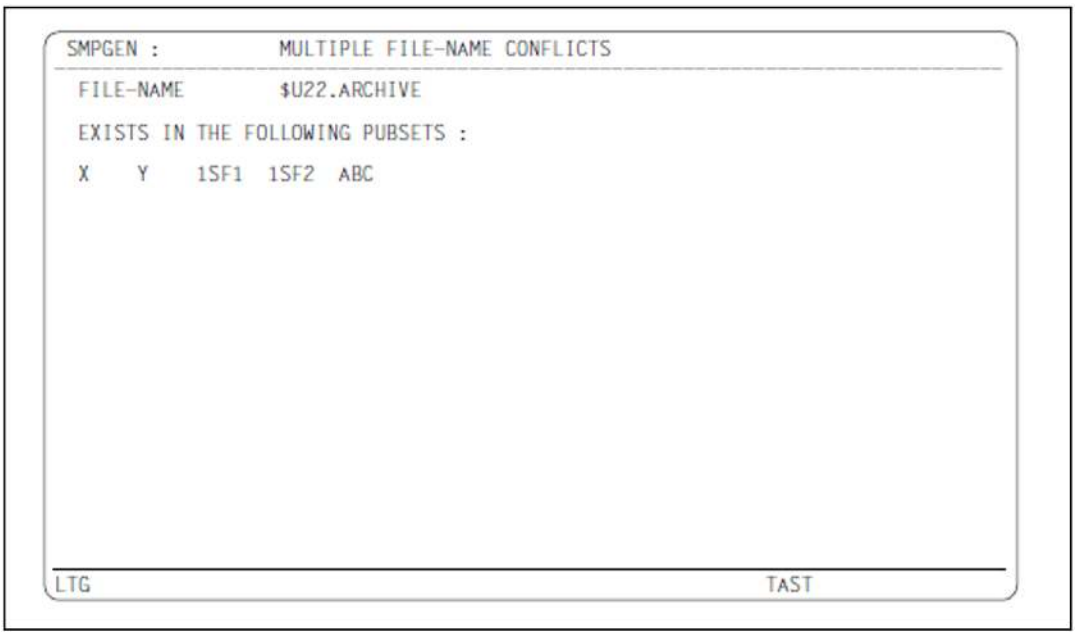
MORE

This field is preset to “Y” if more file name conflicts are present than are shown on the current and previous screen pages.
Overwriting this field with “N” prevents output of further file names at the terminal and to SYSLST.
Overwriting this field with “S” means that no further file names are output to the terminal but that all the file name conflicts are output to SYSLST, independent of the TEXT-OUTPUT operand. Other outputs are continued as normal.

MASK 4: MULTIPLE FILE-NAME CONFLICTS

This mask is generated when file names occur on more than four pubsets. It is output after mask 3 for each file name marked there with MORE=Y if the field was not overwritten with “N”.

The pubsets already displayed in mask 3 are also output. Inputs are not possible.



DATA

FILE-NAME

File name without catalog ID

EXISTS IN THE FOLLOWING PUBSETS

Pubset IDs of the particular pubset

MASK 5: JV NAME CONFLICTS

The structure, meaning and reaction options of this mask are analogous to those of mask 3, with the exception that conflicts among the names of job variables are considered here.

MASK 6: MULTIPLE JV NAME CONFLICTS

The structure, meaning and reaction options of this mask are analogous to those of mask 4, with the exception that conflicts among the names of job variables are considered here.

MASK 7: GUARD-NAME CONFLICTS

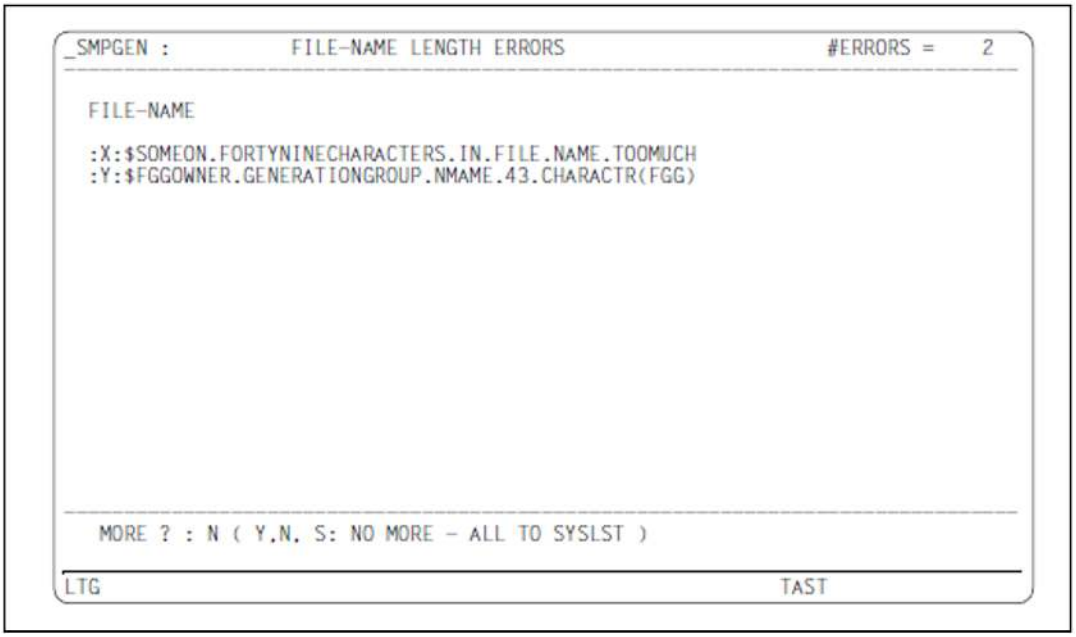
The structure, meaning and reaction options of this mask are analogous to those of mask 3, with the exception that conflicts among the guard names are considered here. A connecting line consisting of periods bridges the space between guard names and pubsets affected.

MASK 8: MULTIPLE GUARD-NAME CONFLICTS

The structure, meaning and reaction options of this mask are analogous to those of mask 4, with the exception that conflicts among the guard names are considered here.

MASK 9: FILE-NAME LENGTH ERRORS

This mask is generated when file names are encountered which exceed the maximum permissible file name length of 54 characters when replacing the old pubset ID with a new one (PUBSET-ID-LENGTH, or for which the file generation group names are longer than 47 characters. The individual file generations are not displayed separately.



HEADER

#ERRORS

Number of file names that exceed the permissible length.
Path names that differ only with respect to the catalog ID are treated as different file names.

DATA

FILE-NAME

Path name (with catalog ID); for file generation group names followed by the generation number

FOOTER

MORE

This field is preset to “Y”, if more files are affected than are shown on the current and previous screen pages.
Overwriting this field with “N” prevents output of further warnings of this type at the terminal.
Overwriting this field with “S” means that no further file names are output to the terminal but that all the file names that are too long are output to SYSLST, independent of the TEXT-OUTPUT operand. Other outputs are continued as normal.

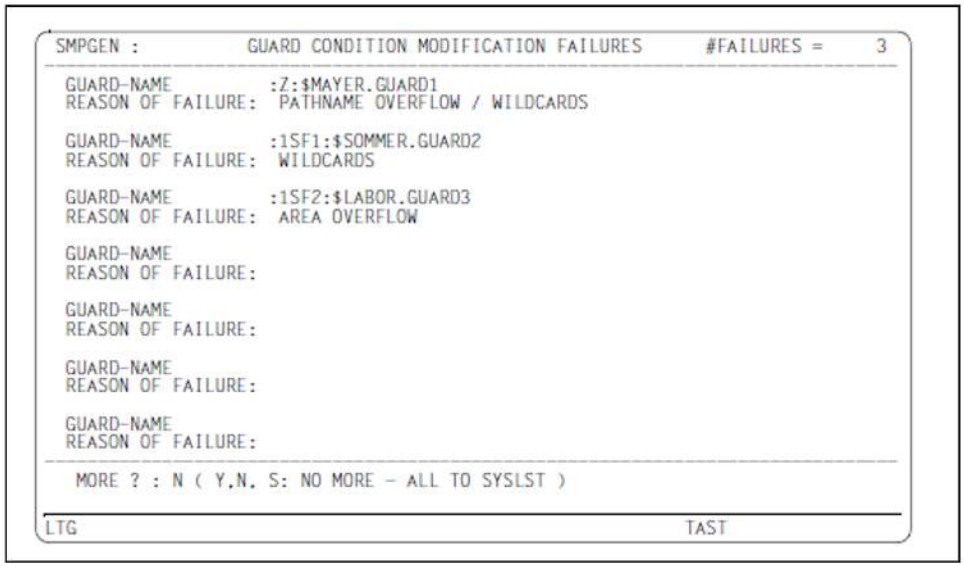
MASK 10: JV-NAME LENGTH ERRORS

Just like mask 9, except that the job variable names are listed here.

MASK 11: GUARD CONDITION MODIFICATION FAILURES

This mask is generated when guard condition errors are encountered, i.e. when program names in guard conditions could not be matched to the new pubset ID. Possible causes:

- the program name contains wildcards
- the program name would be longer than 54 characters
- the guard condition section would be too long (= area overflow)



HEADER

#FAILURES

Number of guard condition failures

DATA

GUARD-NAME

Path name of guard affected

REASON OF FAILURE

Cause or list of causes: WILDCARDS or PATHNAME OVERFLOW or AREA OVERFLOW or any combination of these three causes.

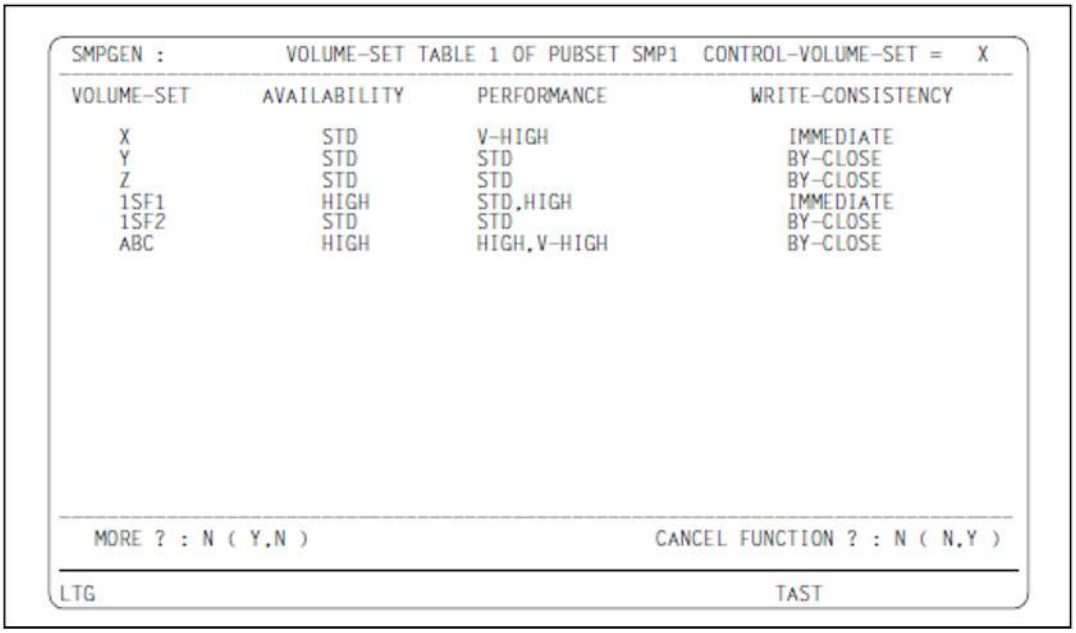
FOOTER

MORE

This field is preset to “Y” if more guards condition failures have occurred than are shown on the current and previous screen pages.
Overwriting this field with “N” prevents output of further warnings of this type at the terminal.
Overwriting this field with “S” means that no further warnings of this type are output to the terminal but all warnings of this type are output to SYSLST, independent of the TEXT-OUTPUT operand. Other outputs are continued as normal.

MASK 12: VOLUME-SET TABLE 1

This mask is generated for the CREATE-/MODIFY-SYSTEM-MANAGED-PUBSET statement if the call comes from an interactive task. This table is output to SYSOUT independent of the OUTPUT operand so that the inputs can be checked once again before the statement is executed. The footer can be used to abort the function.



HEADER

PUBSET

Catalog ID of the SM pubset to be created or extended

CONTROL-VOLUME-SET

The specified pubset ID for the operand of the same name when creating an SM pubset or the volume set ID of the control volume set when extending an SM pubset.

