Adabas

Utilities

Version 8.6.2

October 2026

ADABAS & NATURAL

This document applies to Adabas Version 8.6.2 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 1971-2026 Software GmbH, Darmstadt, Germany and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software GmbH product names are either trademarks or registered trademarks of Software GmbH and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software GmbH and/or its subsidiaries is located at https://softwareag.com/licenses.

Use of this software is subject to adherence to Software GmbH's licensing conditions and terms. These terms are part of the product documentation, located at https://softwareag.com/licenses and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software GmbH Products / Copyright and Trademark Notices of Software GmbH Products". These documents are part of the product documentation, located at https://softwareag.com/licenses and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software GmbH.

Document ID: ADAMF-AADAMFUTILITIES-862-20261013

Table of Contents

Utilities xxi
1 About this Documentation 1
Document Conventions 2
Online Information and Support 2
Data Protection 3
I Conventions 5
2 Conventions 7
About this Documentation 8
Control Statement Syntax 8
Syntax Conventions 9
Control Statement Rules 11
Parameter Values 12
II ADACDC Utility: Changed-Data Capture 15
3 Functional Overview 17
Phases of Operation and Resulting Files 19
Primary Input Data 21
Primary Output File 22
Transaction File 23
4 Running the Utility 25
Optional Parameters 26
Using ADACDC With ISNREUSE 29
5 Operating System Considerations 31
6 The ADACDC User Exit 33
Installing the Exit 34
User Exit Interface 34
User Exit Calls 36
7 Examples 39
8 JCL Requirements and Examples 41
Requirements to Run ADACDC JCL on z/OS 42
III ADACHK Utility: Database Consistency Check Utility that Runs Concurrently with Normal Database Operations 45
9 Functional Overview 47
ADACHK Function Equivalences with Other Utility Functions 50
Detailed Processing 52
ADACHK Statistics 54
Spanned Record Considerations 55
Cipher File Considerations 56
Syntax Checking with the TEST Parameter 56
Sort Data Set Considerations 56
SORTTYPE=EXTERNAL Operating System Considerations 57
ADACHK Print Function Output Format 58
Sample ADACHK Jobs 60
10 ACCHECK: Check Address Converter against Data Storage 61

Essential Parameters 62
Optional Parameters and Subparameters 63
Examples 65
11 ACSCAN: Scan the address converter to identify reusable ISNs 67
Essential parameters 69
Optional Parameters and Subparameters 69
12 ASSOPRINT: Print/Dump Associator Blocks 73
Essential Parameters 74
Optional Parameters and Subparameters 74
Example 76
13 CHECK: Perform all ADACHK Checks 77
Essential Parameters 79
Optional Parameters and Subparameters 79
Example 83
14 CLOGPRINT: Print/Dump Command Log Blocks 85
Essential Parameters 86
Optional Parameters and Subparameters 86
Example 88
15 DATAPRINT: Print/Dump Data Storage Blocks 89
Essential Parameters 90
Optional Parameters and Subparameters 90
Example 92
16 DSCHECK: Check Data Storage 93
Essential Parameters 95
Optional Parameters and Subparameters 95
Examples 100
17 DSIMPRINT: Print Delta Save Image Data Set Blocks 101
Essential Parameters 102
Optional Parameters and Subparameters 102
Example 104
18 DSSTPRINT: Print Data Storage Space Table 105
Essential Parameters 106
Optional Parameters and Subparameters 106
Example 108
19 FCBPRINT: Print/Dump File Control Block (FCB) 109
Essential Parameters 110
Optional Parameters and Subparameters 110
Example 112
20 FDTPRINT: Print/Dump Field Definition Table (FDT) 113
Essential Parameters 114
Optional Parameters and Subparameters 114
Example 116
21 FSTPRINT: Print/Dump Free Space Table (FST) 117
Essential Parameters 118
Optional Parameters and Subparameters 118

Example 120
22 GCBPRINT: Print/Dump General Control Blocks (GCBs) 121
Essential Parameters 122
Optional Parameters and Subparameters 122
Example 124
23 ICHECK: Check Index Physical Structure 125
Essential Parameters 126
Optional Parameters and Subparameters 126
Example 128
24 NIPRINT: Print/Dump Normal Index 129
Essential Parameters 130
Optional Parameters and Subparameters 130
Example 132
25 PLOGPRINT: Print/Dump Protection Log Blocks 133
Essential Parameters 134
Optional Parameters and Subparameters 134
Example 136
26 PPTPRINT: Print/Dump Parallel Participant Table (PPT) 137
Essential Parameters 138
Optional Parameters and Subparameters 138
Example 140
27 RLOGPRINT: Print/Dump Recovery Log 141
Essential Parameters 142
Optional Parameters and Subparameters 142
Example 144
28 SORTPRINT: Print/Dump Sort Data Set 145
Essential Parameters 146
Optional Parameters and Subparameters 146
Example 148
29 TEMPPRINT: Print/Dump Temporary Data Set 149
Essential Parameters 150
Optional Parameters and Subparameters 150
Example 152
30 UIPRINT: Print/Dump Upper Index 153
Essential Parameters 154
Optional Parameters and Subparameters 154
Example 156
31 VALIDATE: Validate All Database Files 157
Essential Parameters 158
Optional Parameters and Subparameters 158
Output 162
Examples 165
32 WORKPRINT: Print/Dump Work Data Set 167
Essential Parameters 168
Optional Parameters and Subparameters 168

Examples 170
33 JCL Requirements and Examples 171
Requirements to Run ADACHK JCL 172
IV ADACMP Utility: Compress-Decompress Data 177
34 Functional Overview 179
COMPRESS Function Overview 180
DECOMPRESS Function Overview 181
35 Input Data Requirements 183
Input Data Structure 184
Multiple-Value Field Count 184
Periodic Group Count 186
System Field Requirements 189
Variable-Length Field Size 190
36 Processing 191
Segmented Record Considerations 192
Logically Deleted Fields 198
Data Verification 198
Data Compression 198
Representation of LOB Values and Value References in Uncompressed Data 200
Identifying MU and PE Occurrences Greater Than 191 in Compressed Records 201
Restart Considerations 201
User Exit 6 201
37 COMPRESS: Compress an Adabas File 203
Essential Parameters and Subparameters 204
Optional Parameters and Subparameters 206
ADACMP COMPRESS Examples 214
38 DECOMPRESS: Decompress an Adabas File 219
Optional Parameters and Subparameters 240
Decompressing Multiclient Files 226
ADACMP DECOMPRESS Examples 227
39 Field Definition Statements 229
FNDEF: Field and Group Definition 231
FNDEF: Periodic Group Definition 250
COLDE: Collation Descriptor Definition 253
HYPDE: Hyperdescriptor Definition 256
PHONDE: Phonetic Descriptor 259
SUBDE: Subdescriptor Definition 261
SUBFN: Subfield Definition 263
SUPDE: Superdescriptor Definition 264
SUPFN: Superfield Definition 273
40 JCL Requirements and Examples 275
User Exits with ADACMP 276
Requirements to Run ADACMP JCL 277

41 COMPRESS Function Output 283
Compressed Data Records 284
Rejected Data Records 284
ADACMP Report 290
42 DECOMPRESS Function Output 295
Rejected Data Records 296
V ADACNV Utility: Database Conversion 301
43 Functional Overview 303
Database Status 304
Procedure 305
44 CONVERT: Convert Database to Higher Version 307
Optional Parameters 308
Conversion Considerations 309
Example 310
45 REVERT: Revert Database to Lower Version 311
Essential Parameter and Subparameter 312
Optional Parameter 312
Reversion Considerations 313
Example 315
46 JCL Requirements and Examples 317
Requirements to Run ADACNV JCL 318
VI ADADBS Utility: Database Services 321
47 Functional Overview 323
Syntax Checking with the TEST Parameter 324
48 ADD: Add Data Set 325
Associator or Data Storage Data Set 326
Essential Parameter and Subparameter 327
Optional Parameters 327
Examples 327
49 ADDCLOG: Dynamically Add CLOG Data Sets 329
Essential Parameters 331
Optional Parameters 331
Examples 332
50 ADDPLOG: Dynamically Add PLOG Data Sets 333
Essential Parameters 335
Optional Parameters 335
Examples 336
51 ALLOCATE: Allocate File Extent 337
Essential Parameters 338
Optional Parameters 339
Example 339
52 CHANGE: Change Standard Length or Format of a Field 341
Essential Parameters 342
Optional Parameters 345
Example 346

53 CVOLSER: Print Adabas Extents on Given Volume 347
Essential Parameter 348
Optional Parameters 348
Example 349
54 DEALLOCATE: Deallocate File Extent 351
Essential Parameters 352
Optional Parameters 353
Example 353
55 DECREASE: Decrease Last Associator or Data Storage Data Set Size 355
Essential Parameter 356
Optional Parameters 356
Example 357
Procedure 357
56 DELCLOG: Dynamically Deleting CLOG Data Sets 359
Essential Parameters 360
Optional Parameters 361
Examples 361
57 DELCP: Delete Checkpoint Records 363
Essential Parameter 364
Optional Parameters 364
Example 365
58 DELDE: Logically Deleting a Descriptor 367
Essential Parameters 368
Optional Parameters 368
Example 369
59 DELETE: Delete File 371
Essential Parameter 372
Optional Parameters 372
Examples 373
60 DELFN: Logically Delete Fields 375
Essential Parameter 377
Optional Parameters 377
Example 378
61 DELPLOG: Dynamically Deleting PLOG Data Sets 379
Essential Parameters 380
Optional Parameters 381
Examples 381
62 DEVENTLOG: Display Adabas Event Log 383
Optional Parameters 384
Examples 385
63 DSREUSE: Reuse Data Storage Blocks 387
Essential Parameters 388
Optional Parameters 388
Example 389
64 ENCODEF: Change File Encoding 391

Essential Parameter 392
Optional Parameters 393
Example 393
65 EXPFILE: Insert or Remove Files in an Expanded File Chain 395
Essential Parameters 396
Optional Parameters 397
Example 397
66 INCREASE: Increase Last Associator or Data Storage Data Set Size 399
Essential Parameter 400
Optional Parameters 400
Example 401
General Procedure 401
z/OS Procedures 401
67 ISNREUSE: Reuse ISNs 405
Essential Parameters 406
Optional Parameters 406
Examples 407
68 MODFCB: Modify File Parameters 409
Essential Parameter 410
Optional Parameters 410
Example 413
69 MUPEX: Set Maximum Count for MU and PE Fields 415
Syntax 417
Essential Parameters 417
Optional Parameters 417
Example 418
70 NEWFIELD: Add New Field 419
Essential Parameter 420
Optional Parameters 421
Example 422
71 ONLADD: Allocating New ASSO or DATA Space Dynamically 423
Essential Parameters 424
Optional Parameters 425
Restrictions and Considerations 426
Example 426
72 ONLINCREASE: Allocating Appended ASSO or DATA Space Dynamically 429
Essential Parameters 430
Optional Parameters 431
Restrictions and Considerations 431
Example 432
73 ONLINVERT: Start Online Invert Process 435
Essential Parameters 436
Optional Parameters 438
Example 438

74 ONLREORFASSO: Start Online Reorder Associator for Files 439
Essential Parameters 440
Optional Parameters 441
Example 441
75 ONLREORFDATA: Start Online Reorder Data for Files 443
Essential Parameters 444
Optional Parameters 445
Example 446
76 ONLREORFILE: Start Online Reorder Associator and Data for Files 447
Essential Parameters 448
Optional Parameters 449
Example 450
77 OPERCOM: Issue Adabas Operator Commands 451
Using OPERCOM Commands in Cluster Environments 452
Optional Parameters 452
Operator Commands 453
78 PRIORITY: Change User Priority 499
Essential Parameter 500
Optional Parameters 500
Example 501
79 REACTLOG: Reactivating Command Logging 503
Optional Parameters 504
Example 504
80 RECORDSPANNING: Enable or Disable Record Spanning 505
Syntax 506
Essential Parameters 506
Optional Parameters 506
Example 507
81 RECOVER: Recover Space 509
Optional Parameters 510
82 REFRESH: Set File to Empty Status 511
Essential Parameter 512
Optional Parameters 512
Example 513
83 REFRESHSTATS: Reset Statistical Values 515
Optional Parameters 516
Example 518
84 RELEASE: Release Descriptor 519
Essential Parameters 520
Optional Parameters 521
Example 521
85 RENAME: Rename File or Database 523
Essential Parameter 524
Optional Parameters 524
Examples 525