*NONE is shown as *NON

DATA

VOLUME-SET

ID of the volume set to be set up (previously SF pubset ID)

AVAILABILITY

Operand value of the ADD-VOLUME-SET suboperand of the same name

PERFORMANCE

Operand value of the ADD-VOLUME-SET suboperand of the same name.
Format: STD / HIGH / STD, HIGH / V-HIGH / HIGH, V-HIGH / STD, HIGH, V-HIGH

WRITE-CONSISTENCY

Operand value of the WRITE-CONSISTENCY suboperand of the ADD-VOLUME-SET operand

FOOTER

MORE

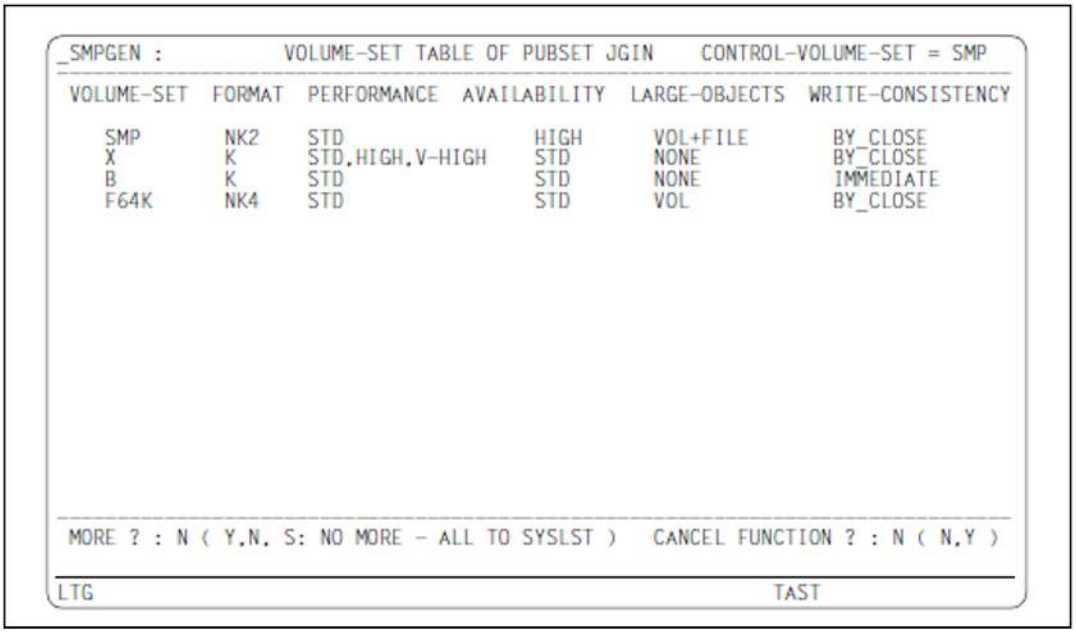
This field is preset to “Y” if more volume sets belong or are added to the SM pubset that is to be created or extended than are shown on the current and previous screen pages. Overwriting this field with “N” prevents output of further volume sets at the terminal. Other outputs are continued as normal.

CANCEL FUNCTION

This field is preset to “N”. If overwritten with another value, the function is aborted.

MASK 13: VOLUME-SET TABLE

This mask is generated for the CREATE-/MODIFY-SYSTEM-MANAGED-PUBSET statement - if not just a check function is being performed.



HEADER

CONTROL-VOLUME-SET

The specified pubset ID for the operand of the same name when creating an SM pubset or the volume set ID of the control volume set when extending an SM pubset.
*NONE is shown as *NON

PUBSET

Catalog ID of the SM pubset to be created or extended

DATA

VOLUME-SET

ID of the volume set to be created (=previously SF pubset ID)

FORMAT

Disk format determined: K / NK2 / NK4

AVAILABILITY

Operand value of the ADD-VOLUME-SET suboperand of the same name

LARGE-OBJECTS

Indicates whether volumes or files that are larger than 32 Gbytes are permitted:

VOL+FILE:	Volumes and files that are larger than 32 Gbytes are permitted
VOL:	Only volumes that are larger than 32 Gbytes are permitted
NONE:	Neither volumes nor files that are larger than 32 Gbytes are permitted

PERFORMANCE

Operand value of the ADD-VOLUME-SET suboperand of the same name.
Format: STD / HIGH / STD, HIGH / V-HIGH / HIGH, V-HIGH / STD, HIGH, V-HIGH

WRITE-CONSISTENCY

Operand value of the WRITE-CONSISTENCY suboperand of the ADD-VOLUME-SET operand

FOOTER

MORE

This field is preset to “Y” if more volume sets belong to the SM pubset that is to be created or extended than are shown on the current and previous screen pages.Overwriting this field with “N” prevents output of further volume sets at the terminal.Overwriting this field with “S” means that no further volume sets are output to the terminal but the complete volume set list is output to SYSLST regardless of what is specified for the OUTPUT operand. Other outputs are continued as normal.

CANCEL FUNCTION

This field is preset to “N”. If overwritten with another value, the function is aborted.

14.10 SMPGEN messages

The messages of the SMPGEN utility routine have the message class SGU or SPG.SMPGEN outputs no guaranteed messages.

See also the section “Messages and their meaning” (Notational conventions).

14.11 Attributes of SM pubsets and volume sets

The important attributes and behavior of SM pubsets and volume sets that are required in order to gain an understanding of SMPGEN are described below. The concept of
SM pubsets is discussed in detail in the “System-Managed Storage” manual [8 (Related publications)] or in the “Introduction to System Administration” [5 (Related publications)].

Pubset attributes

SM and SF pubsets have the following attributes in common:

- they are generally put into and taken out of operation in their entirety
- they form a name space for files, job variables and guards
- the storage location of a file within a pubset is determined by the system (except for “physical” allocation, for which the user requires special rights)
- they can be created as “large pubsets” or be converted to these. Large pubsets support files and volumes which are over 32 GB.
- there are user-related storage, file and job-variable quotas (limits) valid throughout the pubset
- there are user-related rights valid throughout the pubset (right to high-performance file processing, PUBLIC-SPACE-EXCESS, right to physical allocation)

However, unlike SF pubsets, SM pubsets are not unstructured (except for the distinction between Pubres and other disks. Within an SM pubset, there are usually several sets of associated disks with similar VSNs; these are referred to as the volume set.
An SM pubset can consist of a maximum of 255 volume sets.

If one of these volume sets fails, it can be removed from the SM pubset configuration and the SM pubset operated without it. The files that were present in this volume set can be retrieved selectively from existing backups.
Exception: If the control volume set fails, the SM pubset can no longer be operated.

Volume set attributes

Some attributes that apply to the whole of an SF pubset apply in an SM pubset only to the individual volume sets. Such attributes include the following:

- the volume set identifier forms part of the VSN of the associated disks, completely analogous to the SF pubset ID
- the allocation unit is the same on all disks
- the disk format (K, NK2, NK4) matches
- the DRV attribute corresponds as generation attribute
- the DRV/SRV mode is not set for single public disks but for an entire volume set
- the cache configuration is volume-set-related
- all extents of a file lie within a volume set

On account of the division into several volume sets, an SM pubset, and so too a file name space, can support serval different services and may therefore be used as a suitable mass storage option for different files.

Since the volume sets for storage on a particular file are not equally suitable, they are assigned logical volume set attributes:
AVAILABILITY, PERFORMANCE, WRITE-CONSISTENCY. These are used by the system along with the logical file attributes to determine the most suitable storage location for a file.

A volume set of the SM pubset, the control volume set, is used to store the most important metadata (e.g. volume set configuration, user entries, file catalog references, protection profiles). It must therefore not be removed from the SM pubset in the event of an error. If enhanced security is required, it is useful to used DRV disk pairs or a disk pair operated with DUALCOPY for this volume set, or to make a DRV (SF) pubset the control volume set.

The volume sets that form an SM pubset are distinguished according to their use as: USAGE=*STD, USAGE=*WORK and USAGE=*HSMS-CONTROLLED volume sets. USAGE=*STD volume sets are used for normal file storage, USAGE=*WORK volume sets for storing work files, and USAGE=*HSMS-CONTROLLED volume sets are used by HSMS in the background in SM pubsets supported in HSMS.

SMPGEN supports USAGE=*STD and USAGE=*HSMS-CONTROLLED volume sets. Volume sets with USAGE=*WORK must be added using reconfiguration commands.

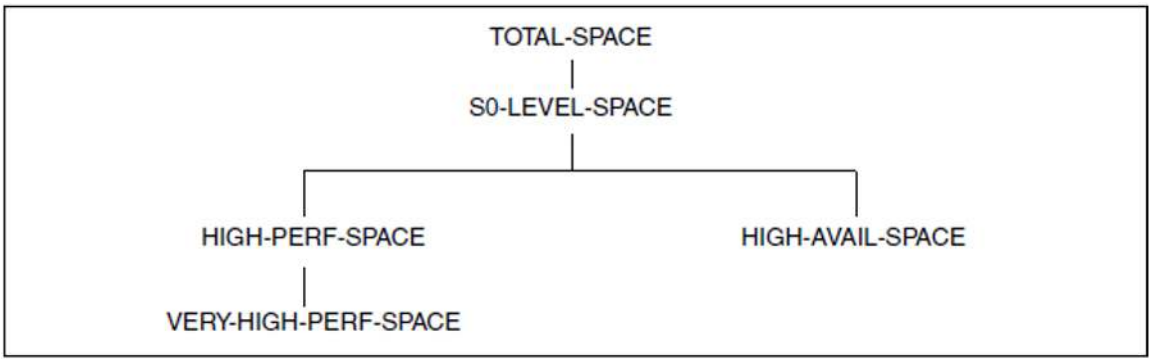
Storage space quotas

The quotas described below refer to a user ID and an SM pubset: they are not broken down over individual volume sets.
The storage space quotas on an SM pubset are more complex than on an SF pubset. The following three quota structures coexist:

- Quota for permanent files
  (these are all files that are not work files and are not temporary files)
- Quota for temporary files
- Quota for work files

“Quota structure” means that there is one storage space quota for these three types of files which defines the entire storage space that can be allocated by these files. In addition, there are so-called subquotas which can be used to further restrict the allocation of especially valuable resources.

For permanent files, the quota structure is as follows:

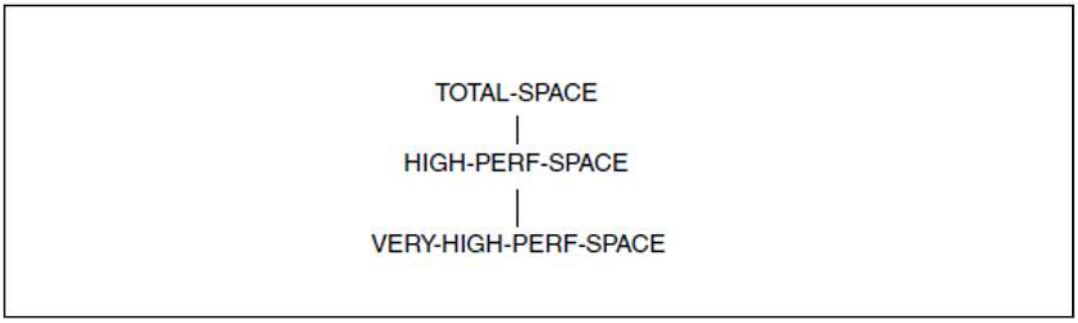


where the quota names have the following meaning:

- TOTAL-SPACE(PERM)
  is the space for all permanent files on the pubset, including those on background levels. If the file lies on a background level, what is allocated is not the real storage space; but the space that it would take up on the S0 level for a recall.

- S0-LEVEL-SPACE(PERM)
  is the space which permanent files take up on the S0 level of the pubset. This corresponds to the quota PUBLIC-SPACE-LIMIT on SF pubsets.
- HIGH-PERF-SPACE(PERM)
  is the space which permanent files with an enhanced-performance attribute (PERFORMANCE=*HIGH or *VERY-HIGH) take up on the S0 level of the pubset.
- VERY-HIGH-PERF-SPACE(PERM)
  is the space which permanent files with the attribute PERFORMANCE=*VERY-HIGH take up on the S0 level of the pubset.
- HIGH-AVAILABILITY-SPACE(PERM)
  is the space which permanent files with the high-availability attribute (AVAILABILITY=*HIGH) take up on the SO level of the pubset.

For temporary files and work files, the quota structure is as follows:



The significance of quotas corresponds to that of permanent files.

The suffix <quota name>(TEMP) or <quota name>(WORK) indicates that the quota applies to temporary files or work files respectively.
The quota for S0-LEVEL-SPACE and HIGH-AVAIL-SPACE is not necessary since only permanent files can be migrated or assigned the attribute AVAILABILITY=*HIGH.
The quota TOTAL-SPACE(TEMP) corresponds to the quota TEMP-SPACE-LIMIT on
SF pubsets.

The storage space allocation counter in the user entries corresponds to the quota structure.

The quota structure is more simple on the group level:
Only the values TEMP-SPACE-LIMIT and PUBLIC-SPACE-LIMIT familiar from SM pubsets, as well as the new limit WORK-SPACE-LIMIT are present.
(The above-mentioned limits form restrictions for the group administrator in how he or she can assign quotas to a member of the group; they should not be understood as quotas for the set of all group members).

15 SPCCNTRL Checking and managing storage allocations on disks

Version: SPCCNTRL V21.0A

The SPCCNTRL routine (SPaCe CoNTRoL) serves to control storage allocations on disk.

The routine is available to non-privileged users as well as to system administration. Although a number of its functions are reserved for system administration (see the descriptions of the individual statements).

SPCCNTRL supports both SRV disks (Single Recording Volumes) and DRV disks (Dual Recording Volumes).

In some cases alternate functions can be used instead of the SPCCNTRL program functions:

- `/SHOW-FILE-LOCK` provides information on file locks.
- For diagnostic purposes, particular catalog entries can be output to a file using `/LIST-CATALOG-ENTRY`.
- `/SHOW-PUBSET-CATALOG-ALLOCATION` provides information on catalog files and their occupancy level.
- File catalogs are automatically expanded in ongoing operation.

15.1 Operating instructions

If the SYSMES.SPCCNTRL.<ver> message file has not yet been assigned, it can be assigned later using the SPCCNTRL statement MODIFY.

The program is started with `/START-SPCCNTRL.`

START-SPCCNTRL	Alias: **SPCCNTRL**
VERSION = ***STD** / <product-version>	
,MONJV = ***NONE** / <filename 1..54 without-gen-vers>	
,CPU-LIMIT = ***JOB-RE**ST / <integer 1..32767 *seconds*>	

It can also be called as follows:

```
/START-EXECUTABLE-PROGRAM FROM-FILE=$SPCCNTRL
```

Command statements of the routine can be entered once the program is running.

15.2 Statements

- Overview of SPCCNTRL statements
- Description of the statements
  - BKPT - Interrupt SPCCNTRL routine
  - CHECK - Perform allocation checks
  - DISPLAY - Direct output to SYSOUT
  - END - Terminate SPCCNTRL
  - HELP - Describe specified statement or list all SPCCNTRL statements
  - LIST - Direct output to SYSLST
  - MODIFY - Modify default values
  - PURGE - Remove dead space and delete catalog entries
  - TRACE - Identify and output blocks and entries in system catalog and VTOC area of private disk

15.2.1 Overview of SPCCNTRL statements

Statement	Function
BKPT	Interrupt the SPCCNTRL routine.
CHECK	Perform allocation checks for a public disk and checks for errored entries in the system catalog or VTOC area of a private disk.
DISPLAY	Direct output to SYSOUT, i.e. display on the screen in interactive mode.
END	Terminate the SPCCNTRL routine.
HELP	Provide supplementary information on one specific statement or output an overview of all SPCCNTRL statements.
LIST	Direct output to SYSLST.
MODIFY	Modify the SPCCNTRL default values.
PURGE	Remove “dead” space on public disks and delete catalog entries from the system catalog and from the VTOC area of a private disk.
TRACE	Locate and output blocks and entries in the system catalog and VTOC area of a private disk, or output any block of a disk.

15.2.2 Description of the statements

In the functions listed below, “filename” must be specified as a fully qualified file name (unless specified otherwise).
In certain cases it is also possible to specify a coded file ID in the form X'hhhhhhhh'.
If no entry is made for “type”, it is assumed that the specified disk is available online.

SPCCNTRL interprets every input that is not a SPCCNTRL statement as a BS2000 command. In program mode system commands can also be entered and processed.

15.2.2.1 BKPT - Interrupt SPCCNTRL routine

The BKPT statement interrupts the program (in interactive mode) or results in a wait state (batch mode).

Format

BKPT

This statement has no operands.

- BKPT can be replaced by a slash in column 1 only.

  If another system command or a SPCCNTRL statement is entered after the slash, the command or statement is executed before the program is interrupted.

  The BKPT statement and the slash should only be used in interactive mode. In batch mode, they initiate a wait state that only ends when a message is sent to the batch job using /INFORM-PROGRAM. Receiving this message corresponds to the system command EOF, which serves to terminate the reading of data from SYSDTA. This also terminates SPCCNTRL. A return to program mode is not possible.

15.2.2.2 CHECK - Perform allocation checks

The following operations can be performed by means of the CHECK statement:

- check the allocation of a public disk
- check for errored entries in the system catalog
- check the contents of the VTOC area (F1 label) of a private disk for errored entries

A message is output for each error found, stating the errored page, the user ID and the file name. Then the check is continued.

This statement is reserved for system administration.

Both imported and non-imported catalogs are supported.

Format

CHECK / C
ALLOCATION,vsn **C**ATALOG [, { CATID = { cat-id [, { VSID = vsid / SEL= { ALL / FILE / PRIV / MIG / JV } }] / vsn[,type] / VSID = vsid }] [,ENTRY = filename / (filename,JV)] **V**TOC,vsn[,type] [,ENTRY = filename]

Operands

A[LLOCATION]
Subjects the allocation of a public volume to a general check, i.e. as to whether there are any “dead” areas or “doublers”.

An overview of the disk allocation is output. The check is aborted if cases of double allocation are detected. Such cases can be localized by means of LIST ALLOCATION.
Specification of a catalog ID is not mandatory.

vsn
Specifies the public disk that is to be checked.

C[ATALOG]
Checks the system catalog for errored entries. Error codes are assigned; these can be queried by means of HELP.

CATID = catid
Specifies

1. the TSOSCAT of an imported SF pubset.
2. the entire catalog of an SM pubset.
   If no additional parameter is specified, the information is output for all subcatalogs of the SM pubset.

VSID = vsid
Specifying a VSID restricts the output to the file catalog of the selected volume set (permissible only for SM pubsets).

SEL = ALL / FILE / PRIV / MIG / JV
Additional selection (only for SM pubsets).

SEL = ALL
Processes all subcatalogs of the selected SM pubset (analogous to no additional parameter). In the case of special catalogs, all catalog files are processed.

SEL = FILE
Selects all file catalogs of the SM pubset.

SEL = PRIV
Processes the special catalog for private disk and tape file entries. Only the first file of the special catalog (TSOSCAT.#P00) is processed.

SEL = MIG
Selects the special catalog for migrated files or files that have not been assigned space. Only the first file of the special catalog (TSOSCAT.#M00) is processed.

SEL = JV
Permits the output of information about the special catalog for a job variable. Only the first file of the special catalog (TSOSCAT.#J00) is processed.

vsn
Specifies a Pubres that contains either the file catalog of a volume set or the TSOSCAT of an SF pubset.

type
Specifies the device type of the disk defined by <vsn>.

VSID = vsid
Specifies the file catalog of a volume set.
This parameter is permissible only when processing an SM pubset.

ENTRY = filename
Specifies a fully qualified file name limiting the check to a single catalog entry.

ENTRY = (filename,JV)
The check is to be restricted to the job variable identified by “filename”. The “filename” entry can also be made in the form of a coded file ID.

V[TOC]
Checks the contents of the F1 label of a private disk for errored entries. This function corresponds to the CHECK CATALOG function.

vsn[,type]
Specifies the private disk whose VTOC area is to be checked. If the disk is not available (online), the disk type “type” must also be specified.

ENTRY = filename
Restricts the check to a specific file entry with the file name “filename”. A coded file ID can also be specified.

Examples

1. Output of information to public disk with the following statement:

```
*check allocation,1SBZ.0
 ASSIGNMENT SUMMARY FOR VOLUME 1SBZ.0
 -------------------------------------
 NUMBER OF FILE ENTRIES:             902
 NUMBER OF EXTENTS:                 1369

 PAM PAGES ASSIGNED TO FILES:     216486
 PAM PAGES NOT ASSIGNED:          115185
 PAM PAGES FREE:                       0
*
```

2. Output of information on catalog ID:

```
*CHECK CATALOG,CATID=1SBZ
%  SPC0060 NO ERROR IN THE CATALOG FILE 1SBZ:$TSOS.TSOSCAT
*
```

or

```
*CHECK CATALOG,CATID=A
ERROR SUMMARY FOR CATALOG :A:$TSOSCAT

ERROR    |    BLOCK  |  BYTE |   USER   | FILENAME
-------------------------------------------------
SPC1506  |     1635  |   584 | TEXACO   | DATOUT
SPC1503  |     1800  |   403 | ULTIMO   | PROC03
```

`/HELP-MSG-INFORMATION SPC1506` provides information on the cause of the error:

```
% SPC1506 VSN IN VOLUME TABLE ENTRY NOT PRINTABLE
```

15.2.2.3 DISPLAY - Direct output to SYSOUT

The DISPLAY statement outputs the information requested in the operands to SYSOUT. In interactive mode, the information is displayed on the terminal.

Format

DISPLAY / D

SPACE

, { **P**UBLIC [,CATID = { cat-id / vsn[,type }] /

PRIVATE[,vsn[,type]] /

EAM [, { TSN = { tsn / (ALL) } / USER = userid / CATID = cat-id }] /

CATALOG [, { CATID = cat-id [, { VSID = vsid / SEL = { ALL / FILE / PRIV / MIG / JV } }] /

vsn[,typ] /

VSID = vsid }] }

CATALOG

,ENTRY = filename / (filename,JV)

[,CATID = catid]

[,HEX = **N**O / **Y**ES]

VTOC

,VSN[,type],ENTRY = filename

[,HEX = **N**O / **Y**ES]

Operands

DISPLAY SPACE
This operand is reserved for system administration.

PUBLIC
Outputs the total amount of free and occupied memory space on all public imported volumes of the home pubset. SM pubsets are not supported. `/SHOW-PUBSET-SPACE-ALLOCATION` provides on the storage space allocation of SM pubsets (see the “Commands” manual [1 (Related publications)].

In addition to the VSN and the device type, the number of free pages and the number of file on the specified disk are also output. If the information on more than one disk is to be supplied, SPCCNTRL lists the values for the individual disks and then outputs the total. If neither CATID nor a VSN are specified, the function is executed for all disks on the home pubset.

vsn
Specifies one specific public disk, for which information on the space allocation is requested.

type
Specifies the disk type

CATID = catid
Specifies the catalog ID of an imported SF pubset. Information is output about the space allocation of disks that are entered in a specific catalog. If this operand is not specified, the home catalog is always assumed. In the case of a non-homogeneous network, information are only output for the MASTER.

PRIVATE
Outputs the total amount of free and occupied space on all private disks (including labels).

vsn
Identifies one specific private disk, on which information is requested.

type
Specifies the disk type (e.g. D3435). This entry is required if the private disk is not online.

EAM
Outputs information on the system file SYSEAM. If no entry is made for “tsn” or ALL, an overview of the overall allocation for SYSEAM is output. If “tsn” is specified, the information is restricted to the allocation on the specified TSN.

TSN = tsn / (ALL)
The information on the system file SYSEAM is to refer solely to the specified TSN (“tsn”) or to all TSNs (ALL).

USER = userid
The information on the system file SYSEAM is to refer to the specified user ID.

CATID = catid
This is the catalog ID of the requested catalog. The home catalog is assumed by default if the operand is omitted.

CATALOG
Provides information on the overall allocation of one or more (for SM pubsets) catalog file(s).

The number of user IDs, the size of the catalog file, the number of pages occupied (according to the counter and BITMAP) and an error indicator are output. The information provided is internal management data of the Catalog Management System (CMS).

Both imported and non-imported catalogs are supported.

¡ For files on LARGE-OBJECTS pubsets this function is reserved for system administration. All other users are rejected, and message `DMS0576` is output.

CATID = catid
Specifies

1. the TSOSCAT of an imported SF pubset.
2. the entire catalog of an SM pubset.If no additional parameter is specified, the information is output for all subcatalogs of the SM pubset.

VSID = vsid
Specifying a VSID restricts the output to the file catalog of the selected volume set (permissible only for SM pubsets).

SEL = ALL / FILE / PRIV / MIG / JV
Additional selection (only for SM pubsets). In the case of special catalogs, all catalog files are processed.