86 RENUMBER: Change File Number 527
Essential Parameter 528
Optional Parameter 528
Example 529
87 REPLICATION: Activating or Deactivating Replication 531
Essential Parameter 532
Optional Parameter 532
Examples 534
88 ADADBS REPTOR: Activate, Deactivate, Open, or Close Event Replicator Resources 535
Essential Parameters 536
Optional Parameters 537
Examples 538
89 RESETDIB: Reset Entries in Active Utility List 541
Essential Parameters 542
Optional Parameters 542
Examples 543
90 RESETPPT: Reset PPT Blocks 545
Syntax 547
Essential Parameters 547
Optional Parameters 547
91 SPANCOUNT: Count Spanned Records 549
Syntax 550
Essential Parameters 550
Optional Parameters 550
Example 551
92 TRANSACTIONS: Suspend and Resume Update Transaction Processing 553
Essential Parameters 555
Optional Parameters 555
Example 556
93 UNCOUPLE: Uncouple Files 557
Essential Parameter 558
Optional Parameters 558
Example 559
94 UNDELDE: Undeleting a Logically Deleted Descriptor 561
Essential Parameters 562
Optional Parameters 562
Example 563
95 UNDELFN: Logically Undelete Fields 565
Essential Parameter 566
Optional Parameters 566
Example 567
96 JCL Requirements and Examples 569
Collation with User Exit 570
Requirements to Run ADADBS JCL 570

VII ADADEF Utility: Define a Database 573
97 Functional Overview 575
Database Components 576
Checkpoint File 576
98 DEFINE: Defining a Database and Checkpoint File 577
Essential Parameters 579
Optional Parameters 580
Examples 584
99 MODIFY: Change Database Options 587
Optional Parameters 588
Examples 590
100 NEWWORK: Defining a Work File 591
Essential Parameter 592
Optional Parameters 592
Example 593
101 JCL Requirements and Examples 595
Requirements to Run ADADEF JCL 596
VIII ADADRU Utility: Database Storage Usage Reports 599
102 Functional Overview 601
Spanned Record Handling 602
Syntax Checking with the TEST Parameter 604
Global OUTPUT and ACODE Parameter Usage 605
103 SPACEUSAGE: Database File Space Usage 607
SPACEUSAGE File Group Processing 608
Syntax 612
Sample Reports 619
104 JCL Requirements and Examples 751
Requirements to Run ADADRU JCL 752
IX ADAFRM Utility: Format Adabas Database Components 755
105 Functional Overview 757
Statement Restrictions 758
Formatting Operation 758
106 Formatting Database Components 759
Formatting Modes 760
Enhanced Formatting 760
Syntax 761
Essential Parameter 763
Optional Parameters 763
Examples 765
107 JCL Requirements and Examples 767
Requirements to Run ADAFRM JCL 768
X ADAINV Utility: Inverted List Management 771
108 Functional Overview 773
109 COUPLE: Define File-Coupling Descriptors 775
Essential Parameters 776

Optional Parameters 776
Example 778
Temporary Space for File Coupling 778
Associator Coupling Lists 779
Space for Coupling Lists 780
Space Allocation 781
Enhanced Temp and Sort Processing 781
110 INVERT: Create Descriptors 783
Essential Parameters 784
Optional Parameters and Subparameters 784
Space Allocation for the INVERT Function 787
Enhanced Temp and Sort Processing 787
Examples 788
111 JCL Requirements and Examples 789
Collation with User Exit 790
Requirements to Run ADAINV JCL 790
XI ADALOD Utility: File Loader 795
112 Functional Overview 797
113 LOAD: Load a File 799
Essential Parameters 802
Optional Parameters and Subparameters 804
Examples 820
LOAD Data and Space Requirements 821
Loading Expanded Files 825
Loading Multiclient Files 827
Enhanced Temp and Sort Processing 828
114 UPDATE: Add/Delete Records 829
Essential Parameters 831
Optional Parameters and Subparameters 832
Examples 838
Formats for Specifying ISNs 839
UPDATE Data and Space Requirements 840
Mass Updates of Expanded Files 842
Enhanced Temp and Sort Processing 843
115 Loader Storage Requirements and Use 845
116 TEMP Data Set Space Usage 847
Sequential TEMP Data Set 848
117 ADALOD Space/Statistics Report 851
118 JCL Requirements and Examples 853
Collation with User Exit 854
Requirements to Run ADALOD JCL 854
XII ADAMER Utility: ADAM Estimation 859
119 Functional Overview 861
120 Estimate ADAM Access Requirements 863
Essential Parameters 864

Optional Parameters 864
Examples 866
121 ADAMER Output Report Description 867
122 JCL Requirements and Examples 869
Requirements to Run ADAMER JCL 893
XIII ADAORD Utility: Reordering Functions 871
123 Functional Overview 873
Reorder Functions 874
Restructure Functions 875
Store Function 875
Space Allocation 876
Adabas 8 Considerations 876
124 REORASSO: Reorder Associator 877
Optional Parameters and Subparameters 879
Examples 883
125 REORDATA: Reorder Data Storage 885
Optional Parameters and Their Subparameters 886
Examples 889
126 REORDB: Reorder Database 891
Optional Parameters and Subparameters 894
Examples 900
127 REORFASSO: Reorder Associator for a Single File 903
Essential Parameter 905
Optional Parameters 905
Examples 908
128 REORFDATA: Reorder Data Storage for a Single File 911
Essential Parameter 912
Optional Parameters 913
Examples 916
129 REORFILE: Reorder File 917
Essential Parameter 920
Optional Parameters 920
Examples 925
130 RESTRUCTUREDB: Restructure Database 927
Optional Parameters and Subparameters 929
Examples 982
131 RESTRUCTUREF: Restructure Single Files 933
Essential Parameter 935
Optional Parameters 935
Examples 938
132 STORE: Store Files 939
Optional Parameters and Subparameters 942
Examples 949
133 JCL Requirements and Examples 951
Requirements to Run ADAORD JCL 952

XIV ADAPLP Utility: Print Data Protection Records from PLOG/Work 955
134 Functional Overview 957
135 ADAPLP Syntax and Examples 959
Optional Parameters and Subparameters 961
Examples 1033
136 JCL Requirements and Examples 967
Requirements to Run ADAPLP JCL 968
XV ADARAI Utility: Adabas Recovery Aid 971
137 Functional Overview 973
Concepts and Components 974
138 CHKDB: Check the Database Recovery Status 977
139 DISABLE: Disable Recovery Logging 979
140 LIST: Display Current RLOG Generations 981
Syntax 983
Optional Parameters 983
Examples 984
141 PREPARE: Initialize and Start the RLOG 991
Syntax 993
Essential Parameter 993
Optional Parameters 993
Examples 994
142 RECOVER: Build a Recovery Job Stream 995
Recovery Processing 996
Optimized Recovery Processing 998
Requirements 998
Restrictions 999
Input Needed for Recovery 1000
Output from the Recovery Operation 1000
Executing the RECOVER Function 1001
File-Level Recovery 1002
Syntax 1003
Optional Parameters and Subparameters 1003
Examples 1121
Skeleton Job Control 1006
User Exit to Change JCL 1009
Prerecovery Checking 1009
Restarting the RECOVER Function or Recovery Job Stream 1010
143 REMOVE: Remove the Recovery Aid 1011
Example 1012
144 JCL Requirements and Examples 1013
Requirements to Run ADARAI JCL 1014
XVI ADAREP Utility: Database Status Report 1021
145 Functional Overview 1023
146 Report Syntax 1025
Optional Parameters 1026

Examples 1033
147 Processing Save Tape Input 1035
Supplying Protection Log Input 1036
Checking Input Tapes 1037
Concurrent Parameters 1037
Reports for Delta Save Tapes 1037
Report Layout 1038
148 Report Description 1039
General Database Information 1040
File Information 1054
Checkpoint Information 1067
149 JCL Requirements and Examples 1075
Requirements to Run ADAREP JCL 1076
XVII ADARES Utility: Database Recovery 1079
150 Functional Overview 1081
Using ADARES in Adabas Nucleus Cluster Environments 1083
151 BACKOUT Functions 1087
152 BACKOUT: Back Out Updates Using the Sequential Protection Log (SIBA) 1089
Essential Parameters 1091
Optional Parameters and Subparameters 1092
Examples 1096
153 BACKOUT DPLOG or MPLOG: Back Out Updates Using the Dual or Multiple Protection Log 1097
Executing the Function 1098
Syntax 1099
Essential Parameter 1100
Optional Parameters 1100
Example 1105
154 CLCOPY: Copy Dual Command Log 1107
Optional Parameters 1108
Examples 1109
155 COPY: Copy a Sequential Protection Log or Save Tape 1111
Optional Parameters 1112
Examples 1114
156 MERGE CLOG: Merge Nucleus Cluster Command Logs 1115
Essential Parameter 1116
157 PLCOPY: Copy Protection Log to Sequential Data Set 1117
Optional Parameters 1119
Examples 1121
158 REGENERATE: Regenerate Updates 1123
Syntax 1125
Essential Parameters 1125
Optional Parameters and Subparameters 1126
Examples 1132

Output Statistics 1133
159 REPAIR: Repair Data Storage Blocks 1135
Syntax 1332
Essential Parameter 1137
Optional Parameters 1137
Examples 1138
160 Multithreaded Processing Statistics 1139
161 JCL Requirements and Examples 1141
Requirements to Run ADARES JCL 1142
XVIII ADASAV Utility: Save/Restore Database or Files 1151
162 Functional Overview 1153
RESTONL and RESTORE Functions 1154
Adabas Release Support 1155
163 RESTONL: Restore Database from Online Source 1157
Conditions 1158
Result 1159
Syntax 1160
Optional Parameters and Subparameters 1160
Examples 1162
164 RESTONL FILES: Restore Files to Original RABNs from Online Source 1163
Conditions 1164
Result 1165
Syntax 1166
Optional Parameters and Subparameters 1167
Examples 1173
165 RESTONL FMOVE: Restore Files to Any RABNs from Online Source 1175
Conditions 1176
Result 1177
Syntax 1178
Optional Parameters 1179
Examples 1189
166 RESTONL GCB: Restore Database Incremental from Online Source 1191
Conditions 1192
Result 1193
Syntax 1194
Optional Parameters and Subparameters 1194
Examples 1200
167 RESTORE: Restore Database from Offline Source 1201
Conditions 1202
Result 1203
Syntax 1204
Optional Parameters 1204
Examples 1206
168 RESTORE FILES: Restore Files to Original RABNs from Offline Source 1207
Conditions 1208

Result 1209
Syntax 1210
Optional Parameters 1211
Examples 1216
169 RESTORE FMOVE: Restore Files to Any RABNs from Offline Source 1219
Conditions 1220
Result 1221
Syntax 1221
Optional Parameters 1223
Examples 1232
170 RESTORE GCB: Restore Database Incremental from Offline Source 1235
Conditions 1236
Result 1237
Syntax 1238
Optional Parameters 1238
Examples 1243
171 RESTPLOG: Restore Protection Log Only 1245
Essential Parameters 1246
Optional Parameters 1247
Example 1248
172 SAVE: Save Database 1249
Syntax 1252
Optional Parameters 1252
Example 1254
173 SAVE FILES: Save Specified Files 1255
Syntax 1258
Optional Parameters 1258
Examples 1260
174 JCL Requirements and Examples 1261
Requirements to Run ADASAV JCL 1262
XIX ADASEL Utility: Select Protection Data 1269
175 Functional Overview 1271
Spanned Record Handling 1272
176 ADASEL Syntax 1275
TEST Parameter 1277
FDTINPUT Parameter 1277
SET GLOBALS Parameter 1280
SET TZ Parameter - Time Zone 1282
SET DST Parameter - Daylight Savings Indicator 1282
SELECT Parameter 1283
177 ADASEL Scenarios 1309
Selecting End Transaction (ET) Information 1310
Selecting Data by Command Sequence Number 1311
Selecting Data by Adabas Session Number 1312
Selecting Fields for an Output File 1313

Selecting Special Update Commands 1315
178 JCL Requirements and Examples 1319
Requirements to Run ADASEL JCL 1320
XX ADAULD Utility: Unload Files 1323
179 Functional Overview 1325
180 UNLOAD FILE: Unload Specified File 1327
Essential Parameter 1328
Optional Parameters and Subparameters 1328
Enhanced Temp and Sort Processing 1334
Examples 1334
181 ADAULD Input Processing 1337
Processing a Save Tape as Input 1338
182 ADAULD Output Processing 1341
183 ADAULD User Exit 9 1343
184 JCL Requirements and Examples 1345
Requirements to Run ADAULD JCL 1346
XXI ADAWRK Utility: Work Area Recovery Reports 1349
185 Functional Overview 1351
Replication-Related Data Processing 1352
ADAWRK EXU User Processing 1353
186 Utility Syntax 1355
187 Report Descriptions 1361
Environment Report 1362
Summary Report 1363
File Report 1366
Transaction Report 1370
Checkpoint Record Reporting 1377
Replication-Related Reporting 1378
188 JCL Requirements and Examples 1385
Requirements to Run ADAWRK JCL 1386
XXII ADAZAP Utility: Display or Modify Asso, Data, and Work Data Sets 1389
189 Functional Overview 1391
190 ADAZAP Syntax 1393
Essential Parameters 1394
Optional Parameters 1394
Examples 1395
191 JCL Requirements and Examples 1397
Requirements to Run ADAZAP JCL 1398
XXIII ADAZIN Utility: Print Adabas Maintenance and SVC Information 1401
192 Functional Overview 1403
Usage Notes and Processing 1404
193 ADAZIN Syntax 1407
Optional Parameters 1408
194 JCL Requirements and Examples 1411
Requirements to Run ADAZIN JCL 1412