SEL = ALL
Processes all subcatalogs of the selected SM pubset (analogous to no additional parameter).

SEL = FILE
Selects all file catalogs of the SM pubset.

SEL = PRIV
Processes the special catalog for private disk and tape file entries. Only the first file of the special catalog (TSOSCAT.#P00) is processed.

SEL = MIG
Selects the special catalog for migrated files or files that have not been assigned space. Only the first file of the special catalog (TSOSCAT.#M00) is processed.

SEL = JV
Permits the output of information about the special catalog for a job variable. Only the first file of the special catalog (TSOSCAT.#J00) is processed.

vsn
Specifies a disk that contains either the file catalog of a volume set or the TSOSCAT of an SF pubset.

type
Specifies the device type of the disk defined by <vsn>.

VSID = vsid
Specifies the file catalog of a volume set. This parameter is permissible only when processing an SM pubset.

DISPLAY CATALOG
This operand is available to all users.

> If the suboperand ENTRY specifies a file > 32 GB or a file on a volume > 32 GB and the caller does not have the TSOS user ID, invalid values are supplied in the output which can be recognized as zero (for the file in Total Space Allocated and for the extents in Low, High, Pages Allocated).

It outputs the information not supplied by `/SHOW-FILE-ATTRIBUTES` from the catalog entry of a file or a job variable to SYSOUT.

The output information includes the size of the file or job variable, the size of the catalog entry, CFID or JVID, primary block number, any locks and the number of extents.

The primary block number when the statement is entered: TRACE CATALOG, block (primary block number), byte. The name of the corresponding catalog/subcatalog is also displayed.

Non-privileged users can only address files or job variable under their own user ID. DISPLAY CATALOG supports only files of imported catalogs.

ENTRY = filename
Requests output of the catalog entry for the file specified by “filename”. “filename” can also be specified as a coded file ID in the form X'hhhhhhhh'.

ENTRY = (filename,JV)
Requests output of the catalog for the job variable specified by “filename”. “filename” can also be specified as JV ID or as a coded file ID in the form X'hhhhhhhh'.

CATID = catid
Specifies the catalog ID of the requested catalog (SF pubset or SM pubset). The default value is the default catid of the caller’s user ID.

HEX = NO / YES
Specifies whether the entire catalog entry is to be output in hexadecimal form. NO is the default value.

DISPLAY VTOC
This operand is available to all users.

It outputs information contained in the F1 label on a private volume. The same information is supplied as for DISPLAY CATALOG, with the difference that the VTOC area of the private disk serves as the source.

vsn[,type]
Specifies the volume from whose VTOC area information is requested. If the disk is not online, the disk type (“type”) must also be specified.

ENTRY = filename
Specifies the file entry in the F1 label on which information is to be output. “filename” can also be specified as a coded file ID in the form X'hhhhhhhh'.

HEX = NO / YES
Specifies whether the entire entry is to be output in hexadecimal form. NO is the default value.

Examples

1. Output of DISPLAY SPACE,PUBLIC

```
DISPLAY S,PUBLIC
VOLUME I DEVICE I PAGES FREE I PAGES USED I FILES
-------I--------I------------I------------I--------
2SBZ.0 I  D3435 I    157758  I    450948  I  1660
2SBZ.1 I  D3435 I    152823  I    455886  I  1655
2SBZ.2 I  D3435 I      3408  I    328266  I  1656
-----------------------------------------------
TOTAL           I    313989  I   1235100  I  4971
```

2. Output of DISPLAY SPACE,EAM

```
D S,EAM
SUMMARY FOR :K:$TSOS.SYSEAM:

SYSEAM SPACE ASSIGNED:              348 PAGES
SYSEAM SPACE ALLOCATED:           38712 PAGES
SYSEAM CATALOG ENTRY SIZE:          468 BYTES
```

3. Output of DISPLAY SPACE,EAM,TSN=(ALL)

```
D S,EAM,TSN=(ALL)
CATID I TSN  I USER     I TYPE I FILES I PAGES USED I ALLOCATED
------I------I----------I------I-------I------------I----------
 K    I 5015 I TSOS     I   3  I    2  I          6  I       6
 K    I 5060 I A2A3A4A5 I   3  I    5  I          6  I       6
 K    I 5068 I RZDATEX  I   3  I    1  I         32  I      33
 K    I 5158 I TERM     I   3  I    1  I          0  I       0
 K    I 5167 I DATA     I   3  I   40  I          1  I       3
 K    I 5208 I BASIS    I   3  I    3  I          3  I       6
 K    I 5232 I ARCHIV   I   3  I    3  I          0  I       0
 K    I 5233 I CONS     I   3  I    2  I         62  I      63
 K    I 5239 I DELTA    I   3  I    6  I         23  I      24
 K    I 5328 I F1F2F3F4 I   3  I    1  I          5  I       6
 K    I 5346 I BERGER   I   3  I   17  I          4  I       6
 K    I 5352 I FIFO     I   3  I    5  I         45  I      45
```

4. Output of DISPLAY SPACE,CATALOG

```
D S,C
 TSOSCAT/HOME
-------------------------------------------------------------------------------
 USERID' S IN JOINFILE               69
 EFFECTIVE FILE SIZE               2049
 PAGES USED (4K)    (COUNTER)       279
 PAGES USED (4K)    (BIT TABLE)     279
 PREASSIGNED (ERROR IF NOT 0)         0
```

5. Output of DISPLAY CATALOG, ENTRY=TEMP.12

```
D C,ENTRY=$TSOS.TEMP.12
PUBLIC DISK FILE:   :2BV:$TSOS.TEMP.12
------------------------------------------------------------------------
TOTAL SPACE ALLOCATED:                3 PAGES
CODED FILE IDENTIFICATION:     8D32C3B4
*   FLT LOCK ENTRY:                00000000 00000000 00000001 00000000
*                                  00000000 00000000 00000000 00000000
*                                  00000000 00000000 00000000 00000000
*                                  00000000 00000000 00000000 00000000
*                                  00000000 00000000 00000000 00000000
*                                  00000000 00000000 00000000 00000000
PRIMARY BLOCK NUMBER (BLK=4KB):      16
CURRENT BLOCK NUMBER:                16 CATALOG ENTRY AT BYTE:     3665
CATALOG ENTRY SIZE:                 121 BYTES
# OF EXTENT ENTRIES:                  1
EXTENT I  VOLUME  I       LOW       I      HIGH      I   PAGES ALLOCATED
-------I----------I-----------------I----------------I------------------
   1   I  2BV.01  I        6679     I        6681    I           3
```

* This output is linked to the TSOS privilege.

Meaning of the output fields for FLT LOCK ENTRY:

linewise from left to right, value specified in hexadecimal format

Number of openers with OPEN=INPUT,SHARUPD=WEAK
Number of openers with OPEN=INPUT,SHARUPD=NO
Number of openers with OPEN=INPUT,SHARUPD=YES
Number of openers with DIV, MODE=*INPUT,SHARUPD=YES

Number of openers with DIV, MODE=*INPUT,SHARUPD=WEAK
Number of input reservation blocks for file transfer
Number of input reservation blocks for SPOOL (print locks)
Number of openers with OPEN=OUTPUT,SHARUPD=YES

Number of openers with DIV, MODE=*OUTPUT,SHARUPD=YES
Number of openers with DIV, MODE=*OUTPUT,SHARUPD=WEAK
Number of openers with OPEN=OUTPUT,SHARUPD=NO
Number of write exclusive reservations

Number of openers with OPEN=OUTPUT,SHARUPD=WEAK
Number of secure locks
Number of output reservation locks for file transfer
Number of HSMS secure locks

Number of concurrent copy locks
Number of concurrent copy locks for the setup of the concurrent copy session
Number of exclusive path locks for files on private disks
Number of 0 locks

Reserved for internal use
Reserved for internal use
Number of host environment locksNumber of XCS environment locks

File locks can also be displayed using `/SHOW-FILE-LOCKS`.

6. Output of DISPLAY VTOC, RZWORK, ENTRY=SPC.TEST.PR

```
D V,RZWORK,ENTRY=SPC.TEST.PR
PRIVATE DISK FILE: :HOME:$TSOS.SPC.TEST.PR
TOTAL SPACE ALLOCATED:              3 PAGES
CODED FILE IDENTIFICATION:    92032342
FLT LOCK ENTRY:                   NONE
PAM LOCK ENTRY:                   NONE
VTOC LOCK ENTRY:                  NONE
VTOC SHARER ENTRIES:              NONE
PRIMARY BLOCK NUMBER:               2
CURRENT BLOCK NUMBER:            2426        CATALOG ENTRY AT BYTE:
VTOC/F1 ENTRY ON VOLUME:        RZWORK       ON-LINE
VTOC/F1 BLOCK POINTER:              10
VTOC/F1 ENTRY IN BLOCK:             10       CATALOG ENTRY AT BYTE:   1874
CATALOG ENTRY SIZE:                125 BYTES
# OF EXTENT ENTRIES:                 1
EXTENT I  VOLUME  I       LOW       I      HIGH      I   PAGES ALLOCATED
-------I----------I-----------------I----------------I-------------------
   1   I  RZWORK  I        2878     I        2880    I           3
```

15.2.2.4 END - Terminate SPCCNTRL

The END statement terminates SPCCNTRL.

Format

END

This statement has no operands.

> If E is entered followed by an operand, SPCCNTRL assumes `/ENTER-JOB` is meant and executes this command.

15.2.2.5 HELP - Describe specified statement or list all SPCCNTRL statements

The HELP statement allows you to obtain a description of a specific statement or an overview of all the SPCCNTRL statements.

Format

HELP
[stmt[,mode]]

Operands

stmt

Specifies the name of the SPCCNTRL statement. A description of this statement is to be output. Possible values are:

```
HELP
BKPT
END
MODIFY
DISPLAY
LIST
CHECK
PURGE
TRACE
```

If “stmt” is not specified, an overview of all statements is output.

mode
Specifies one of the following keywords of the statement designated by “stmt”:

```
A[LLOCATION]
S[PACE]
C[ATALOG]
V[TOC]
```

A description of the specified function is output. If no operand is specified, an overview of the available statement function modes is output.

If the HELP statement is specified in non-abbreviated form and an invalid operand is specified for “stmt” instead of an SPCCNTRL statement, `/HELP-MSG-INFORMATION` is executed.

15.2.2.6 LIST - Direct output to SYSLST

The LIST statement enables information to be written to SYSLST. This makes it useful for redirecting information which is too extensive to be output on the terminal screen.

This statement is reserved for the TSOS user ID.

Format

LIST / L
ALLOCATION,vsn[,type] [USER = userid] **C**ATALOG [, { CATID = cat-id [,{ VSID = vsid / SEL = { ALL / FILE / PRIV / MIG / JV } }] / vsn[,type] / VSID = vsid }] [,USER = userid] **V**TOC,vsn[,type] [,ENTRY = filename]

Operands

ALLOCATION
Outputs general allocation information for a volume.

The list sorts the allocation information according to extent. The start of each extent being specified in the form “cchhr”.

Information can only be output on volumes that are identified by their volume serial number. When specifying operands it is not necessary to distinguish between public and private volumes.

The following comments may occur in the output list:

RESERVED	Space allocated without file names. It is possible to enter a file name if you wish.
FREE SPACE	Free area on public disks.
OVERLAY!	Space allocated twice.
UL-SVL-F5-F1	Space reserved by SVL, F5 and F1 (private disks).
NO VTOC REF.	Free area on disks.
F1 OTH. DISC	Area described by F1 of another private disk.
NOT ASSIGNED	Area not assigned.
NO REFERENCE	There may be outstanding allocator requirements, or an error has been indicated by CMS.
SPACE PROBLEM	Bad area.

vsn[,type]
Specifies the disk for which allocation information is to be listed. If a private disk is not online, the disk type “type” must also be specified. In the case of public disks, both free and “dead” space are listed.

“Dead” space is an area which the entry in the F5 label identifies as being allocated but for which there is no file in the system catalog.

USER = userid
Specifying a user ID limits the output. In this case, no information on free or dead space is output, even for public volumes.

CATALOG
Provides a list of allocated blocks in a catalog file.

Both imported and non-imported catalogs are supported.

For each catalog block, the user ID, the block counter, the primary block number (hexadecimal and decimal), the number of entries in the block and the number of unused bytes in the block are output. Then the statement has been executed properly, the message 'LIST PROCESSING COMPLETED' is output.

CATID = catid
Specifies

1. the TSOSCAT of an imported SF pubset.
2. the entire catalog of an SM pubset.
   If no additional parameter is specified, the information is output for all subcatalogs of the SM pubset.