195 Sample ADAZIN Report 1413
XXIV 1415
196 Adabas Sequential Files 1417
Sequential File Table 1418
z/OS Dependencies 1421
Enhancements to TEMP and SORT Processing in ADAINV, ADALOD and ADAULD 1422
197 Adabas Personnel Demo File 1425
Index 1427

Utilities

This documentation is organized in the following parts:

Conventions	Describes the control statement conventions, syntax conventions, control statement rules, and parameter values.
ADACDC Utility: Changed-Data Capture	Describes the ADACDC utility, which allows you to produce a file containing the delta of all changes made to the database over the period covered by the input protection logs. You can also use this utility to produce a file that lists each individual insert and delete transaction made to the database over the period covered by the input protection logs (without producing a delta of these changes).
ADACHK Utility: Adabas Common Check Processing	Describes the ADACHK utility, the Adabas Common Check Processing utility. This utility can be used to check the integrity of the entire database and it can be executed while concurrent updates are running against the file. The ADACHK utility can also be used to print or dump Associator blocks, command log blocks, Data Storage blocks, the File Control Block, the field definition table, the free space table, the general control blocks, the normal index, the upper index, protection log blocks, recovery log blocks, sort data set blocks, temporary data set blocks, the upper index, and work data set blocks.
ADACMP Utility: Compress-Decompress Data	Describes the ADACMP utility, which allows you to edit and compress data records that are to be loaded into the database. This utility also allows you to decompress individual files in the Adabas database.
ADACNV Utility: Database Conversion	Describes the ADACNV utility, which allows you to convert an Adabas database to a higher version of Adabas or to revert an Adabas database to a lower version of Adabas.
ADADBS Utility: Database Services	Describes the ADADBS utility, which allows you to perform many database definition and maintenance functions.
ADADEF Utility: Define a Database	Describes the ADADEF utility, which allows you to define a database and checkpoint file or define a new Work file. It also allows you to modify file encodings for the database.
ADADRU Utility: Database Storage Usage Reports	Describes the ADADRU utility, which allows you to produce summary and detail reports about database storage usage.
ADAFRM Utility: Format Adabas Data Sets	Describes the ADAFRM utility, which allows you to format Adabas database components.
ADAINV Utility: Inverted List Management	Describes the ADAINV utility, which allows you to create descriptors in a file and identify the descriptors used to couple two files.
ADALOD Utility: File Loader	Describes the ADALOD utility, which allows you to load a file into a database and to add or delete a large number of records (ISNs) to or from an existing file.

ADAMER Utility: ADAM Estimation	Describes the ADAMER utility, which allows you to produce statistics that indicate the number of Data Storage accesses required to find and read a record when using an ADAM descriptor.
ADAORD Utility: Reordering Functions	Describes the ADAORD utility, which allows you to reorder the Associator or Data Storage for a database or specified files. This utility also allows you to restructure a database or file.
ADAPLP Utility: Protection Log/Work Print	Describes the ADAPLP utility, which allows you to print data protection records contained on the Adabas Work data set or the Adabas data protection log.
ADARAI Utility: Adabas Recovery Aid	Describes the ADARAI utility, which allows you to manage recovery logging.
ADAREP Utility: Database Status Report	Describes the ADAREP utility, which allows you to produce the database status report.
ADARES Utility: Database Recovery	Describes the ADARES utility, which allows you to perform database recovery functions.
ADASAV Utility: Save/Restore Database or Files	Describes the ADASAV utility, which allows you to save or restore a database or specific database files.
ADASEL Utility: Select Protection Data	Describes the ADASEL utility, which allows you to select and decompress information in the Adabas sequential (SIBA) or dual/multiple (PLOG) protection log and write it to a print data set (DDDRUCK/ DRUCK) or a user-specified output data set.
ADAULD Utility: Unload Files	Describes the ADAULD utility, which allows you to unload an Adabas file.
ADAWRK Utility: Work Area Recovery Reports	Describes the ADAWRK utility, which allows you to produce reports from records in the autorestart area of Work part 1. This information can be used when the database autostart fails and the database will not come up. It can help you determine how to handle database recovery.
ADAZAP Utility: Display or Modify Asso, Data, and Work Data Sets	Describes the ADAZAP utility, which allows you to display (in hexadecimal format) and optionally change the contents of the Associator, Data Storage, or Work data sets.
ADAZIN Utility: Print Adabas Maintenance and SVC Information	Describes the ADAZIN utility, which allows you to print Adabas maintenance and SVC information.
ADABAS Sequential Files	Describes the sequential files used by the Adabas utilities as well as characteristics of file and device definitions by operating system.
Adabas Personnel Demo File	Provides the FDT for the Personnel demo file distributed with Adabas.

1 About this Documentation

- Document Conventions 2
- Online Information and Support 2
- Data Protection 3

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at **https://documentation.softwareag.com**.

Product Training

You can find helpful product training material on our Learning Portal at **https://learn.softwareag.com**.

Tech Community

You can collaborate with Software GmbH experts on our Tech Community website at **https://techcommunity.softwareag.com**. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software GmbH news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at **https://github.com/softwareag** and **https://hub.docker.com/publishers/softwareag** and discover additional Software GmbH resources.

Product Support

Support for Software GmbH products is provided to licensed customers via our Empower Portal at **https://empower.softwareag.com**. Many services on this portal require that you have an account. If you do not yet have one, you can request it at **https://empower.softwareag.com/register**. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

I Conventions

2 Conventions

- About this Documentation 8
- Control Statement Syntax 8
- Syntax Conventions 9
- Control Statement Rules 11
- Parameter Values 12

This chapter covers the following topics:

Notation *vrs*, *vr*, or *v*: When used in this documentation, the notation *vrs* or *vr* stands for the relevant version of a product. For further information on product versions, see *version* in the *Glossary*.

About this Documentation

Each Adabas utility is described in a separate part. For a single-function utility, the part begins with a syntax diagram showing the utility statement and all possible parameters. Parts for utilities with multiple functions begin with a brief overview of the functions, followed by the individual function syntax diagrams and descriptions.

Each function description contains:

- syntax diagram with all parameters;
- individual parameter descriptions describing coding rules, restrictions, and defaults; and
- utility function examples.

Following the function descriptions are job control examples for the z/OS operating system.

Note: Where an Adabas utility function applies to an add-on product only, see the relevant add-on product documentation for details.

Control Statement Syntax

Utility control statements have the following format:



where

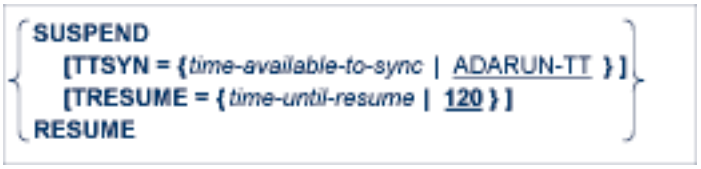
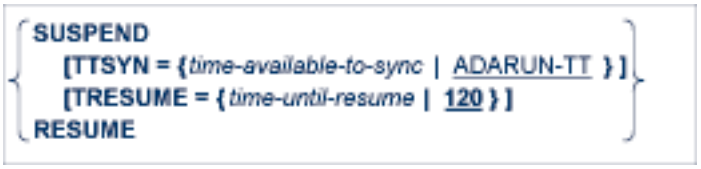
utility	is the name of the utility to be executed. Examples of utility names include: ADAORD ADADBS ADAINV
function	is the name of the specific utility operation to be executed. For example: ADAORD REORDATA ADADBS ADD ADAINV COUPLE Most single-function utilities (ADASEL, ADAULD, etc.) whose function is implicit have either no function value or an optional one.
parameter-list	is a list of parameters following the function. Parameters in the list are almost always keywords with the format: *parameter=value* A parameter may have one or more operands, and keyword parameters may be specified in any order. Most parameters require that you select or otherwise specify an operand value. Some operands are positional (value1 , value2 ,..., valuex), meaning that the values must be in a certain order as described in the text. All parameters must be separated by commas. In the statement syntax descriptions in this documentation, parameters are listed vertically (stacked) or are separated by vertical bars (|). Each list shows all possible parameters, from which one or more can (or must) be specified. Although parameters in the list must be separated by commas, these commas are omitted in the syntax statements when the parameters are stacked.

Syntax Conventions

The following table describes the conventions used in syntax diagrams of Adabas statements.

Convention	Description	Example
uppercase, bold	Syntax elements appearing in uppercase and bold font are Adabas keywords. When specified, these keywords must be entered exactly as shown.	**ADADBS CHANGE FILE =** *file-number* The syntax elements ADADBS, CHANGE, and FILE are Adabas keywords.

Convention	Description	Example
lowercase, italic, normal font	Syntax elements appearing in lowercase and normal, italic font identify items that you must supply.	**ADADBS CHANGE FILE =** *file-number* The syntax element *file-number* identifies and describes the kind of value you must supply. In this instance, you must supply the number of the file affected by the ADADBS CHANGE operation.
mixed case, normal font	Syntax elements appearing in mixed case and normal font (not bold or italic) identify items established by other Adabas control statements. This notation is usually used to identify how default values are determined for some parameters in Adabas syntax.	**[SORTDEV = {** *device-type* **|** ADARUN-device **}]** The syntax element "ADARUN-device" indicates that the device type identified by the ADARUN DEVICE parameter will be used if a different device type is not specified. The literal "ADARUN-device" should *not* be specified for the SORTDEV parameter.
underlining	Underlining is used for two purposes: 1. To identify default values, wherever appropriate. Otherwise, the defaults are explained in the accompanying parameter descriptions. 2. To identify the short form of a keyword.	**[LRECL = {***record-buffer-length* **|** 4000 **}]** In the example above, 4000 is the default that will be used for the LRECL parameter if no other record buffer length is specified. **DEVICE** In the example above, the short version of the DEVICE parameter is DE.
vertical bars (|)	Vertical bars are used to separate mutually exclusive choices. **Note:** In more complex syntax involving the use of large brackets or braces, mutually exclusive choices are stacked instead.	**ADAORD { RESTRUCTUREF | REF }** In the example above, you must select RESTRUCTUREF or REF for this ADAORD function. There are no defaults.
brackets ([ ])	Brackets are used to identify optional elements. When multiple elements are stacked or separated by vertical bars within brackets, only one of the elements may be supplied.	**SORTSEQ =** { *descriptor* **[,MU] [,NU]** / **ISN [, STARTISN =** *value***]** / physical-sequence } In this example, the SORTSEQ parameter and the MU, NU, and STARTISN subparameters are optional.

Convention	Description	Example
		Note: Note that the mutually exclusive choices for the SORTSEQ parameter are stacked.
braces ({ })	Braces are used to identify required elements. When multiple elements are stacked or separated by vertical bars within braces, one and only one of the elements must be supplied.	 In this example, either the SUSPEND or RESUME parameter is required.
indentation	Indentation is used to identify subparameters of a parameter.	 In this example, TTSYN and TRESUME are subparameters of the SUSPEND parameter.
ellipsis (...)	Ellipses are used to identify elements that can be repeated. If the term preceding the ellipsis is an expression enclosed in square brackets or braces, the ellipsis applies to the entire bracketed expression.	 In this example, the FIELD parameter can be repeated. In addition, more than one option can be associated with a field.
other punctuation and symbols	All other punctuation and symbols must be entered exactly as shown.	 In this example, the single quotation marks must be specified around the field definitions and their associated options. In addition, options must be separated by commas.

Control Statement Rules

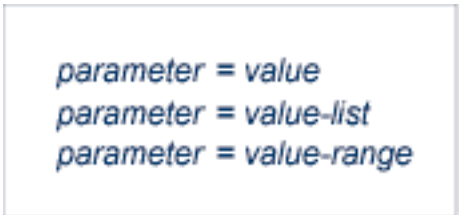
The following rules apply for the construction of utility control statements:

1. Each control statement must contain a utility name in positions one through six.
2. The utility function name follows the utility name, separated by at least one space.
3. Keyword parameter entries and multiple values within keyword entries must be separated by commas.
4. No space is permitted before or after an equals symbol (=).