   VSID = vsid
   Specifying a VSID restricts the output to the file catalog of the selected volume set (permissible only for SM pubsets).

   SEL = ALL / FILE / PRIV / MIG / JV
   Additional selection (only for SM pubsets).

   SEL = ALL
   Processes all subcatalogs of the selected SM pubset (analogous to no additional parameter). In the case of special catalogs, all catalog files are processed.

   SEL = FILE
   Selects all file catalogs of the SM pubset.

   SEL = PRIV
   Processes the special catalog for private disk and tape file entries. Only the first file of the special catalog (TSOSCAT.#P00) is processed.

   SEL = MIG
   Selects the special catalog for migrated files or files that have not been assigned space. Only the first file of the special catalog (TSOSCAT.#M00) is processed.

   SEL = JV
   Permits the output of information about the special catalog for a job variable. Only the first file of the special catalog (TSOSCAT.#J00) is processed.

vsn
Specifies a Pubres that contains either the file catalog of a volume set or the TSOSCAT of an SF pubset.

type
Specifies the device type of the disk defined by <vsn>.

VSID = vsid
Specifies the file catalog of a volume set. This parameter is permissible only when processing an SM pubset.

USER = userid
Specifying a “userid” limits the output of information to this user ID.

VTOC
Requests information from the F1 label of a private disk.

vsn[,type]
Specifies the volume serial number of the disk from whose VTOC area information is requested. If the volume is not online, the disk type “type” must also be specified.

ENTRY = filename
Limits output to the file entry “filename”. Fully qualified file names are not prohibited, but only the specification of partially qualified file names is practical.

Examples

1. Output of LIST ALLOCATION,PAGM.0

```
S P C C N T R L    VERSION <version>             DATE: <date> TIME: <time>                  PAGE 1
COMMAND IN EXECUTION:   LIST ALLOCATION,PAGM.0
                                        EXTENTS ON VOLUME PAGM.0
  CCHHR (LOW) ||      LOW     |     HIGH     ||    ASSIGNMENT    | PAM PAGES ||   USERID    |  FILENAME
--------------||--------------|--------------||------------------|-----------||------------|-----------------
 -----------  ||           1  |          54  ||  RESERVED        |        54 ||            |
 -----------  ||          55  |        4152  ||                   |      4098 ||  TSOS        |  TSOSCAT
 -----------  ||        4153  |        4176  ||                   |        24 ||  TSOS        |  SYSSRPM
 -----------  ||        4177  |        4182  ||                   |         6 ||  TSOS        |  SYSSRPM.BACKUP
 -----------  ||        4183  |        4224  ||                   |        42 ||            |
 -----------  ||        4225  |        4272  ||                   |        48 ||  TSOS        |  SYSCAT.GUARDS
 -----------  ||        4273  |        4305  ||                   |        33 ||  TSOS        |  SYSDAT.APUB...
 -----------  ||        4306  |        4320  ||                   |        15 ||            |
 -----------  ||        4321  |        4368  ||                   |        48 ||  TSOS        | SYSCAT.GUARD...
 -----------  ||        4369  |        4416  ||                   |        48 ||            |
 -----------  ||        4417  |     1028418  ||                   |   1024002 ||  TSOS        | SYS.PAGING.P...
 -----------  ||     1028419  |     1966080  ||                   |    937662 ||            |
...
```

Notes on the ASSIGNMENT column

RESERVED	Space allocated without file names. It was not possible to enter a file name (column 7).
FREE SPACE	Free area on public disks.
OVERLAY!	Space allocated twice.
UL-SVL-F5-F1	Space reserved by SVL, F5 and F1 (private disks).
NO VTOC REF.	Free area on disks.
F1 OTH. DISC	Area described by F1 of another private disk.
NOT ASSIGNED	Area not assigned.
NO REFERENCE	There may be outstanding allocator requirements, or an error has been indicated by CMS.

SPACE PROBLEM Bad area.

2. Output of LIST CATALOG

```
S P C C N T R L    VERSION <version>             DATE: <date> TIME: <time>                  PAGE     1
COMMAND IN EXECUTION:   LIST CATALOG,CATID=2SBZ,USER=TSOS
                                        SUMMARY OF CONTENTS IN :2SBZ:$TSOS.TSOSCAT
                                                    FOR USERID TSOS
```

	FILE ENTRIES					JOB VARIABLE ENTRIES				
USERID	BLK COUNT	LBN(DEC)	LBN(HEX)	# ENTRIES	UNUSED	BLK COUNT	LBN(DEC)	LBN(HEX)	# ENTRIES	UNUSED
TSOS	1	2	0002	1	3762	1	281	0119	16	96
	2	4	0004	12	188	2	311	0137	14	181
	3	17	0011	12	236	3	321	0141	17	24
	4	18	0012	12	212	4	329	0149	16	112
	5	19	0013	12	170	5	503	01F7	14	171
	6	20	0014	12	205	6	506	01FA	15	75
	7	21	0015	12	110	7	510	01FE	13	183
	8	22	0016	12	225	8	517	0205	14	47
	9	23	0017	12	237	9	518	0206	14	47
	10	24	0018	12	196	10	528	0210	16	0
	11	25	0019	12	268	11	562	0232	12	80
	12	26	001A	12	259	12	629	0275	14	59
	13	27	001B	12	244	13	591	024F	17	165
	14	28	001C	12	144	14	592	0250	15	98
	15	29	001D	12	194	15	577	0241	13	14
	16	30	001E	12	218	16	646	0286	13	13
	17	31	001F	12	256	17	584	0248	14	27
	18	32	0020	12	256	18	645	0285	1	3722
	19	33	0021	12	205	19	648	0288	7	2172
	20	34	0022	12	183	20	654	028E	1	3730
	21	35	0023	12	196					
	22	36	0024	12	141					
	23	37	0025	12	237					
	24	38	0026	12	231					
	25	39	0027	12	177					
	26	40	0028	12	207					
	27	41	0029	12	175					
	28	42	002A	12	139					
	29	43	002B	12	63					
	30	44	002C	12	115					
	31	45	002D	12	147					
	32	46	002E	12	131					
	33	47	002F	12	154					
	34	48	0030	12	149					
	35	49	0031	12	164					
	36	50	0032	12	169					
	37	51	0033	12	128					
	38	52	0034	12	176					
	39	53	0035	12	153					
	40	54	0036	12	142					
	41	55	0037	12	153					
	42	56	0038	12	167					
	43	57	0039	12	167					
	44	58	003A	12	167					

15.2.2.7 MODIFY - Modify default values

The MODIFY statement enables SPCCNTRL default values to be modified if other values are desired. Any such modification applies only to the current program run.

This statement can also be used for the subsequent assignment of the SPCCNTRL message file or of any other message file. This is possible only under the TSOS user ID. The message file remains in the system until the end of the session, unless the assignment is changed by means of `/MODIFY-MSG-FILE-ASSIGNMENT`.

Format

MODIFY
{ LINES = nn / MSG = { **SPC**CNTRL / (FILE,filename) } }

Operands

LINES = nn
Specifies the desired number (“nn”) of lines per page for the list output (where 14 <= nn<= 127). If no value is specified, the default value is 60.

MSG = SPCCNTRL
TSOS only.
Specifies that the message file valid for SPCCNTRL is to be assigned. The expected file name is SYSMES.SPCCNTRL.version.
“version” is the version number of the SPCCNTRL routine used.

MSG = (FILE,filename)
Dynamically loads messages from the file specified here by “filename”.

15.2.2.8 PURGE - Remove dead space and delete catalog entries

The PURGE statement can be used to remove dead space from public volumes and to delete catalog entries from the system catalog. It can also be used to remove unused space in individual files.

Format

```
PURGE
```

```
SPACE
    , { PUBLIC [,{ CATID = cat-id / VSID = vsid / vsn } ]
    PRIVATE[,vsn[,type]] }
    [,USER = userid]
ALLOCATION,vsn
CATALOG
    [,CATID = catid]
    ,ENTRY = filename / (filename,JV)
```

Operands

SPACE
Releases unused space in the individual user files.
The assigned buffer sizes are taken into account, i.e. only as much space is released as will enable the file in question to be processed without errors. This function is available to all users. However, in this case only files of the user's own ID are taken into consideration.Password-protected files are not affected. In the event of an unused area of a file being the reserved area of this file, 1 allocation unit remains for the file.

PUBLIC
Releases unused space on public volumes. This function is only performed on imported pubsets.

CATID = catid
Specifies the catalog ID of an SF pubset or an SM pubset from which entries are to be deleted.
If neither VSN nor CATID is specified, the default catalog ID of the user ID specified via USER= is assumed.

VSID = vsid
Represents the ID of a volume set.

This operand is reserved for system administration.

vsn
Specifies the VSN of a particular public volume on which unused space is to be released.

PRIVATE
Releases unused space on private volumes. Non-privileged users can only process files under their own user ID and which also have an entry in the system catalog.

vsn[,type]
Specifies the disk on which space is to be released. Non-privileged users must specify both the VSN and the disk type “type”. When working under TSOS, however, it is sufficient to enter the VSN provided the disk is online.

USER = userid
Specifies the user ID whose files are to be processed. If no entry is made and SPCCNTRL is running under the TSOS user ID, the files of all users except TSOS are processed.
PURGE SPACE,PUBLIC checks all files of the default pubset of the caller ID for space that can be released if no parameter is specified under an ID that is not from TSOS.PURGE SPACE,PRIVATE processes the files of all users (including TSOS) if USER is not specified and SPCCNTRL is running under TSOS.

ALLOCATION
This operand is reserved for system administration.
It reorganizes the allocation of space on a public volume, i.e. dead space is to be removed. This does not happen in the same SPCCNTRL run, however; instead, the run has to be terminated and `/IMPORT-PUBSET` has to be entered first.
Dead space may have been created by system errors or explicitly created by means of the statement: PURGE CATALOG,ENTRY.

vsn
Specifies the volume serial number of a public volume to be cleaned up.

CATALOG
This operand is reserved for system administration.
It removes an entry from the system catalog. This is useful in the case of errored entries which can no longer be accessed.

CATID = catid
Specifies the catalog from which an entry is to be deleted. If this operand is omitted, the home catalog is assumed.

ENTRY = filename
Specifies the name of the file whose catalog entry is to be deleted.

ENTRY = (filename,JV)
Specifies the name of the job variable whose catalog entry is to be deleted.

Example

Output fom PURGE SPACE,PUBLIC

```
PURGE S,PUBLIC
% SPC0025 PURGE PROCESSING COMPLETED
NUMBER OF FILES SELECTED FOR PURGE:     1062   -------------------------- 1.
NUMBER OF ERRORS DURING DEALLOCATION:    121   -------------------------- 2.
AMOUNTED OF PAM PAGES DEALLOCATED:     20787   -------------------------- 3.
```

1. Number of files found to contain purgeable memory.
2. Number of files which could not be purged.
3. Total number of PAM pages released by the purge.

15.2.2.9 TRACE - Identify and output blocks and entries in system catalog and VTOC area of private disk

The TRACE statement enables blocks and entries in the system catalog and the VTOC area of a private disk to be identified and output to SYSOUT or SYSLST. In addition, any block of a public or private disk can also be displayed. In order to save write operations, the default values for the keyword operands remain valid until they are modified by a subsequent entry.

This statement is reserved for the TSOS user ID.

Format

TRACE
CATALOG, block[,byte] [, { CATID = cat-id [,{ VSID = vsid / SEL = { PRIV / MIG / JV} }] / vsn[,type] / VSID = vsid }] [,USER = userid] [,JV = **N**O / **Y**ES] [,OUTPUT = (BLOCK[,{ SYSOUT / SYSLST }]) / (OWNER[,SYSOUT]) / SYSOUT / SYSLST] **V**TOC ,{ block / (block,PHP) } [,byte],vsn[,type] [,OUTPUT = (BLOCK[,{ SYSOUT / SYSLST }]) / SYSOUT / SYSLST] **S**PACE,block[,byte],vsn[,type] [,OUTPUT = (BLOCK[,{ SYSOUT / SYSLST }]) / SYSOUT / SYSLST]

Operands

CATALOG
Supplies information from the system catalog. Any block, including the control blocks with the bit tables, can be displayed even if it is not assigned to a user.

Both imported and non-imported catalogs are supported.

block
Specifies the number of the block to be output. This entry can also be made in hexadecimal form (X'hhhh'). If a user ID is specified under USER=userid, the block specification is relative to the user's catalog chain; otherwise "block" is interpreted as an absolute block number.

Together with OUTPUT=(BLOCK,SYSLST), "block" can be omitted.
In this case the complete catalog is output. In the case of catalogs with an odd number of pages, the last (half) page is not output. This is not used and is therefore empty.

byte
Specifies the byte number within the block from which output is to start. Here, too, the entry can be made in hexadecimal form (X'hhhh').

CATID = catid
Specifies

1. the TSOSCAT of an imported SF pubset.
2. the entire catalog of an SM pubset. In this case, further information must be specified in the VSID or SEL operand.

   VSID = vsid
   Specifying a VSID restricts the output to the file catalog of the selected volume set (permissible only for SM pubsets; see also note below).

   SEL = PRIV / MIG / JV
   Specifies a special catalog (only for SM pubsets):

   SEL = PRIV
   Processes the special catalog for private disk and tape file entries. Only the first file of the special catalog (TSOSCAT.#P00) is processed.

   SEL = MIG
   Selects the special catalog for migrated files or files that have not been assigned space. Only the first file of the special catalog (TSOSCAT.#M00) is processed.

   SEL = JV
   Permits the output of information about the special catalog for a job variable. Only the first file of the special catalog (TSOSCAT.#J00) is processed.

> The VSID and SEL operands are prohibited for SF pubsets. In the case of SM pubsets, however, either VSID or SEL must be specified.

vsn
Specifies a Pubres that contains either the file catalog of a volume set or the TSOSCAT of an SF pubset.

type
Specifies the device type of the disk defined by <vsn>.

VSID = vsid
Specifies the file catalog of a volume set.
This parameter is permissible only when processing an SM pubset.

USER = userid
Specifies the user ID of the user from whose catalog entry blocks are to be output. If a user ID is specified for SM pubsets, it is possible only to process a subcatalog. This subcatalog must be defined by specifying a <vsid> or the parameter SEL.

JV = NO / YES
This parameter is permissible only for processing an SF pubset. It is ignored for SM pubsets.

JV = NO
Ignores job variables (default). A block from the user's file name chain is output.

JV = YES
Displays blocks from a job variable chain. This entry is practical only in conjunction with USER=userid, in all other cases it is ignored.

OUTPUT = (BLOCK,SYSOUT)
Directs output to SYSOUT (default).

OUTPUT = (BLOCK,SYSLST)
Output is to be to SYSLST.

OUTPUT = (OWNER,SYSOUT)
utputs the owner (user ID) of the desired block rather than its contents to SYSOUT.

OUTPUT = SYOUT / SYSLST
This is the same as (BLOCK,SYSOUT) or (BLOCK,SYSLST).

VTOC
Outputs information from the VTOC area of a private disk.

block
Specifies the number of the block to be output. This entry can also be made in hexadecimal form (X'hhhh'). Together with OUTPUT=(BLOCK,SYSLST), “block” can be omitted. In this case the entire VTOC area is output.

PHP
Specifies a physical block number, i.e. the block number is not relative to the beginning of the VTOC area.

byte
Specifies the byte number within the block from which output is to start. This entry can also be made in hexadecimal form (X'hhhh').

vsn[,type]
Specifies the volume serial number of the private disk whose VTOC area is to be output. If the disk is not available (online), the disk type “type” must also be specified.

OUTPUT = (BLOCK,SYSOUT)
Directs output to SYSOUT (default).

OUTPUT = (BLOCK,SYSLST)
Output is to be to SYSLST.

OUTPUT = SYOUT / SYSLST
This is the same as (BLOCK,SYSOUT) or (BLOCK,SYSLST).

SPACE
Outputs a freely selectable item of information from the specified disk.

block
Specifies the number of the block to be output. This entry can also be made in hexadecimal form (X'hhhh'). Together with OUTPUT=SYSLST, the “block” specification can be omitted. In this case the entire SVL together with the F5 label is output.

byte
Specifies the byte number within the block from which output is to start. This entry can also be made in hexadecimal form (X'hhhh').

vsn[,type]
Specifies the volume serial number of the public or private disk on which information is to be output. If the disk is not available (online), the disk type “type” must also be specified.

OUTPUT = (BLOCK,SYSOUT)
Directs output to SYSOUT (default).

OUTPUT = (BLOCK,SYSLST)
Output is to be to SYSLST.

OUTPUT = SYOUT / SYSLST
This is the same as (BLOCK,SYSOUT) or (BLOCK,SYSLST).

Example

```
(IN)     TRACE CATALOG,1,10,CATID=2SBZ
(OUT)    :2SBZ:TSOSCAT       LBN:     1  (0001)    PHP:       28 ON  VOLUME 2SBZ.0
(NL)     ------------------------------------------------------------------------
(NL)
(NL)      0009   (000):     FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF      ~~~~~~~~~~~~~~~~
(NL)      0019   (010):     FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF      ~~~~~~~~~~~~~~~~
(NL)      0029   (020):     FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF      ~~~~~~~~~~~~~~~~
(NL)      0039   (030):     FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF      ~~~~~~~~~~~~~~~~
(NL)      0049   (040):     FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF      ~~~~~~~~~~~~~~~~
(NL)      0059   (050):     FFFFFFFF 00000000 00000000 00000000      ~~~~............
(NL)      0069   (060):     00000000 00000000 00000000 00000000      ................
(NL)      0079   (070):     00000000 00000000 00000000 00000000      ................
(NL)      0089   (080):     00000000 00000000 00000000 00000000      ................
(NL)      0099   (090):     00000000 00000000 00000000 00000000      ................
(NL)      00A9   (0A0):     00000000 00000000 00000000 00000000      ................
(NL)      00B9   (0B0):     00000000 00000000 00000000 00000000      ................
(NL)      00C9   (0C0):     00000000 00000000 00000000 00000000      ................
(NL)      00D9   (0D0):     00000000 00000000 00000000 00000000      ................
(NL)      00E9   (0E0):     00000000 00000000 00000000 00000000      ................
(NL)      00F9   (0F0):     00000000 00000000 00000000 00000000      ................
(NL)      0109   (100):     00000000 00000000 00000000 00000000      ................
(NL)      0119   (110):     00000000 00000000 00000000 00000000      ................
(NL)      0129   (120):     00000000 00000000 00000000 00000000      ................
(OUT)    %  SPC0020 SCREEN OVERFLOW, CONTINUE (Y/N)?
```

16 TPCOMP2 Comparison of data on tapes

Version: TPCOMP2 V21.0A

The TPCOMP2 utility routine serves to compare data recorded on two magnetic tapes. It provides a listing of all portions of these tapes that are not identical.

Comparison of data is done on a decade basis. This allows for the printing of 5 groups of variant data in one line on a 132-character printer or 6 groups per line on a 160-character printer.

Input and output

Input to the tape compare routine consists of 2 magnetic tapes recorded in the same format. This routine requires no parameters for its standard functions. For other options, parameters are entered via SYSDTA.

Output is a printed listing of any discrepancies that exist between the tapes being compared.

16.1 Functions

In addition to the standard functions listed below, this routine provides other functions which can be selected by specifying parameters (see “Optional functions”).

Standard functions

TPCOMP2 has a preset option which provides the following standard functions for tapes:

1. Rewind both tapes to BOT at the start and end of the comparison run.
2. Display differences in both hexadecimal and EBCDIC format on a 132-character printer.
3. Terminate the comparison if a double tape mark is sensed.

   (If a double tape mark for one tape is sensed, and the other tape is not also positioned at a double tape mark, the remaining data on the second tape is printed until the double tape mark is encountered.)

Optional functions

By the use of parameters (entered via SYSDTA), the following options can be selected:

1. Positioning of tapes in a forward or reverse direction, based on a count of tape marks or blocks.
2. Terminating the comparison on the basis of a given number of tape marks or blocks.
3. Designating a 160-character printer; specifying print formats of hexadecimal, EBCDIC, or a combination of both.
4. Processing of unblocked variable-length records or blocked fixed-length records.

16.2 Starting the program run

The TPCOMP2 program is started using

```
/START-EXECUTABLE-PROGRAM $.SYSPRG.TPCOMP2.210
```

The CPU-LIMIT, TEST-OPTIONS, MONJV, RESIDENT-PAGES and VIRTUAL-PAGES operands of `/START-EXECUTABLE-PROGRAM` are also available, for example to monitor the program run (see the “Commands” manual [1 (Related publications)]).

16.3 Statements

- Overview of TPCOMP2 statements
- Description of the statements
  - COM - Compare areas of input tapes
  - END - Terminate TPCOMP2
  - LIM - Terminate tape comparison after a certain number of blocks
  - POS - Position magnetic tapes
  - RCD - Specify processing of variable-length input records
  - STP - Stop TPCOMP2 in the event of discrepancies between magnetic tapes

16.3.1 Overview of TPCOMP2 statements

Operation	Operands	Function
COM	stcnnnnn	Compares certain areas of the tape and specifies the print line length and printer mode
END		Terminates statement input
LIM	nnnnn	Terminates comparison after the specified number of blocks
POS1 POS2	pppnnnnn pppnnnnn	Positions the magnetic tapes
RCD	t nnnnn	Specifies the record format and (for fixed-length records) the record length
STP		Stops program execution when a mismatch is detected

16.3.2 Description of the statements

- COM - Compare areas of input tapes
- END - Terminate TPCOMP2
- LIM - Terminate tape comparison after a certain number of blocks
- POS - Position magnetic tapes
- RCD - Specify processing of variable-length input records
- STP - Stop TPCOMP2 in the event of discrepancies between magnetic tapes

16.3.2.1 COM - Compare areas of input tapes

This statement specifies that only certain areas of the input tapes are to be compared. It can also be used to specify the print line length and to select the print format.

Format

COM
stcnnnnn

Operands

s
Number of character positions:

s = 0 132-character print line (default value).

s = 1 160-character print line.

t
Print mode:

t = C Hexadecimal and EBCDIC mode (default value).

t = H Hexadecimal.

t = G EBCDIC.

c
Specifies when to terminate:

c = T Terminate after reading nnnnn tape marks on both tapes.

c = B Terminate after reading nnnnn blocks on both tapes.

nnnnn
Decimal number of tape marks or blocks (with leading zeros).

Examples

```
COM OCT00006
COM 1HB00672
```

> After the operation specified by a COM statement has been completed, the routine will accept a POS statement, if issued. At this point, the comparison process can be terminated by means of an END statement, or the user can call for further comparisons of the input tapes by entering additional COM and POS statements.
> If two consecutive POS statements are used, the comparison process is carried out using default values.

16.3.2.2 END - Terminate TPCOMP2

This statement indicates the end of parameter input.

Format

END

This statement has no operands.

16.3.2.3 LIM - Terminate tape comparison after a certain number of blocks

With this statement, tape comparison is terminated on both tapes after a certain number of blocks.

Format

LIM
nnnnn

Operands

nnnnn
Number of blocks on the two tapes to be compared. If this operand is omitted, all blocks are compared.

16.3.2.4 POS - Position magnetic tapes

This statement positions the magnetic tapes.

Format

POS1 / POS2
pppnnnnn

Operands

The POS2 statement with its operands is needed only when both magnetic tapes are to be positioned.

ppp
Positioning:
ppp=FTM Read forward nnnnn tape marks.
ppp=RTM Read reverse nnnnn tape marks.
ppp=FBK Read forward nnnnn blocks.
ppp=RBK Read reverse nnnnn blocks.
ppp=BOT Rewind to BOT.

nnnnn
Decimal number of tape marks or blocks (with leading zeros).

Examples

```
POS1 FTM00002
POS2 FBK00333
POS1 FBK00017,POS2 FBK00017
```

16.3.2.5 RCD - Specify processing of variable-length input records

This statement signifies that input records are variable in length or that blocked, fixed-length records are to be processed on a logical record level. This statement is required when comparing variable-length records or blocked, fixed-length records.

Format

RCD
t nnnnn *)

*) The operands t and nnnnn are separated by a blank space.

Operands

t
Specifies whether records have a fixed length (F) or a variable length (V).

nnnnn
Fixed-length record size (with leading zeros).
This entry does not apply to variable-length records.

16.3.2.6 STP - Stop TPCOMP2 in the event of discrepancies between magnetic tapes

This statement is used to stop TPCOMP2 when data discrepancies are encountered during comparison of the data on two magnetic tapes. This statement must be repeated before each COM statement or before two consecutive POS statements.

Format

STP

This statement has no operands.