5. The comma following the last parameter entry of a statement is optional.
6. Control statement processing ends with position 72 or when a space is encountered after the beginning of the parameter list. Entries made in positions 73-80 are not processed.
7. A statement that contains an asterisk (*) in position one is read as a comment and is not processed.
8. Control statements are continued by specifying the extra parameters on a new statement following (and separated by at least one space from) the utility name in positions one through 6.

Parameter Values

Variable values actually specified following the equals symbol (=) in parameters (represented by italicized labels in the preceding examples and elsewhere in this documentation) have the following syntax:



where *value* is as described in the following sections. Parameters *value-list* and *value-range* are variations of *value*, and are allowed either in place of or with *value*, depending on the individual parameter rules as described in the text.

value

The *value* parameter may consist of a number or a string of alphanumeric or hexadecimal characters. In some optional keyword parameters, a default value is assumed if the parameter is not specified.

Alphanumeric Values

Alphanumeric values are specified in one of the following ways:

If the value comprises . . .	Apostrophes around it are . . .
only upper- or lowercase letters, numeric digits and minus (-)	optional
any other characters including an apostrophe itself (which must be entered twice)	required

Numeric Values

Numeric values are specified as follows:

If the value represents . . .	Specify . . .
a number of either blocks or cylinders	the letter B must immediately follow the value if blocks are being specified; otherwise, cylinders are assumed: SIZE=200B (200 blocks) SIZE=200 (200 cylinders)
an Adabas file	a one- to four-digit number (leading zeros permitted): FILE=3 FILE=03 FILE=162
a device type	a four-digit number corresponding to the model number of the device type to be used: DEVICE=3390
a field name or descriptor	a two-character field name corresponding to the field name or descriptor: FIELD1=NA

Hexadecimal values are accepted if this is specified in the parameter description. Hexadecimal values must be within apostrophes following the indicator X:

```
X'0002DC9F'
```

value-list

value,... (numeric values)

```
BITRANGE=2,10,2
```

or

'value,...' (alphanumeric values)

```
UQDE='AA,AC,AE'
```

value-range

value - value, ...

```
ISN=600-900,1000-1200
```

Individual values within a value list or value range may be positional if they relate to values specified on corresponding parameters. For example:

```
ADADBS  UNCOUPLE  FILES=13,20,PASSWORD='PW13,PW20'
```

-instructs the ADADBS UNCOUPLE function to uncouple files 13 and 20, which are password-protected.

The passwords (specified by the PASSWORD parameter) must be in the same order as their corresponding files in the FILES parameter.

If file 13 is *not* password-protected, either the PASSWORD parameter must be specified with a placeholder comma as shown below

```
... PASSWORD=',PW20'
```

-to position the password "PW20" to the corresponding position of file 20 in the FILES value list, or FILES must specify file 20 first.

II ADACDC Utility: Changed-Data Capture

This section covers the following topics:

Functional Overview
Running the Utility
Operating System Considerations
The ADACDC User Exit
Examples
JCL Requirements and Examples

3 Functional Overview

- Phases of Operation and Resulting Files 19
- Primary Input Data 21
- Primary Output File 22
- Transaction File 23

The ADACDC utility takes, as input, one or more sequential protection logs and produces ultimately, as output, a *primary output file* containing the delta of all changes made to the database over the period covered by the input protection logs. *Delta of changes* means that the last change to each ISN in a file that was altered during this period appears on the primary output file.

Notes:

1. If the ADACDC ISN parameter is specified for the run, the delta is not produced. Instead, each individual delete and insert transaction is written to the primary output file and no delta of changes is produced.
2. Spanned records are supported by the ADACDC utility when the `SPANREC` parameter is specified. However, when the `IGNORESPANNED` parameter is specified in an ADACDC run, ADACDC processing ignores any spanned records, issues a warning message, and continues its processing. A return code of "4" is returned.
3. If a BT has been issued for a spanned record update, the primary record will occur before the associated secondary records on the output data set. This differs from the order in which information appears on the PLOG data set, where the secondary records occur before the primary record.
4. Date-time fields with the TZ option will be decompressed in UTC time (Coordinated Universal Time, also known as Greenwich Mean Time).

The ADACDC utility helps to extract data for usage outside of Adabas.

The data in the primary output file output from an ADACDC run may be used on a regular basis to extract delta data for use with other applications.

In order to run the ADACDC utility:

- an external sorter must be available and installed as the standard sorter in the operating system. See **Operating System Considerations** for more information.
- the ADACDC utility must have access to the database's Associator containing the FDTs of the files for which records are to be processed.

ADACDC uses this sort package to produce its output in ISN sequence, so all changes are written to the primary output file in ISN sequence.

Note: A logically deleted field cannot be decompressed in ADACDC utility runs. In other words, the output from the ADACDC run will not contain logically deleted fields.

Phases of Operation and Resulting Files

ADACDC processes sequential protection logs in two phases. You can execute phase 1 and phase 2 separately, or both at once (the default):

- If phase 2 is being run separately or both phases are being completed together, the data is decompressed and written to the *primary output file*.
- If only phase 1 is being executed, the data is written to an *extract file* . This extract file may then be processed multiple times by a phase 2 operation to decompress the records and write to primary output files.

The extract file contains data records in compressed format whereas the primary output file contains records in decompressed format. Refer to the section *ADACMP (Compress - Decompress)* in the *Adabas Utilities* documentation for more information about these formats.

The primary output file and the extract file are standard sequential files that can handle variable length records.

Phase 1 and the Extract File

During phase 1, updates from the protection logs are analyzed and prefixed with a standard structure called the CDCE. The format of each record on the file is a constant CDCE prefix followed by the compressed record information.

Usually, these records are passed to an external sort routine to establish the most recent update for each ISN on a file. Only the last change for a given file and ISN combination is written to the extract file. However, if the ISN parameter is specified for the ADACDC run, the updates are still sorted in ISN order, but they are not summarized. Instead, every change transaction for an ISN is recorded in the extract file.

The extract file created when phase 1 is run separately makes it possible to process the PLOG data once and then optionally produce multiple primary output files from it based, for example, on file selection criteria. The option is useful if different file changes are required for different purposes.

When the phase 1 process is being run, the extract file is opened for output. As records are output from the sort processing, the updates for each file and ISN combination is written to the extract file if:

- the update was performed by an ET user and belongs to a completed transaction; or
- the update was performed by an EXU user and belongs to a completed command; or
- NOET is specified.

All other updates for the file and ISN combination for that period are discarded if there are no controlled utility operations against that file (see *Checkpoints Written to the Primary Output File*).

Note: It is possible to have duplicate file and ISN combinations on the file if the ADACDC user exit (described later) adds records with file and ISN combinations that already exist. A record added or modified by the user exit is so marked in the CDCE structure.

Phase 2 or Both and the Primary Output File

The primary output file is used when both stages of ADACDC are run together, or for phase 2 processing only.

- If both phases are run together, the primary output file is opened and created directly using the output from the sort processing. In this case, processing occurs as for the extract file in phase 1 processing.
- If only phase 2 is run, the primary output file is created using input from the extract file.

The format of each record on the file is a constant CDCO prefix followed by the decompressed record information. If for some reason the record cannot be decompressed, a warning message is issued and the compressed record is written to the primary output file. A flag in the CDCO structure informs a user program when decompression for the record has failed.

Note: It is possible to have duplicate file and ISN combinations on the file if the ADACDC user exit (described later) adds records with file and ISN combinations that already exist. A record added or modified by the user exit is so marked in the CDCO structure.

Checkpoints Written to the Primary Output File

The primary objective of the ADACDC utility is to provide an output file containing the most recent summarized changes for each ISN in a file that has been modified for the period concerned. If the ISN parameter is specified for the run, the primary objective of the ADACDC utility is to provide an output file containing the changes for each ISN in a file that has been modified for the period concerned.

Apart from simple changes to a file, some utility operations executed against a file may fundamentally affect its contents. For example, if the file is deleted, simply providing the last updates for ISNs in the file does not accurately reflect the state of the file since all ISNs have been deleted.

For this reason, the following checkpoints are recorded and written to the primary output file as appropriate with the associated indication in the output record:

```
ADASAV RESTORE FILE          File created
ADAORD STORE FILE            File created
ADALOD LOAD FILE             File created
ADALOD UPDATE FILE           File updated
ADADBS DELETE FILE           File deleted
ADADBD REFRESH FILE          File deleted
```

Because these operations can fundamentally impact a file and its appearance, the checkpoint is written to the primary output file when it occurs relative to the other updates.

ADACDC retains the last change to all ISNs before each of the above checkpoints. This means that a file and ISN combination could appear multiple times on the primary output file if one or more checkpoints were written to it.

Primary Input Data

The primary input data comprises sequential protection logs produced either by the database directly or by the ADARES PLCOPY function. If there are multiple input protection logs, concatenate them.

Note: Spanned records are supported by the ADACDC utility when the `SPANREC` parameter is specified. However, when the `IGNORESPANNED` parameter is specified, ADACDC processing ignores any spanned records, issues a warning message, and continues its processing. When ending normally, the utility sets return code "4".

ADACDC processes this data to ensure that:

- when a new PLOG block is read and the PLOG number is the same, the PLOG block number is 1 greater than the previous PLOG block number.
- when the PLOG number itself changes, the new PLOG number is higher than the previous PLOG number and the new PLOG block number is 1.

Note: When the PLOG number changes and the difference between the PLOG numbers is greater than 1, a warning message is issued and processing continues as this can legitimately happen if online saves are used.

If any of these checks fail, the utility execution terminates.

Primary Output File

The primary output file is a sequential file comprising all database records that were added, updated, or deleted during the period covered by the input protection logs.

If a record was changed several times, only its last change appears in the output file; ADACDC employs a sort process to identify multiple changes to the same record. However, if the ISN parameter is specified for the ADACDC run, all changes for an ISN appear in the primary output file; ADACDC still employs the sort process to put the primary output file in ISN sequence.

Each primary output file record comprises a fixed-length record prefix followed by the database record in decompressed form. The decompressed data corresponds in format to the output of the **ADACMP DECOMPRESS** function.

The primary output record prefix is described by the CDCO DSECT. It has the following structure:

Bytes	Description	
0-1	record length (binary)	
2-3	set to zeros	
4-7	constant 'CDCO'	
8-9	database ID	
10-11	file number	
12-15	ISN of the updated record	
16-19	length of the decompressed data in bytes	
20-47	28-byte communication ID of the last user who updated the record	
48	change indicator:	
	X'04'	record added
	X'08'	record updated
	X'0C	record deleted
	X'10'	file created
	X'14'	file updated
	X'18'	file deleted or refreshed
49	flags (independent bit settings):	
	X'80'	record added by user exit
	X'40'	record modified by user exit
	X'20'	record still compressed; decompression failed
50	database version indicator	
51	reserved for future use	

Bytes	Description
52-59	4-byte STCK, followed by a 4-byte hexadecimal counter. Users can sort on this 8-byte field to put the primary output file records back into PLOG sequence, when necessary. Read ***Using ADACDC With ISNREUSE***, elsewhere in this section, for more information about when this might be necessary.
60-67	security system user ID. If the security system user ID is not known, blanks are stored in these bytes.
68-...	decompressed record data

When SPANREC is specified, the new spanned record CDCH and CDCN output headers are used for all CDCOUT output. DSECTs for the CDCH and CDCN headers can be found in the Adabas source library. These new spanned record headers will be used when the decompressed spanned records from the PLOG exceed the physical record length, requiring the creation of multiple physical records for a single logical record. In this case, the CDCH header will prefix every logical record written to CDCOUT, regardless of whether or not the record is spanned; subsequent physical records belonging to the same logical record will be prefixed by the CDCN header. ADACDC will copy the CDCH sort key to any subsequent CDCN records.



Note: In some cases no CDCN records may be produced. For example, if the input PLOG logical record is short enough to fit into one output physical record, only a CDCH record will be built.

Transaction File

To maintain input data checking over multiple runs of the utility, ADACDC stores information on the transaction file in a transaction control record containing the last database ID, the PLOG number, and the PLOG block number processed. This information is used to verify the latest input (unless the RESETTXF option is specified - see section ***RESETTXF : Reset Input Transaction File*** in ***ADACDC Optional Parameters***, elsewhere in this section).

ADACDC actually recognizes two different transaction files: input and output. Both transaction files are standard sequential files that can handle variable length records.

Input Transaction File Processing

During the input processing stage, ADACDC processes the input transaction file to the sort program.

Following the control record on the input transaction file, zero or more records may be found. These are database updates related to transactions not completed during the last run of the utility. These records are processed again as part of the input as their transactions will normally have been completed in the next sequential protection logs provided to the utility. This is the reason the sequence of protection logs is so important: updates may remain outstanding forever if the correct sequence is not maintained.