16.4 Using /ADD-FILE-LINK

Prior to calling TPCOMP2, the user must define a file link name for each tape file to be compared (/ADD-FILE-LINK):

First tape: LINK-NAME=COM001
Second tape: LINK-NAME=COM002

For non-cataloged tape files, a catalog entry must first be generated by means of /IMPORT-FILE.

The following parameters of the required commands can be specified:

Cataloged tape file

```
/ADD-FILE-LINK
   LINK-NAME=COM00X
   ,FILE-NAME=dateiname
   [,SUPPORT=*TAPE(VOLUME-LIST=*CATALOG(VOL-SEQ-NUM=n))]
```

Non-cataloged tape file

```
/IMPORT-FILE
   SUPPORT=*TAPE
   (VOLUME=archivnummer
   ,DEVICE-TYPE=XXXX
   ,FILE-NAME=dateiname
   [,PREMOUNT-LIST=n])
/ADD-FILE-LINK
   LINK-NAME=COM00X
   ,FILE-NAME=dateiname
```

In addition, the following operands may be specified in /ADD-FILE-LINK:

```
[,SUPPORT=*TAPE(LABEL-PROCESSNG=*PAR(LABEL-TYPE=...))]
```

When comparing two master tapes, LABEL-TYPE=*NON-STD must be entered. The type of labeling has to be specified via the program parameters in this case.

Examples

1. The preset compare option is to be applied to a cataloged and a non-cataloged tape file.

```
/ADD-FILE-LINK LINK-NAME=COM001,FILE-NAME=DATEI1
/IMPORT-FILE SUPPORT=*TAPE(VOLUME=BAND1,DEVICE-TYPE=<device-type>, -
/            FILE-NAME=DATEI2)
/ADD-FILE-LINK LINK-NAME=COM002,FILE-NAME=DATEI2
/START-EXECUTABLE-PROGRAM FROM-FILE=$TPCOMP2
END
```

2. Data located between the 4th and 5th tape marks on one magnetic tape is to be compared with data between the first 2 tape marks on another magnetic tape. Any discrepancies are to be output in EBCDIC mode on a 160-character printer. The tape files to be compared have both been cataloged.

```
/ADD-FILE-LINK LINK-NAME=COM001,FILE-NAME=DATEI1
/ADD-FILE-LINK LINK-NAME=COM002,FILE-NAME=DATEI2
/START-EXECUTABLE-PROGRAM FROM-FILE=$.SYSPRG.TPCOMP2.210
POS1 FTM00004,POS2 FTM00001
COM  1GT00001
END
```

16.5 TPCOMP2 messages

If an unusual condition is encountered, one of the following messages is output to SYSOUT:

`2801A PARAMETER ERROR`

Meaning
Invalid format.

Response
Terminate.

`2802 ERROR READING PARAMETER CARD`

Meaning
Self-explanatory.

Response
Terminate.

`2803 RECORD TRUNCATED FILE zzz BLK yyyyy IPT x`

Meaning
Block read is greater than block size specified.

Response
Terminate.

`2804 WR xxx ERROR. RETURNCODE: yyyy`

Meaning
Error (yyy) on output to SYSLST (xx=LST) or SYSOUT (xxx=OUT)

Response
Terminate.

`2809A BOT OR DOUBLE TAPE MARK WHILE POSITIONING IPTx`

Meaning
While positioning input tape (x=1/2), double tape mark or EOT was sensed before specified block count was reached. x=1 stands for the first tape, x=2 for the second.

Response
Continue. Tape will rewind and restart in the case of initial positioning. In the case of final positioning, final printout will occur.

`2810 UNMATCHED TM ON IPTx`

Meaning
No suitable tape mark found on input tape (x=1/2).

2810A DBL TM ON IPTx

Meaning
Double tape mark sensed during comparison before specified block count is reached. x=1 stands for the first tape, x=2 for the second.

Response
Continue. Remaining tape will be printed.

```
2821A SHORT RECORD
```

Meaning
Record length is less than fixed length specified.

Response
Continue.

```
2830 yyyyy BLOCK IPTx READ ERROR
```

Meaning
Unrecoverable read error in block yyyyy of input tape. x=1 stands for the first tape, x=2 for the second.

Response
Block yyyyy is compared with the corresponding block of other tape. Processing continues.

```
2800 VERSION xx OF TAPE COMPARE READY
```

Meaning
Program is loaded. "xx" = version number.

Response
Processing continues.

```
2835 OPEN ERROR xy CODE=xxxx COMPARE TERMINATED
```

Meaning
OPEN error for input tape "xy". "xxxx" = DMS error code.

Response
Terminate.

```
2836 CLOSE ERROR xy CODE=xxxx COMPARE TERMINATED
```

Meaning
CLOSE error for input tape "xy". "xxxx" = DMS error code.

Response
Terminate.

```
2840 BTAM ERROR CODE=xxxx
```

Meaning
I/O error. "xxxx" = DMS error code.

Response
Terminate.

17 VOLIN Initialization of disk storage units

Version: Volin V21.0A

The VOLIN (VOLume INitializer) utility routine prepares a disk so that it can be used as a virgin volume in BS2000. Whether the prepared volume is actually new or has already been used in BS2000 or in another operating system is not important. All that matters is that the disk type is supported by BS2000 (see the “System Installation” manual [7 (Related publications)]).

In addition to the label generation, VOLIN allows the mode of use to be defined for a disk. The Data Management System uses this to determine whether the access functions implied by the presence of a PAM key are permitted. If the usage mode is non-key (NK), only NK access methods are allowed.

The volume type is defined by the following characteristics:

- DMS PAM key characteristic, i.e. usage type (PK):
  key (K) / no key (NK)
- minimum allocation unit (AU), i.e. size of the smallest file or, more precisely, the smallestunit of memory: 6 Kbytes / 8 Kbytes / 64 Kbytes
- minimum transfer unit (TU), i.e. smallest unit of transfer between the disk and main memory: 2 Kbytes

Combinations of these characteristics defined with VOLIN yield the following volume types:

Usage type (PK)	Allocation unit (AU)	Transfer unit (TU)	Abbreviation	For disk type
K	6K	2K	K disk	Private, public
NK	6K	2K	NK2 disk	Private, public
NK	8K	2K	NK2(8K) disk	Public
NK	64K	2K	NK2(64K) disk	Public

17.1 VOLIN program execution

The VOLIN subsystem cannot be loaded by means of `/START-SUBSYSTEM`. Instead, it is automatically loaded dynamically by calling the VOLIN utility routine:

```
/START-VOLIN
```

START-VOLIN	Alias: **VOLIN**
VERSION = ***STD** / <product-version> **,MONJV** = ***NONE** / <filename 1..54 without-gen-vers> **,CPU-LIM**IT = ***JOB-RE**ST / <integer 1..32767 *seconds*>	

VOLIN is automatically loaded dynamically by calling the SIR utility routine:

```
/START-SIR (/SIR)
```

VOLIN is terminated with the END statement.

When the utility routine is terminated, the subsystem is unloaded automatically.

17.2 Generating BS2000 labels (label generation)

VOLIN writes new labels on a volume either after the overwriting (statements FMT=YES or FMT=n) or immediately (statement FMT=NO).

If FMT=YES, the disk is overwritten once with a specific bit pattern, if FMT=n it is overwritten “n” times.

VOLIN concludes the initialization of the disk by transferring the new standard volume label. Its presence is a guarantee that the other labels on the volume have also been renewed.

How the disk is used following the VOLIN run depends on whether the initialization has created a private volume, a public volume or the Pubres disk of a pubset.

1. A private volume can be used by the BS2000 Data Management System immediately following the VOLIN run.
2. A public volume can be mounted during the subsequent loading of a BS2000 system (startup) or with a subsequent /IMPORT-PUBSET in order to extend an existing public volume set (pubset).

   The following attributes have global validity in the pubset and must be taken into account:

   - The VSN must match the pubset.
   - The PK (PAM key), AU (allocation unit) and TU (transfer unit) attributes must be the same as those of the Pubres. The following pubset types are thus possible:
     - K pubset (implicit AU=6 Kbytes)
     - NK2 pubset (implicit AU=6 Kbytes)
     - NK2 pubset with AU = 8 Kbytes
     - NK2 pubset with AU = 64 Kbytes
3. A volume which, according to the labels, has been generated by VOLIN for use as Pubres is inoperable and has no direct use in BS2000. An operational pubset can be directly created only with SIR or with an initial installation.
4. A disk initialized with VOLIN, no matter whether it is a private or public volume, can serve as the target volume in any physical data-saving operation (see the FDDRL backup program, and the Copy or Reload functions). Note, however, that the disk type, capacity, and block size of the target volume (2 Kbytes) must be the same as that of the source volume.

Explanatory remarks on the various labels:

1. IPL block

   The IPL block is the first block (2 Kbytes) on the volume. VOLIN uses a dummy IPL here. If an attempt is made to load from this volume, the dummy IPL issues the message `ILLEGAL LOAD FROM BS2000 DATA VOLUME vsn` and causes the system to be placed in the wait state.
2. Standard volume label (SVL)

   The standard volume label is the second block (2 Kbytes) on the volume. It contains flags that identify the disk to the system as a particular volume, pointers to the other labels, and formatting information.

3. Format 5 label (F5)

   The start of this label is anchored in the standard volume label (SVL). The total number of blocks (2 or 4 Kbytes) allocated to this label depends on the type of disk. The format 5 label is used for managing the storage space on the volume. The smallest allocation quantity used by the BS2000 Data Management System is called an allocation unit (1 AU = 6 / 8 / 64 Kbytes, see description on "VOLIN Initialization of disk storage units", blocks are allocated consecutively, beginning with the first 2-Kbyte block). The most important part of the format 5 label is a table specifying which units on the volume are occupied and which are free. After a VOLIN run, all units preceding the F1 start address, as well as the units of the F1 label (see item 4 below), are marked as occupied.

4. Format 1 label (F1)

   This label begins at a unit boundary after the F5 label. Its start address is identical for all the disks of one type.

   The format 1 label exists only on private volumes. It contains the catalog of the files that begin on that volume. On public volumes, the F1 start address corresponds to the beginning of the storage space available for files. Together with the F5 label, the F1 label forms an area on the disk referred to as the VTOC area (VTOC = volume table of contents).

   System administration can define the length of the F1 label during the VOLIN run (by means of the F1SIZE statement; see "Description of the statements").
   VOLIN initializes the F1 label with empty catalog blocks.

17.3 Program execution

VOLIN runs under BS2000 and in the frame of initial installation. Its functionality is also integrated in the SIR utility routine.

The entire range of VOLIN statements (with the exception of a few outdated statements) is now available under SIR in the form of an SDF interface. Therefore, if VIOLIN is called, the old ISP interface will be available; if SIR is called and the appropriate VIOLIN statements are entered, the SDF interface will be available.

This section describes the execution of VOLIN under BS2000. Any differences between this and the initial installation or the version integrated in SIR are described in chapter “SIR Creation of pubsets”. There is also a description of the VOLIN SDF syntax which is integrated in SIR. This manual shows the correspondence between ISP and SDF statements and operands (see "Correspondence between ISP and SDF statements"). Any references to VOLIN statements should be taken to mean ISP statements.

VOLIN uses privileged functions and can run only under the TSOS user ID. Any attempt to execute VOLIN under another user ID results in message `NVL0001` and immediate termination of the routine.

VOLIN can be called in an interactive job or in a batch job.
In interactive mode VOLIN handles questions and answers regarding disk initialization via the terminal, otherwise it handles them via the console to which system administration duties have been assigned.

Any number of disks can be initialized in one VOLIN run. Disks are initialized one after the other. Every initialization is controlled by a set of statements read in from SYSDTA. In the event of an error, initialization of the volume currently being processed is terminated. In interactive mode, a new set of statements may then be entered; in a batch job, the VOLIN run is also terminated (TERM MODE=ABNORMAL).

> Multiple concurrent VOLIN runs can degrade BS2000 operation. They should therefore be avoided. At the very least, disks to be overwritten should not be accessed via a channel or controller that is also being used for other heavily used disks.

Assignment of volumes

VOLIN requests the disk to be initialized via the system device management facility and reserves it exclusively at its own system.

VOLIN requires at least the specification of the device type or the mnemonic device name (DEVICE and UNIT statements) in order to assign a disk. If a specific disk is to be initialized, VOLIN must be provided with the old volume serial number as well as the disk type the mnemonic device name (via the CHECK statement; see "Description of the statements"). If the volume thus specified is already mounted but not reserved, then it is immediately assigned by device management. If device management cannot reserve the volume specified, VOLIN terminates initialization.