The transaction file also records whether the NOET option was specified during the last phase 1 run of the utility. When ADACDC detects that this option has changed from one utility execution to the next, it uses the information from the control record on the input transaction file; however, all transactional information in the other records is ignored. This is due to the fact that changing this option may cause inconsistent data to be written to the primary output file or extract file, as appropriate. ADACDC issues a warning when this happens.

Output Transaction File Processing

Once output processing from the sort program starts, the input transaction file is closed and the output transaction file is opened. The control record is written to the output transaction file followed by any updates that relate to incomplete transactions or, in the case where the NOET option is in effect or an EXU user is in control, to incomplete commands. The output transaction file is closed once processing is complete.

Using a Single Transaction File

It is possible to use the same file as both the input and output transaction file; however, if the utility fails while writing to the output transaction file (that is, at any time during the output processing of the sort utility), the input transaction file will no longer exist and therefore, rerunning the utility will yield a different result.

For this reason, the transaction file must be backed up prior to the utility run so that it can be restored in the event of a failure.

Alternatively, you could use a facility on your operating system (if available) that produces a new version of a file whenever a program updates the file.

4 Running the Utility

- Optional Parameters 26
- Using ADACDC With ISNREUSE 29

```
ADACDC  [FILES= filelist ]
        [IGNORESPANNED]
        [ISN]
        [MAXLOGRECLEN = { n | 1048576 } ]
        [NOET]
        [PHASE={ 1 | 2 | BOTH ]
        [RESETTXF]
        [SPANREC]
        [TZ='timezone']
        [DST]
```

The first time you run the ADACDC utility, use the following syntax and either do not specify or dummy the input transaction file (CDCTXI) to create a valid transaction file for input to subsequent ADACDC runs:

```
ADACDC RESETTXF,PHASE=BOTH
```

The RESETTXF option ignores the absent or dummied input transaction file, reads the primary input data, and produces primary output using the input data.

After the input transaction file has been created during the first run, only the utility name ADACDC is required to run this utility; the PHASE parameter defaults to BOTH. Parameter options are explained in the following sections.

Optional Parameters

DST: Daylight Savings Indicator

The DST parameter can be specified to indicate that date-time data includes a daylight savings time indicator. If a time zone uses daylight savings time, you must be sure to store and retrieve the daylight savings indicator with your date-time data or there will be no way to distinguish date-time values in the hour before the time is switched back to standard time. The two-byte daylight savings indicator directly follows the date-time value in uncompressed output and specifies the binary integer value of the daylight saving time offset from standard time in seconds.

If you specify the DST parameter for files containing date-time fields defined with option TZ, the field value is followed by the daylight savings time indicator in the uncompressed output, otherwise it is not. The default is that the date-time data is without a daylight savings time indicator in the uncompressed output.

The DST parameter requires that the **TZ** parameter be set in the same ADACDC run.

FILES: Files Processed

Use the FILES parameter to limit the file(s) processed by the utility and written to the output file:

- For phase 1 operation, only records relating to the files specified are written to the extract file.
- For phase 2 and BOTH operations, only records relating to the files specified are written to the primary output file.



Note: Clearly, files required for phase 2 processing must have been specified on the previous phase 1 operation that created the input extract file.

When this parameter is not specified, all files are processed by the utility.

IGNORESPANNED: Ignore Spanned Records

When the `IGNORESPANNED` parameter is specified in an ADACDC run, ADACDC processing ignores any spanned records, issues a warning message, and continues its processing. A return code of "4" is returned.

ISN: Record Delete and Insert Transactions Separately

Ordinarily, ADACDC processing consolidates all delete and insert transactions to the same ISN, creating a single update transaction for the ISN. However, if you specify the ISN parameter, each delete and insert transaction is recorded in the primary output file (CDCOUT) individually. So, when you use the ADACDC utility with the ISN parameter, the number of records produced in the primary output file will increase, possibly dramatically.

MAXLOGRECLEN: Uncompressed Buffer Size

Use this parameter to specify the size of the uncompressed record buffer allocated by the ADACDC utility for use in spanned record processing. The default value of MAXLOGRECLEN is 1048576 bytes (or 1MB). If the value specified for MAXLOGRECLEN is appended with the letter "K", it is multiplied by 1024. The minimum value is 32768 bytes.

NOET: Bypass ET Processing

ADACDC normally accepts for processing only those records that are part of completed transactions or, in the case of EXU users, records that are part of completed commands.

Use the NOET option to bypass this transaction processing when PHASE=1 or PHASE=BOTH. NOET has no effect when PHASE=2 because the input is the extract file from phase 1 which has already processed the protection log (PLOG) input with or without the NOET option.

When NOET is specified, any update made to the database is processed and written to the extract file (PHASE=1) or primary output file (PHASE=BOTH) as soon as it is encountered on the PLOG.



Caution: Specifying this option may result in updates recorded on the primary output file that are related to transactions that were not complete at the end of the input PLOG.

PHASE: Execution Phase

The PHASE parameter determines the input the utility requires and the output it produces:

- PHASE=1 reads the sequential PLOG input and produces an interim extract file for later processing by a phase 2 step.
- PHASE=2 reads an extract file produced by a previously executed phase 1 step and produces a primary output file containing the delta of changes made to the file.
- PHASE=BOTH (the default) reads the sequential PLOG input and produces the primary output file containing the delta of changes directly without reading or writing an extract file.

Refer to the section *Phases of Operation and Resulting Files*, elsewhere in this section, for more information.

RESETTXF: Reset Input Transaction File

ADACDC checks the primary input data to the utility to ensure that the PLOGs are read in sequence, by PLOG block and PLOG number. If these checks fail, the utility execution terminates.

To maintain the checks over multiple runs of the utility, ADACDC maintains input and output transaction files. These files also track record updates related to incomplete transactions or, in the case of EXU users, incomplete commands from one utility execution to the next. Normally, such incomplete transactions or commands are completed in the next sequential PLOGs provided to the utility.

However, if the need arises to process PLOGs out of sequence and the information in the transaction file can be safely removed, the RESETTXF option can be used to reset the transaction file so that the checks are bypassed and all outstanding transaction or command data is ignored for a given run. ADACDC ignores information on the input transaction file and writes the output transaction file at end of job.

Caution: If the sequence of PLOGs is interrupted, record updates related to incomplete transactions recorded in the transaction file may remain outstanding indefinitely.

SPANREC: Spanned Record Headers

When SPANREC is specified, the new spanned record CDCH and CDCN output headers are used for all CDCOUT output. DSECTs for the CDCH and CDCN headers can be found in the Adabas source library. These new spanned record headers will be used when the decompressed spanned records from the PLOG exceed that actual physical record limitation, requiring the creation of multiple physical records for a single logical record. In this case, the CDCH header will prefix every logical record written to CDCOUT, regardless of whether or not the record is spanned; subsequent physical records belonging to the same logical record will be prefixed by the CDCN header. ADACDC will copy the CDCH sort key to any subsequent CDCN records.

Notes:

1. In some cases no CDCN records may be produced. For example, if the input PLOG logical record is short enough to fit into one output physical record, only a CDCH record will be built.

2. If a BT has been issued for a spanned record update, the primary record will occur before the associated secondary records on the output data set. This differs from the order in which information appears on the PLOG data set, where the secondary records occur before the primary record.

TZ: Time Zone

The TZ parameter can be used to specify the local time zone. As records are decompressed, the date-time data is converted from UTC time (Coordinated Universal Time, also known as Greenwich Mean Time) to the corresponding time for the specified time zone. Date-time field data is always stored in UTC time in an Adabas file. Valid time zone names are listed in the TZINFO member of the Adabas source library and are specified in single quotes. The following is an example of a valid TZ specification:

```
TZ='America/New_York'
```



Note: Time zone names are case-sensitive.

This example sets the time zone to Eastern US. When decompressing input, local time is assumed to be Eastern US. When retrieving the stored UTC time data, it is converted to Eastern US time.

Adabas uses the time zone data taken from the tz database, which is also called the zoneinfo or Olson database. The specific list of time zone names that Adabas supports in any given release can be found in the TZINFO member of the Adabas source library (ADA*vrs*.SRCE). For more information about the TZINFO member of the time zone library, read *Supported Time Zones*.



Note: Review important information about the daylight savings time (**DST**) parameter. If a time zone uses daylight savings time, you must be sure to retrieve the daylight savings indicator with your date-time data or there will be no way to distinguish date-time values in the hour before the time is switched back to standard time.

Using ADACDC With ISNREUSE

Normal ADACDC processing produces a primary output file in ISN sequence. Ordinarily, this processing works fine. However, if the database file was created with the ADADBS or ADALOD ISNREUSE option specified, errors (response 98) can occur. These errors can occur because an ISN might have been reused, so multiple transactions may reside in the PLOG for the same unique descriptor key (UQ) with different ISNs.

To resolve these problems, the following steps should be taken:

1 Run the ADACDC utility with the ISN parameter specified. This will give you a granular list of changes in the primary output file, instead of attempting to summarize the changes by ISN. The data in the primary output file after this run will still be in ISN sequence.

2 Sort the primary output file (CDCOUT) in PLOG sequence prior to applying its data to your other application. This sort should be performed on the CDCOUT data at offset 52 for 8 bytes.

Once the data is sorted in PLOG sequence, the data can be applied to your other application.

Note: When spanned records are in being processed, the CDCH sort key is copied to any subsequent CDCN records. In addition, documented CDCO offsets are the same for the CDCH and CDCN output headers. However, with spanned records, multiple physical records may be required to form a single logical record, so this may affect how the data gets applied to your other application.

3 Sort the transaction file (CDCTXI/CDCTXO) back into PLOG sequence prior to running any additional ADACDC jobs. If you neglect to do this, future runs of ADACDC may be compromised.

To get the transaction file (CDCTXI/CDCTXO) in the correct sequence, two sorts are actually required, in the following sequence:

1. First run a sort that puts the CDCE records in PLOG sequence. This sort should be run on offset 68 for 4 bytes and then on offset 16 for 4 bytes. For example:

```
SORT FIELDS=(68,4,BI,A,16,4,BI,A),RECORD TYPE=V,LENGTH=32756
```

2. The second sort should sort the transaction file back into CDCC, CDCE, CDCX order. This sort should be run on offset 4 for 4 bytes.

When these two sorts have been run, the transaction file should be ready to be processed by future ADACDC jobs.

5 Operating System Considerations

For its sort requirements, the ADACDC utility uses a standard sort function installed in the operating system.

Note: Special sorts of the primary output file and the transaction file are required if ADACDC is to be run for a file for which ISNs can be reused (ADADBS or ADALOD ISNREUSE is specified). For complete information, read ***Using ADACDC With ISNREUSE***, elsewhere in this chapter.

No additional job steps are required by ADACDC when the sort function is invoked. However, depending on the amount of data to be sorted, the ADACDC job step may require additional sort-related DD statements for work files or for other sort-specific facilities. Refer to the sort documentation for more details.

Note: A sort package generally supplies summary information when a SYSOUT DD statement is specified.

When ADACDC invokes sort, it expects by default to transfer control to a load module named 'SORT'. If the sort module has a different name, you must reassemble and link the Adabas options module ADAOPD, specifying the name of the external sort program as follows:

1. Modify the OPDOS member, specifying the name of the sort program in parameter SORTPGM=.
2. Modify and run member ASMLOPD to assemble and link the module ADAOPD.

6 The ADACDC User Exit

- Installing the Exit 34
- User Exit Interface 34
- User Exit Calls 36

ADACDC calls a user exit at various points in its processing, providing you with the opportunity to intercede in that processing.

Note: The user exit may not update or add compressed spanned records. A flag in the CDCU DSECT will indicate if the input compressed record is spanned.

Installing the Exit

To install the user exit

1 Compile the user exit you wish ADACDC to use as module name ADACDCUX.

2 Make the module available to the ADACDC utility.

A sample user exit called ADACDCUX is provided on the source data set. The only function of the sample is to show you how to add, delete, and update records using the user exit interface.

Caution: ADACDCUX is a sample user program and is not supported under any maintenance contract agreement.

User Exit Interface

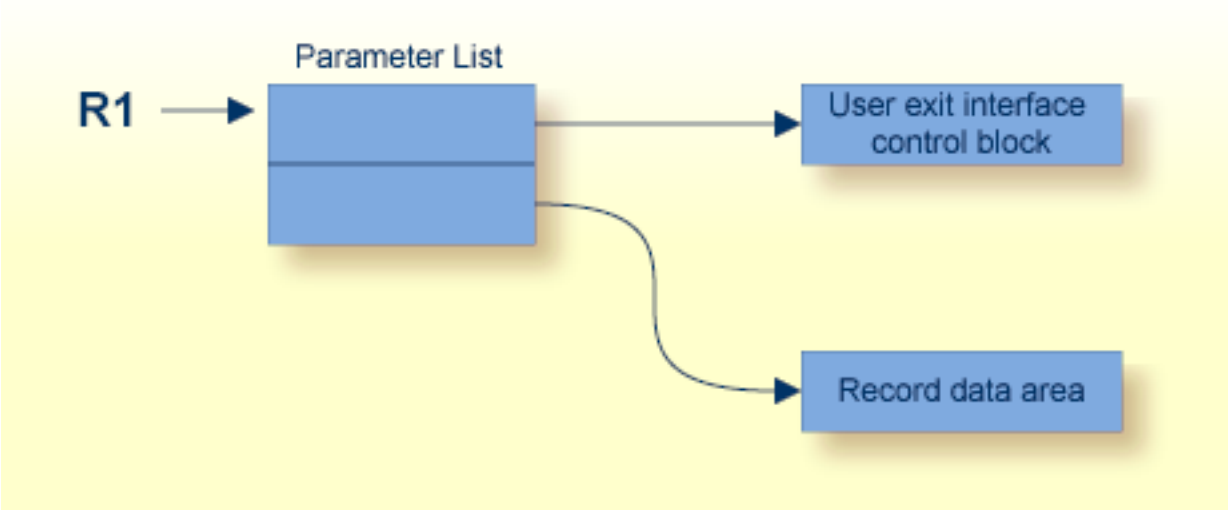
The user exit is called with the following registers set:

R1	user parameter list
R13	standard 72-byte register save area
R14	return address
R15	entry point

The user parameter list contains two pointers:

- the first to the ADACDC user exit parameter list mapped by the CDCU DSECT; and
- the second to the record area for the user exit where the record being processed is passed as appropriate.

The action to be performed is indicated in the CDCUFUNC field whereas the action the user exit directs ADACDC to take on return is indicated using the CDCURESP field.



The structure of the ADACDC user exit interface control block (CDCU DSECT) is as follows:

Bytes	Description	
0-3	constant 'CDCU'	
4-7	available for use by user exit	
8-11	length of record in second parameter	
12	function identifier:	
	X'00'	initialization
	X'04'	before pass to SORT input
	X'08'	before write to extract file
	X'0C'	before write to primary output file
	X'10'	termination
13	response code from user exit:	
	X'00'	normal processing
	X'04'	ignore this record
	X'08'	record has been updated
	X'0C'	insert new record
14-31	reserved for future use	

User Exit Calls

The following subsections describe the calls made to the user exit and their purpose.

Initialization Call (CDCUFUNC=CDCUINIT)

During initialization, ADACDC calls the user exit so that it can set up any areas it requires for future processing. The CDCUUSER field is provided in the CDCU for anchoring a user control block, if appropriate.

The record area pointer points to data that has no relevance for this call.

Termination Call (CDCUFUNC=CDCUTERM)

During termination, ADACDC calls the user exit so that it can close any open files or clean up any areas still outstanding after ADACDC execution. For example, if an anchor pointer was set in CDCUUSER, this area could be freed and the CDCUUSER field set to nulls.

The record area pointer points to data that has no relevance for this call.

SORT Input Call (CDCUFUNC=CDCUINPT)

ADACDC calls the user exit before a record is passed to the SORT routine as input.

The record area pointer points to the compressed data record to be returned prefixed by the CDCE control block.

The exit may elect to

- continue processing normally;
- request that the record be ignored;
- update the current record; or
- add a record, in which case the record pointed to by the record area pointer is passed to the SORT routine. Thereafter, each time the exit is called, the same record is presented again until
  - normal processing continues; or
  - the record is ignored or updated, at which time processing continues with the next input record.

Extract Output Call (CDCUFUNC=CDCUWRTE)

ADACDC calls the user exit before a record is written to the extract file during phase 1 processing. This exit point is *only* called during phase 1 processing and has no relevance in other cases.

The record area pointer points to compressed the data record to be written prefixed by the CDCE control block.

The exit may elect to

- continue processing normally;
- request that the record be ignored;
- update the current record; or
- add a record, in which case the record pointed to by the record area pointer on return is written to the extract file. Thereafter, each time the exit is called, the same record is presented again until
  - normal processing continues; or
  - the record is ignored or updated, at which time processing continues with the next record to be written to the extract file.

Primary Output Call (CDCUFUNC=CDCUWRTO)

ADACDC calls the user exit before a record is written to the primary output file. This exit point is *not* called during phase 1 processing and has no relevance in this case.

The record area pointer points to the decompressed data record to be written prefixed by the CDCO control block.

The exit may elect to

- continue processing normally;
- request that the record be ignored;
- update the current record; or
- add a record, in which case the record pointed to by the record area pointer on return is written to the primary output file. Thereafter, each time the exit is called, the same record is presented again until
  - normal processing continues; or
  - the record is ignored or updated, at which time processing continues with the next record to be written to the output file.

Updating or Adding Records

Consider the following points when updating or adding records from the exit:

- The CDCELEN/CDCERECL fields must correctly reflect the length of data following the CDCEDATA field for the input and write extract exit points.
- The CDCOLEN/CDCORECL fields must correctly reflect the length of data following the CDCODATA field for the input and write extract exit points.
- For the input exit point, records added should have a unique ISN if no ISN update is to be replaced.
- For the input exit point where an ISN is to be replaced, the last occurrence of the ISN should be updated or the replacement record for the ISN should be added after all occurrences for the ISN have been seen.

- When adding records at the extract or primary output exit points, be aware that if file and ISN combinations are duplicated, the duplicated information is written to the primary output file which may confuse processing routines for that file.

7 Examples

```
ADACDC RESETTXF,PHASE=BOTH
```

Ignoring any information on the input transaction file, reads the primary input data and produces primary output using the input data.

Use this syntax and either remove or dummy the input transaction file (CDCTXI) the first time you run the utility to create a valid transaction file for input to subsequent runs.

```
ADACDC PHASE=1
ADACDC FILES=20
ADACDC FILES=40-50
```

Reads the primary input data and processes data only for files 20 and 40 to 50 inclusive. The latest updates for each ISN on those files are written to the extract file.

```
ADACDC PHASE=2
ADACDC FILES=44-47
```

Reads a previously created extract file and writes all records for files 44, 45, 46, and 47 to the primary output file.

8 JCL Requirements and Examples

- Requirements to Run ADACDC JCL on z/OS 42

This section describes the job control information required to run ADACDC on z/OS and shows an example job stream.

Requirements to Run ADACDC JCL on z/OS

Data Set	DD Name	Storage	More Information
Associator	DDASSORn	disk	required to read the GCB and FDT entries
Protection log	DDSIIN	tape/ disk	sequential log (not required when PHASE=2)
Input transaction file	CDCTXI	tape/ disk	not required when RESETTXF is specified or when PHASE=2
Output transaction file	CDCTXO	tape/ disk	not required when PHASE=2
Extract file	CDCEXT	tape/ disk	output or input extract file (not required when PHASE=BOTH)
Primary output file	CDCOUT	tape/ disk	not required when PHASE=1
ADARUN parameters	DDCARD	reader	*Operations*
ADACDC parameters	DDKARTE	reader	*Utilities*
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADACDC messages	DDDRUCK	printer	*Messages and Codes*

ADACDC JCL Example

Refer to ADACDC in the JOBS data set for this example.

```
//ADACDC    JOB
//*
//*    ADACDC: CAPTURE DELTA CHANGES
//*
//CDC      EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD          <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1    <=== ASSO
//DDSIIN    DD   DSN=EXAMPLE.DByyyyy.PLOG(-3),DISP=SHR  <== PLOG TAPE
//           DD   DSN=EXAMPLE.DByyyyy.PLOG(-2),DISP=SHR  <== PLOG TAPE
//           DD   DSN=EXAMPLE.DByyyyy.PLOG(-1),DISP=SHR  <== PLOG TAPE
//           DD   DSN=EXAMPLE.DByyyyy.PLOG(0),DISP=SHR   <== PLOG TAPE
//CDCTXI    DD   DSN=EXAMPLE.input.trans.file,DISP=SHR
//CDCTXO    DD   DSN=EXAMPLE.output.trans.file,DISP=OLD
//CDCOUT    DD   DSN=EXAMPLE.output.primary.file,DISP=OLD
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
```

```
ADARUN PROG=ADACDC,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADACDC PHASE=BOTH,FILES=1,10,20-30
/*
```

III ADACHK Utility: Database Consistency Check Utility that Runs Concurrently with Normal Database Operations

This chapter covers the following topics:

Functional Overview
ACCHECK: Check Address Converter against Data Storage
ACSCAN: Scan the address converter to identify reusable ISNs
ASSOPRINT: Print/Dump Associator Blocks
CHECK: Perform all ADACHK Checks
CLOGPRINT: Print/Dump Command Log Blocks
DATAPRINT: Print/Dump Data Storage Blocks
DSCHECK: Check Data Storage
DSIMPRINT: Print Delta Save Image Data Set Blocks
DSSTPRINT: Print Data Storage Space Table Blocks
FCBPRINT: Print/Dump File Control Block (FCB)
FDTPRINT: Print/Dump Field Definition Table (FDT)
FSTPRINT: Print/Dump Free Space Table (FST)
GCBPRINT: Print/Dump General Control Blocks (GCBs)
ICHECK: Check Associator Physical Structure
NIPRINT: Print/Dump Normal Index
PLOGPRINT: Print/Dump Protection Log Blocks
PPTPRINT: Print/Dump Parallel Participant Table (PPT)
RLOGPRINT: Print/Dump Recovery Log
SORTPRINT: Print/Dump Sort Data Set
TEMPPRINT: Print/Dump Temporary Data Set
UIPRINT: Print/Dump Upper Index
VALIDATE: Validate All Database Files
WORKPRINT: Print/Dump Work Data Set

JCL Requirements and Examples

9 Functional Overview

- ADACHK Function Equivalences with Other Utility Functions 50
- Detailed Processing 52
- ADACHK Statistics 54
- Spanned Record Considerations 55
- Cipher File Considerations 56
- Syntax Checking with the TEST Parameter 56
- Sort Data Set Considerations 56
- SORTTYPE=EXTERNAL Operating System Considerations 57
- ADACHK Print Function Output Format 58
- Sample ADACHK Jobs 60

The ADACHK utility can be used to check the integrity of the entire database. All ADACHK utility functions can be run online with concurrent updates running.

You can use the ADACHK utility to:

- Perform all integrity checks and print functions previously available using the ADAPRI, ADAACK, ADAICK, ADADCK and ADAVAL utilities.
- Check the address converter against data storage (ADACHK ACCHECK utility function).
- Check Data Storage and the Data Storage space table (DSST) (ADACHK DSCHECK utility function).
- Check the physical structure of the index (ADACHK ICHECK utility function).
- Validate any or all files within an Adabas database, except the checkpoint and security files (VALIDATE utility function).
- Print or dump Associator blocks, command log blocks, Data Storage blocks, the File Control Block, the field definition table, the free space table, the general control block, the normal index, the upper index, protection log blocks, recovery log blocks, sort data set blocks, temporary data set blocks, and work data set blocks.
- Scan address converter for a file or files and identify the ranges of unused (reusable ISNs) with the option to specify RESET to set the rotating AC ISN pointer to the first, largest range identified.

In addition, the ADACHK CHECK utility function performs a combination of the ACCHECK, DSCHECK, ICHECK, and VALIDATE utility functions as well as the gap and overlap checks provided in the ADAREP utility. A table showing the expanded file chains is also provided when LAYOUT=LONG.

This functionality was provided in releases of Adabas prior to 8.3 only by running the ADAACK, ADAICK, ADADCK, ADAVAL ADAPRI, and ADAREP utilities. The ADACHK utility executes all of these functions in one function, ADACHK CHECK, and can be run while concurrent updates are being performed against the file.

You can run ADACHK against the entire database, just one file, or a range of files. Processing occurs on a file by file basis rather than by processing multiple files in parallel. Processing multiple files at once may hinder performance, particularly during the sort functions.

In a single ADACHK run, database-wide checks, such as checks for expanded files, gaps, and overlaps, are performed only once regardless of how many functions are specified in the run (once per invocation when using the multifunction option). Such checks involve reading all of the file control blocks (FCBs) and would incur too much overhead if they were done for each individual function in the run. Before Adabas 8.3 introduced the ADACHK CHECK utility function, different utilities (ADAACK, ADADCK, ADAICK, and ADAVAL) were used to check the integrity of the entire database. Consequently, several reads of each database block were necessary to check the whole database. With ADACHK CHECK, each block is processed only once (except for rare occasions) thus greatly improving the performance of the integrity checks. Multiblock I/Os are also used whenever possible, also enhancing performance.