If VOLIN is given only the device type, any disk of this type can be assigned (“scratch volume”). VOLIN uses the SVL to inform system administration about the mounted volume and asks whether the volume may be overwritten. If the answer is YES (i.e. system administration permits overwriting), the mounted disk is initialized. If the answer is negative, three different results are possible:

- Initialization is terminated if the volume is on a fixed-disk device specified with the UNIT statement.
- If no disk device was specified with the UNIT statement, a different disk device is requested.
- If the operator declines to provide another disk when responding to the message from device management, initialization is terminated.

It is possible to specify the mnemonic name instead of the device type. VOLIN then determines the device name from the WORD mnemonic name. The procedure is then the same as when both the device type and mnemonic name are specified.

You must use the mnemonic device name rather than the device type to select a device when running VOLIN from SIR.

It is important to ensure that initialization is coordinated with the work done by other systems that also have access to the device occupied by the volume.

Special features for snap and clone units

Initialization of a snap or clone unit is rejected by VOLIN. When an original unit is initialized using the VOLIN utility routine, VOLIN terminates all snap or clone sessions with this unit as required after an operator query, and possibly those of the SRDF target unit.
This corresponds to `/STOP-SNAP-SESSION UNIT=...,SNAP-UNIT=*ALL` or `/STOP-CLONE-SESSION UNIT=...,CLONE-UNIT=*ALL`.

17.4 Operating instructions

Command level

The program is started under TSOS by `/START-VOLIN` or `/VOLIN`.
For reasons of compatibility, the command `/START-EXECUTABLE-PROGRAM FROM-FILE=$VOLIN` is also permitted.
In this case, the statements are provided in the ISP interface.

As an alternative, SIR can be started under TSOS with `START-SIR` or `/SIR` VOLIN can then be run under the SDF interface.

Console level

VOLIN sends messages to the console (routing code A) and to SYSOUT. When VOLIN is running in a batch job, responses to messages are also read in from the console, See section “Program execution”.

VOLIN statements

Every initialization of a disk is controlled by a set of statements that VOLIN reads in from SYSDTA. Once a set of statements has been read in, VOLIN checks it for consistency and completeness. If no errors are present, initialization is performed and a new set of statements is then requested.

The statements are passed to VOLIN in input lines. Each input line may contain several statements, separated by commas, but the length of an input line must not exceed 251 characters. In accordance with the rules of the BS2000 command language it is possible to insert blanks and comments.

VOLIN requests the first line of a set of statements with the message:

```
NVL0002 ENTER VSN AND DEVICE TYPE AND/OR UNIT, OR TERMINATE VOLIN WITH 'END'
```

All subsequent input lines are requested with the message:

```
NVL0003 ENTER ADDITIONAL INITIALIZATION PARAMETERS OR TERMINATE INPUT WITH 'EOT=YES'
```

A set of statements is concluded with a line containing EOT=YES, either alone or together with other statements. A blank line (ETX) has the same effect as EOT=YES. Each line may contain any combination of different statements. Statements that have already been specified can be repeated in new input lines and thus modified.

VOLIN is terminated when message `NVL0002` is answered with END.
If the same response is given to message `NVL0003`, VOLIN first generates an EOT=YES condition and processes the set of statements thus concluded. Subsequently, no new set of statements is requested and the run is terminated.

End of file (EOF) on SYSDTA has the same effect as END.

17.5 Statements

The statements of SIR are described in section “Statements”.

17.5.1 Overview of VOLIN statements

Format	Function
[VSN=vsn]	Volume serial number after initialization.
[DEVICE=device-type]	Device type of the disk (must be specified if no UNIT is specified).
[C[HECK]=SCRATCH / vsn]]	Volume serial number prior to initialization, or SCRATCH for an arbitrary disk.
[UNIT=mn]	Mnemonic device name; must be specified when initialization is to take place on a specific device.
[F[MT]=YES / NO / n]	Initialization with (number = n) or without overwriting.
[UNAME=name]	User (owner) name to be left in the SVL (not evaluated by the system).
[{ F1SIZE / VTOC } =n]]	Number of PAM blocks for the format 1 label.
[FORMAT= { UNCHANGED / K / NK([PHYSICAL-BLOCK-SIZE=] 2K(ALLOCATION-UNIT = 6 / 8 / 64)) }]	Defines the volume type.
[EOT=Y[ES]]	Concludes the statement set with the current input line.
[END / **H**ALT]	Terminates the VOLIN run after processing the current set of statements.

17.5.2 Description of the statements

VSN = vsn
Volume serial number (must be specified)

The alphanumeric value, max. 6 characters, is written in the volume serial number field of the standard volume label (SVL). If the volume is to be initialized for later use as a public volume, the following should be noted, depending on the length of the pubset ID:

- single character ID for the pubset
  The volume serial number begins with the characters PUB. The fourth character is the pubset ID. The final two characters must be numeric, e.g. PUBA01. Zeros in these two positions identify a volume as the system residence, e.g. PUBA00.
- two- to four-character ID for the pubset
  The first characters are the pubset identifier. The end of the identifier is indicated by a period in the next position. The conventions applying to the remaining positions are as follows:
  - NK2 (8K, 64K) disks:
    digits (0..9) and letters (A..Z) are permissible
    Examples
    VSN with two-character pubset identifier: RZ.90Z
    VSN with three-character pubset identifier: KH2.W0
    VSN with four-character pubset identifier: AFLN.8
  - NK2 (6K) disks:
    The last character must be a digit or a letter from the set A..V. All other positions must contain zeros.
    Examples
    VSN with two-character pubset identifier: AB.005
    VSN with three-character pubset identifier: ABC.0F
    VSN with four-character pubset identifier: ABCD.0

DEVICE = device-type
Device type (must be specified if no UNIT was specified).
Type of disk to be initialized/repaired.
The permissible values for this statement: see the “System Installation” manual [7 (Related publications)].

The DEVICE specification may be omitted if the device has already been specified in a UNIT statement.

CHECK = SCRATCH
Entry (default value) when no specific volume is to be initialized. VOLIN reports on the volume assigned and asks whether it can be overwritten.

CHECK = vsn
Volume serial number (max. 6 characters).
Old volume serial number of the disk to be initialized. Only a volume bearing this volume serial number will be initialized. Following assignment, the volume is overwritten without first requesting confirmation from the user.

UNIT = mn
Mnemonic device name.
mn is the 2- or 4-character name assigned to a particular device at hardware generation. This statement must be entered if the user wishes to specify a particular device for initialization. If the statement UNIT=mn is not specified, either the system selects an available device of the correct type, or the operator does so by responding to the MOUNT message.

The DEVICE specification may be omitted if the device has already been specified in a UNIT statement.

The initialization is not carried out if the DEVICE and UNIT specifications contradict each other.

FMT = YES
The disk is to be overwritten with a specific bit pattern once during the initialization run (default value). YES is equivalent to FMT=1 (see FMT=n below).

FMT = NO
Initialization of the disk is to be carried out without overwriting.

FMT = n
The disk is to be overwritten with a specific bit pattern “n” times during the initialization run. n is a decimal number between 0 and 10.
0 is equivalent to FMT=NO, i.e. no overwriting. In each overwrite pass, a specific test pattern is recorded. For a new pass, the test pattern is changed to the next one in the (cyclical) sequence.

> FMT is ignored for Thin Devices . Thin devices can only be recognized with SHC-OSD.

UNAME = name
Owner name consisting of up to 10 alphanumeric characters and written in the name field of the SVL. The first character must be a letter or one of the following characters: If no value is specified, blanks are used.

F1SIZE = n
Decimal number (1 - 32766).
This statement applies only to private volumes. The decimal number is the number of PAM blocks to be reserved for the format 1 label. If necessary, the specified value is rounded up to give an integral number of units. A maximum of about 16 files can be cataloged in one block.
If no value is specified, 1000 PAM blocks are allocated for the F1 label on a private volume. (For compatibility reasons, the keyword VTOC may be used instead of F1SIZE.)

FORMAT = UNCHANGED
The volume type is to remain unchanged. This value assumes the volume has a valid SVL. Otherwise value K is used by default.

FORMAT = K
The volume is to be usable with PAM keys.

FORMAT = NK(PHYSICAL-BLOCK-SIZE = 2K(ALLOCATION-UNIT = 6/8/64))
The volume is to be usable without PAM keys. Only the NK DMS access methods are permissible. The block size is to be 2 Kbytes, the smallest allocation unit (i.e. the smallest memory unit provided by the system) is to be 6, 8 or 64 Kbytes.
Default for ALLOCATION UNIT is 6 Kbytes.

PHYSICAL-BLOCK-SIZE = 2K
Length of the minimum transfer unit (TU), i.e. size (in Kbytes) of the smallest possible data block for transfer between the disk and main memory.

ALLOCATION-UNIT = 6 / 8 / 64
Minimum allocation unit in Kbytes, i.e. smallest unit in memory made available by the system when a file is created.

Restrictions

The following can be specified for public volumes only:
FORMAT=NK(PHYSICAL-BLOCK-SIZE=2K(ALLOCATION-UNIT=8))
FORMAT=NK(PHYSICAL-BLOCK-SIZE=2K(ALLOCATION-UNIT=64))

EOT = YES
The set of statements is to be concluded with the current input line and initialization carried out. (All statements in the current input line are processed.)

END
VOLIN is to be terminated after the current set of statements has been processed. (For reasons of compatibility, H or HALT may be used instead of END. Apart from END or HALT no other statements will be accepted in the input line.)

17.5.3 Correspondence between ISP and SDF statements

All relevant VOLIN statements are also provided in the form of an SDF interface.
A prerequisite is that the SIR program, which starts the VOLIN subsystem implicitly, is called. A detailed description of the SDF statements for SIR can be found in the section “Statements”.

For your convenience, this section contains a brief overview of the correspondence between ISP and SDF statements and operands. Please note the following:

- The SF statements INITIALIZE-PRIVATE-VOLUME and INITIALIZE-PUBLIC-VOLUME provide a label generation feature for private and public disks respectively.
- All other ISP statements are provided in the form of operands under the relevant SDF statements.
- Some ISP statements have become superfluous or are relevant only for outdated hardware, and have therefore been omitted from the SDF interface.

The digits 1 and 2 in the right-hand column of the table below stand for the following statements:

1 = `INITIALIZE-PRIVATE-VOLUME`

2 = `INITIALIZE-PUBLIC-VOLUME`

ISP statement	SDF operand / statement	in statement 1	in statement 2
VSN	VOLUME	x	x
DEVICE	Not supported (select using UNIT)		
CHECK	OVERWRITE-DISK	x	x
UNIT	UNIT	x	x
FMT	FUNCTION = *FORMAT-DISK	x	x
	FUNCTION = *CREATE-LABELS-ONLY	x	x
UNAME	Not supported		
F1SIZE	F1-LABEL-SIZE	x	
FORMAT	FORMAT	x	x
EOT	Implicit in termination of the statement		
END	SIR END statement		

18 Related publications

You will find the manuals on the internet at *http://bs2manuals.ts.fujitsu.com*. You can order printed versions of those manuals which are displayed with the order number.

[1] **BS2000 OS DX**
Commands
User Guide

[2] **BS2000 OS DX**
Diagnostics Handbook
User Guide

[3] **BS2000 OS DX**
DMS Macros
User Guide

[4] **BS2000 OS DX**
Introduction to DMS
User Guide

[5] **BS2000 OS DX**
Introduction to System Administration
User Guide

[6] **BS2000 OS DX**
Executive Macros
User Guide

[7] **BS2000 OS DX**
System Installation
User Guide

[8] **BS2000 OS DX**
System-Managed Storage
User Guide

[9] **BS2000 OS DX**
Performance Handbook
User Guide

[10] **Fujitsu Servers BS2000 SE Series**
Operation and Administration
User Guide

[11] **openNet Server**
BCAM
User Guide

[12] **BINDER**
Binder in BS2000
User Guide

[13] **BLSSERV**
Dynamic Binder Loader / Starter in BS2000
User Guide

[14] **EDT**
Statements
User Guide

[15] **IMON**
Installation Monitor
User Guide

[16] **JV**
Job Variables
User Guide

[17] **LMS**
SDF Format
User Guide

[18] **MAREN**
Tape Management in BS2000
User Guide

[19] **POSIX**
POSIX Basics for Users and System Administrators

User Guide

[20] **SDF**
SDF Dialog Interface
User Guide

[21] **SECOS**
Security Control System - Audit
User Guide

[22] **SHC-OSD**
Storage Management for BS2000
User Guide

[23] **VM2000**
Virtual Machine System
User Guide