Anytime ADACHK reads a block it verifies the consistency of the block on its own and reports any inconsistencies that it detects. ADACHK has been designed in such a manner that each record (Data Storage, address converter, index) is only read once. This is a substantial performance improvement from the existing check utilities. A further performance enhancement is that ADACHK functions (including ADACHK VALIDATE) can check the entire database on their own, without making nucleus calls (the ADAVAL utility must make nucleus calls when it runs). It only makes nucleus calls if it encounters an error that may be transient (caused by a concurrent update), in which case ADACHK must confirm whether the block in question is currently being updated.

The ADACHK utility does not require that its SORT data set be preformatted and it can dynamically increase the size of the SORT data set. If a single SORT data set (DDSORTR1) is in use, ADACHK can dynamically extend it if it fills up. If two SORT data sets are in use (DDSORTR1 and DDSORTR2), ADACHK can dynamically extend the DDSORTR2 data set if it fills up. In addition, instead of using its own, internal sorter, ADACHK can also be configured to use an external sorter -- that is, the sorter (if any) provided by your installation -- by specifying SORTTYPE=EXTERNAL. In this configuration, DDSORTR1 and/or DDSORTR2 need not be specified and any sorter-related specifications must be done for the external sorter.



Notes:

1. Bear in mind that the ADACHK utility can use large record buffer lengths when making nucleus calls to verify spanned Data Storage records or an index structure with many levels. If this is the case, the settings of your LU and NAB ADARUN parameters may need to be increased.
2. ADACHK does not require the Adabas nucleus to be active.
3. A pending autorestart condition is detected and reported since this can contribute to errors. Message CHK012W will typically be returned as soon as ADACHK detects the autorestart and the next error is encountered. Due to timing issues, it is possible that an error will be reported when we have not yet detected the pending autorestart and the CHK012I message will follow later. The nucleus status is only checked when an error is encountered since ADACHK does all the processing on its own and there is no need to call the nucleus unless there is an error.
4. ADACHK checks the nucleus status each time an error is encountered to determine whether the nucleus status has changed.
5. If the nucleus is restarted during the time that ADACHK is running, ADACHK will reestablish its communication with the nucleus when an error is detected and it attempts to confirm the error with the nucleus. Messages are written when ADACHK notices a nucleus status change.

The physical layout check of the entire database, including checks for overlaps, gaps, and the integrity of expanded file chains, is only performed for the check functions of ADACHK (ACCHECK, DSCHECK, ICHECK, CHECK, and VALIDATE). In addition, this check is performed only once per ADACHK execution, even if you are performing a multifunction run of the ADACHK utility running multiple check functions. Checking the physical layout of the database is an enhancement with ADACHK but produces a lot of overhead because this ADACHK processing requires that it read the FCB for every file in the database. Consequently, to maximize performance gains in ADACHK processing, the physical layout check is only performed once per ADACHK execution.

Therefore, if an error exists related to the physical layout of the database (gaps, overlaps, expanded file chains), this error is only seen and reported for the first check function in the multifunction run.

The following example shows where the physical layout check is performed in a multifunction execution of ADACHK using the following statements:

```
ADACHK NIPRINT FILE=1
ADACHK GCBPRINT
ADACHK ACCHECK <-----------Physical layout check is performed here before processing ↩
first file
```

Here is a second example showing where the physical layout check is performed if you are running a multifunction execution of ADACHK.

```
ADACHK ACCHECK FILE=1 <-----------Physical layout check is performed here ADACHK ↩
FCBPRINT FILE=2
ADACHK CHECK
ADACHK DSCHECK FILE=95
ADACHK CHECK FILE=20
```

This chapter covers the following topics:

ADACHK Function Equivalences with Other Utility Functions

The following table matches ADACHK utility functions with the equivalent functions previously available with the ADAPRI, ADAACK, ADAICK, ADADCK and ADAVAL utilities. ADACHK utility functions can be run online with concurrent updates running.



Note: The LAYOUT=SHORT (the default) parameter setting is used for each of these ADACHK functions to minimize output when all structures are correct. LAYOUT=LONG can be used if the additional output is desired.

ADACHK Utility Function	Equivalent Other Utility Function	Comments
ACCHECK	ADAACK ACCHECK ADAICK ACCHECK	ACCHECK can also be run online while concurrent updates are running. This utility function includes expanded file chain and gap/overlap checks similar to those provided by the ADAREP utility.
ACSCAN	none	---
ASSOPRINT	ADAICK ASSOPRINT ADAPRI ASSOPRI	The results from the ADACHK ASSOPRINT LAYOUT=LONG setting are what you get when you run the equivalent ADAICK or ADAPRI utility function.

ADACHK Utility Function	Equivalent Other Utility Function	Comments
CHECK	none	The CHECK utility function performs a combination of the ACCHECK, DSCHECK, ICHECK, and a VALIDATE utility functions as well as the gap and overlap checks provided in the ADAREP utility. CHECK can also be run online while concurrent updates are running.
CLOGPRINT	ADAPRI CLOGPRI	
DATAPRINT	ADAICK DATAPRINT ADAPRI DATAPRI	The results from the ADACHK DATAPRINT LAYOUT=LONG setting are what you get when you run the equivalent ADAICK or ADAPRI utility function.
DSCHECK	ADADCK DSCHECK ADAICK DSCHECK	DSCHECK can also be run online while concurrent updates are running. This utility function includes expanded file chain and gap/overlap checks similar to those provided by the ADAREP utility.
DSIMPRINT	ADAPRI DSIMPRI	
DSSTPRINT	none	---
FCBPRINT	ADAICK FCBPRINT ADAPRI	The results from the ADACHK FCBPRINT LAYOUT=LONG setting are what you get when you run the equivalent ADAICK or ADAPRI utility function.
FDTPRINT	ADAICK FDTPRINT ADAPRI	The results from the ADACHK FDTPRINT LAYOUT=LONG setting are what you get when you run the equivalent ADAICK or ADAPRI utility function.
FSTPRINT	none	---
GCBPRINT	ADAICK GCBPRINT ADAPRI	The results from the ADACHK GCBPRINT LAYOUT=LONG setting are what you get when you run the equivalent ADAICK or ADAPRI utility function.
ICHECK	ADAICK ICHECK	ICHECK can also be run online while concurrent updates are running. This utility function includes expanded file chain and gap/overlap checks similar to those provided by the ADAREP utility.
NIPRINT	ADAICK NIPRINT ADAPRI	The results from the ADACHK NIPRINT LAYOUT=LONG setting are what you get when you run the equivalent ADAICK or ADAPRI utility function.
PLOGPRINT	ADAPRI PLOGPRI	
PPTPRINT	ADAICK PPTPRINT ADAPRI	The results from the ADACHK PPTPRINT LAYOUT=LONG setting are what you get when you run the equivalent ADAICK or ADAPRI utility function.
RLOGPRINT	ADAPRI RLOGPRI	
SORTPRINT	ADAPRI SORTPRI	
TEMPPRINT	ADAPRI TEMPPRI	

ADACHK Utility Function	Equivalent Other Utility Function	Comments
UIPRINT	ADAICK UIPRINT ADAPRI	The results from the ADACHK UIPRINT LAYOUT=LONG setting are what you get when you run the equivalent ADAICK or ADAPRI utility function.
VALIDATE	ADAVAL VALIDATE	VALIDATE can also be run online while concurrent updates are running. This utility function includes expanded file chain and gap/overlap checks similar to those provided by the ADAREP utility.
WORKPRINT	ADAPRI WORKPRI	

Detailed Processing

Anytime ADACHK reads a block, it verifies the consistency of the block on its own and reports any inconsistencies that it detects. Data structures involving more than one block are more challenging to check in an efficient manner.

Each block (Data Storage, address converter, indices) is only read once in an ADACHK run. As Data Storage blocks are read, depending on the function being requested, parts of each record are extracted and fed into the sorter for comparison with data from other blocks at a later point in time. The information from Data Storage is extracted first. Therefore, when an inconsistency with the address converter, index, or DSST is discovered, the entire Data Storage record is no longer available to ADACHK; only the portion of the record that was extracted for sort purposes is available. The information in the sorter may not always reflect the current status if there have been concurrent updates. For example, any records that have been added by users since the start of the ADACHK run will not be in the sorted output. Similarly, any items that were deleted by users since the start of the ADACHK run will be in the sorted records but will not be contained in the database. When this happens, verification calls to the nucleus are made. If ADACHK is not able to ascertain for sure if there is an inconsistency, the nucleus is called to verify if the error is real or transient. When multiple blocks are involved and parallel updates are occurring, before ADACHK can report an error as real, a call is made to the nucleus for a fresh copy of the blocks involved to verify if the error is real or transient. This allows concurrent updating to take place on the files while the consistency checks are running.

In more detail, the following is a high-level overview of the processing that occurs for each ADACHK function:

ADACHK Function	Processing
ACCHECK	1. Read through each Data Storage record, extracting the ISN and the Data Storage RABN. Feed the ISN and RABN into the sorter. 2. If a Data Storage record is a spanned record, assemble the spanned Data Storage record. Then extract the secondary ISNs and Data Storage RABNs and feed them into sorter. 3. Read through the primary and secondary address converters and compare them against the sorted records from Data Storage. 4. Report errors.
ACSCAN	Scan address converter for reusable ISNs.
DSCHECK	1. Read through each Data Storage record, extracting the Data Storage RABN and DSST value from data. Feed them into the sorter. 2. Read through the DSST and compare it against the sorted records from Data Storage. 3. Report errors.
ICHECK	1. Read through the index structure. 2. Check the index structure.
VALIDATE	1. Read through each Data Storage record, extracting the descriptors, descriptor values, and the ISN. Feed them into the sorter. 2. Call the ADACHK ICHECK utility function. 3. Read the NI structure ,comparing it against the sorted descriptor values. 4. Report errors.
CHECK	Processing for CHECK is a combination of processing for the ACCHECK, DSCHECK, ICHECK and VALIDATE utility functions.

The sort input phase runs first in all functions, except the ADACHK ICHECK utility function, where the sorter is not invoked. This allows ADACHK to read each component only once. Sorting all extracted data in one large operation also eliminates the need to have a temporary data set to store the data. It does however slightly increase the likelihood of concurrent updates not captured in the sorted data. Therefore, in the event that ADACHK encounters an error, it must determine whether the error may be due to concurrent updates and, if necessary, check with the nucleus.

With the previously supported Adabas integrity checking utilities (ADAACK, ADAICK, ADADCK and ADAVAL), several reads of each block were necessary to check the integrity of the entire database. With ADACHK CHECK, each block is processed only once (except for rare occasions), thus greatly improving the performance of integrity checks. Multiblock I/Os are also used whenever possible to enhance performance. ADACHK functions (including ADACHK VALIDATE) can also check the entire database on their own, without making nucleus calls. This is accomplished by reading compressed records from Data Storage and creating a DVT that is validated against each corresponding value in the indices. ADACHK only makes nucleus calls if it encounters an

error that could be due to concurrent updates, in which case it must confirm whether the block in question is currently being updated.

ADACHK Statistics

The following are statistics normally printed at the end of each function, one set per database file.

```
<ADACHK> Nucleus Statistics - Error Verifications
Number of transient errors = nn
Number of nucleus verification calls  = nn
Number of nucleus retries = nn
Number of AC blocks returned from nucleus = nn
Number of Index blocks returned from nucleus = nn
Number of Data blocks returned from nucleus = nn
Number of nucleus status changes = nn
Number of Buffer flushes = nn
```

If the REPAIR option of the ADACHK DSCHECK utility function is run, the following additional statistics are printed:

```
Number of repairs attempted = nn
Number of repairs completed = nn ↩
```

The statistics provide information on how much interaction took place with the nucleus during the ADACHK processing. These can help aid in performance tuning if heavy updating is occurring at the same time ADACHK is running. These same statistics can be displayed at any point during ADACHK processing by issuing the DSTAT command against the ADACHK job. In addition to these statistics, information is displayed on ADACHK processing progress, including identifying the file ADACHK is currently processing. The statistics displayed for the operator command DSTAT are the same as these ADACHK statistics with the exception that only statistics for the file currently being processed are displayed along with the function in progress.

If NOSYNC, UTYPE=EXU or UTYPE= EXF are specified in an ADACHK run, no nucleus calls are made and the statistics are suppressed.

The following table describes the statistics in more detail:

Statistics	Description
Number of transient errors	A *transient error* is an error that ADACHK detected and determined was due to a concurrent update in progress. A transient error can be determined in some cases without the help of the nucleus by rereading the block in question. When this number is higher than the number of nucleus verification calls, you know that ADACHK resolved one or more transient errors on its own. Every error encountered goes through the transient error verification process. Even if the nucleus was down when ADACHK started, the nucleus

Statistics	Description
	could have been restarted and therefore ADACHK attempts to call the nucleus unless instructed by parameters that indicate not to do so (UTYPE, NOSYNC, and MAXCALLS parameters).
Number of nucleus verification calls	The number of calls made from ADACHK to the nucleus to verify if an error was real or transient.
Number of nucleus retries	The number of retries the nucleus performed due to concurrent updates.
Number of AC blocks returned from nucleus	The number of address converter blocks returned from the nucleus to ADACHK. ADACHK stores these newly read blocks in its storage in an effort to reduce further nucleus calls.
Number of Index blocks returned from nucleus	The number of index blocks returned from the nucleus to ADACHK. ADACHK stores these newly read blocks in its storage in an effort to reduce further nucleus calls. ADACHK and the nucleus track which blocks have previously been returned and, if ADACHK already has the same copy of the block and it has not been updated, the nucleus will not return the block again in an effort to maximize performance.
Number of Data blocks returned from nucleus	The number of Data Storage blocks returned from the nucleus to ADACHK. ADACHK stores these newly read blocks in its storage in an effort to reduce further nucleus calls.
Number of nucleus status changes	The number of times the nucleus changed its status (for example, from active to inactive).
Number of Buffer flushes	The number of buffer flushes requested by ADACHK.
Number of repairs attempted	If the REPAIR option is specified for the ADACHK DSCHECK utility function, this statistic shows the number of repairs attempted for the run.
Number of repairs completed	If the REPAIR option is specified for the ADACHK DSCHECK utility function, this statistic shows the number of repairs completed for the run.

Spanned Record Considerations

ADACHK can use large record buffer lengths when making nucleus calls to verify spanned Data Storage records. If this is the case, you may need to increase the settings of your LU and NAB ADARUN parameters.

In addition, bear in mind that integrity checks are done on spanned Data Storage RABNs as they are being assembled and also later, depending on what ADACHK function is running. Therefore, you may see a particular error message more than once during ADACHK processing. For example, it is possible, when running the ADACHK CHECK function, that the ACCHECK portion of the CHECK run may identify an error that is also identified in the DSCHECK portion of the CHECK run.

Cipher File Considerations

If your database contains cipher files, you cannot run the ADACHK CHECK or ADACHK DSCHECK utility functions on the entire database in a single run. In addition, you can only run these functions on a single cipher file in one run and on the other (non-cipher) files by listing them specifically in the FILE parameter of another run.

Be sure to include the following parameters:

- Specify the cipher key in the CODE parameter.
- Specify a single cipher file in the FILE parameter.

If you do not specify both of these parameters, errors will result.



Note: The TEST parameter does not check the types of files in the database. This means that the TEST run will not show any ADACHK errors related to cipher files in the run; these errors will only become apparent when you perform the ADACHK run without the TEST parameter.

Syntax Checking with the TEST Parameter

The ADACHK functions include a syntax-checking-only mode. When the TEST parameter is specified, the ADACHK function syntax is checked, but the function is not performed.



Note: The TEST parameter does not check the types of files in the database. This means that the TEST run will not show any ADACHK errors related to cipher files in the run; these errors will only become apparent when you perform the ADACHK run without the TEST parameter. This is particularly pertinent for the ADACHK CHECK and DSCHECK utility functions.

```
ADACHK ACCHECK TEST ↩
```

Sort Data Set Considerations

The sort data sets (DDSORTR1/DDSORTR2) need only be specified in a run when:

- The ADACHK SORTPRINT utility function is run; or
- The ADACHK ACCHECK, CHECK, DSCHECK, and VALIDATE utility functions are run with parameter SORTTYPE=INTERNAL specified (this is the default) and when the amount of data to be sorted exceeds the space declared by the LWP parameter in these utility function runs.

- To perform sorting when the amount of data to be sorted exceeds the space declared by the LWP parameter and the internal sorter (SORTTYPE=INTERNAL) is being used, one or two sort data sets defined in the ADACHK job may be used as temporary storage.
- The VALIDATE function determines the length of their sort keys by the longest descriptor in the file or MAXDESCLEN plus the static part of the sort key (10 bytes). The number of sort records is determined by the number of values for the descriptors.
- For DSCHECK and ACCHECK, the number of SORT records is based off of the number of records in the file and the length of these keys is shorter (10 bytes).
- CHECK is a combination of all the functions so of course uses a combination of all the SORT records.
- The sort data set(s) should be at least as large as the combined ASSO and DATA extents of the largest file to be processed by ADACHK.
- The sort data sets are denoted by their DD names DDSORTR1 and DDSORTR2 in the ADACHK job. There is no need to format the sort data sets using the Adabas ADAFRM utility; even temporary files may be used.
- The sort data sets will initially be allocated in the size of the primary allocator. When this space is exhausted, the last sort data set will be extended by the size of the secondary allocator. Whether an extension is possible depends on the availability of secondary space.
- The following sample z/OS job statement defines a sort data set that will initially be allocated as 100 cylinders and that can be extended in multiples of 500 additional cylinders.

```
//DDSORTR1 DD DSN=&&MYSORTR1,DISP=NEW,SPACE=(CYL,(100,500))
```

SORTTYPE=EXTERNAL Operating System Considerations

When the ADACHK parameter SORTTYPE=EXTERNAL is specified, the ADACHK utility uses a standard sort function installed in the operating system. The following additional considerations should be taken into account when running on z/OS.

No additional job steps are required by ADACHK when the sort function is invoked. However, depending on the amount of data to be sorted, the ADACHK job step may require additional sort-related DD statements for work files or for other sort-specific facilities. Refer to your system's sort documentation for more details.



Note: A sort package generally supplies summary information when a SYSOUT DD statement is specified.

When ADACHK invokes sort, by default it expects to transfer control to a load module named "SORT". If the sort module has a different name, you must reassemble and link the Adabas options module ADAOPD, specifying the name of the external sort program as follows:

1. Modify the OPDOS member, specifying the name of the sort program in parameter SORTPGM.
2. Modify and run member ASMLOPD to assemble and link the module ADAOPD.

ADACHK Print Function Output Format

The output produced by the ADACHK print functions (**ASSOPRINT**, **CLOGPRINT**, **DATAPRINT**, **DSIMPRINT**, **DSSTPRINT**, **FCBPRINT**, **FDTPRINT**, **FSTPRINT**, **GCBPRINT**, **NIPRINT**, **PLOGPRINT**, **PPTPRINT**, **RLOGPRINT**, **SORTPRINT**, **TEMPPRINT**, **UIPRINT**, and **WORKPRINT**) has three sections:

1. Section 1 is a general section, showing the ADACHK print function settings and the RABN ranges of the ASSO, DATA, WORK, and DSST sections of the database.
2. Section 2 shows GCB information for each RABN in the ADACHK run.
3. Section 3 shows the dumped blocks for each RABN in the ADACHK run.

The following example depicts these sections in a very simple (only one RABN) ADACHK ASSOPRINT run.

```
ADACHK ASSOPRINT RABN=1,LAYOUT=LONG


 CHK498I, ******* ASSOPRINT started *******     2014-09-17 21:55:41

 DB   00199
             Version 83 data base
                  First      Last
                   RABN      RABN
            ASSO 00000001  00006966  (    1 -  26982)
                 00006967  00018882  (26983 - 100482)
            DATA 00000001  0000AFBE  (    1 -  44990)
            WORK 00000001  0000696F  (    1 -  26991)
            DSST 00000427  00000438  ( 1063 -   1080)


*****************************************************************************
**
  ASSOPRINT function: From RABN 1 (x'00000001')
                    : To   RABN 1 (x'00000001')

*****************************************************************************
**

 CHK001I, ASSO block 1 (x'00000001') - GCB
 DB   00199 GCB  +0000              Database number: 00C7
 DB   00199 GCB  +0002      Maximum number of files: 00C8
 DB   00199 GCB  +0004     Date/time of DB creation: CD99C977
 DB   00199 GCB  +0008               First FCB RABN: 0000001F
 DB   00199 GCB  +000C                   RABN size: 00000003
 DB   00199 GCB  +0010                Database name: LEH199
 DB   00199 GCB  +0020          ADABAS version code: CB
 DB   00199 GCB  +0021              ADABAS SM level: 01
 DB   00199 GCB  +0024                Status byte 1: 40
 DB   00199 GCB  +0025                Status byte 2: 00
 DB   00199 GCB  +0028       Checkpoint file number: 0013
 DB   00199 GCB  +002A         Security file number: 0000
 DB   00199 GCB  +002C          Trigger file number: 0000
 DB   00199 GCB  +002E   Reptor definitions file nr: 0000
 DB   00199 GCB  +0030     Reptor SLOG file number: 0000
 DB   00199 GCB  +0032                 ATM file 1: 0000
 DB   00199 GCB  +0034                 ATM file 2: 0000
 DB   00199 GCB  +0050          Highest file loaded: 0064
 DB   00199 GCB  +0052     Last used session number: 0001
 DB   00199 GCB  +0054       PLOG number of session: 0000
 DB   00199 GCB  +0058          Highest used PETLID: 00000000
 DB   00199 GCB  +005C               First PPT RABN: 00000407
 DB   00199 GCB  +0060                Last PPT RABN: 00000426
 DB   00199 GCB  +0064              First DLOG RABN: 00000000
 DB   00199 GCB  +0068               Last DLOG RABN: 00000000
 DB   00199 GCB  +006C             RLOG device type: 0000
 DB   00199 GCB  +0070       Default alpha encoding: 00000000
 DB   00199 GCB  +0074        Default wide encoding: 00000000
 DB   00199 GCB  +0078 Default alpha encoding ASCII: 00000000
 DB   00199 GCB  +007C   Default wide encoding user: 00000000
 DB   00199 GCB  +0080               First WORK RABN: 00000001
 DB   00199 GCB  +0084                Last WORK RABN: 0000696F
 DB   00199 GCB  +0088             WORK device type: 3390
 DB   00199 GCB  +008A              WORK identifier: W
 DB   00199 GCB  +008B     WORK constant set number: 14
 DB   00199 GCB  +0090           First PET data RABN: 00000000
 DB   00199 GCB  +0094            Last PET data RABN: 00000000
 DB   00199 GCB  +0098     Total PET data area size: 00000000
 DB   00199 GCB  +009C          PET data block size: 00000000
 DB   00199 GCB  +00A0         User index table size: 00000000

 CHK001I, ASSO block 1 (x'00000001') - GCB
 0000  00C700C8 CD99C977  0000001F 00000003  * G H?rI?   .   .*
 0010  D3C5C8F1 F9F94040  40404040 40404040  *LEH199          *
 0020  CB010000 40000000  00130000 00000000  *?.       .      *
 0030  00000000 00000000  00000000 00000000  *                *
       SAME
 0050  00640001 00000000  00000000 00000407  * ? .           ..*
 0060  00000426 00000000  00000000 00000000  *  ..             *
 0070  00000000 00000000  00000000 00000000  *                 *
 0080  00000001 0000696F  3390E614 00000000  *    .  ??.? W.    *
 0090  00000000 00000000  00000000 00000000  *                 *
       SAME
 01F0  00000000 00000000  00000000           *            *


 CHK499I, ******* ASSOPRINT ended *******     2014-09-17 21:55:41     CC = 00
```

Section 1: General Section

Section 2: RABN Section

Section 3: Dumped Blocks Section

Depending on the setting of the LAYOUT parameter, two or more sections are printed.

- If you specify LAYOUT=SHORT, sections 1 and 2 are printed.
- If you specify LAYOUT=MEDIUM, sections 1 and 3 are printed.
- If you specify LAYOUT=LONG, all three sections are printed.

Sample ADACHK Jobs

The following sample ADACHK jobs are provided with your Adabas installation in the ADA*vrs*.JOBS library. You will need to tailor the jobs to direct them to your database and database files before you run them:

Sample Job Name	Description
ADACHKC	This sample job checks the entire database for every possible ADACHK function. This is the equivalent of running the ADACHK ACCHECK, DSCHECK, ICHECK, and VALIDATE functions on every file. In addition, the entire database is checked for gaps and overlaps.
ADACHKP	This sample job runs multiple ADACHK print functions, including GCBPRINT, PPTPRINT, ASSOPRINT and DATAPRINT showing an example of one RABN or multiple RABNs.
ADACHKS	This sample job executes the ADACHK CHECK function using sort data sets.
ADACHKV	This sample job runs multiple ADACHK print and check functions, as follows: ■ ADACHK ICHECK: The ICHECK utility function is run for all database files. ■ ADACHK DSCHECK FILE=1,2: The DSCHECK utility function is run for files 1 and 2. ■ ADACHK DSCHECK FILE=17: The DSCHECK utility function is run for file 17. ■ ADACHK VALIDATE FILE=3: The VALIDATE utility function is run for file 3. ■ ADACHK ACCHECK: The ACCHECK utility function is run for all files. ■ ADACHK CHECK FILE=21: The CHECK utility function is run for file 21. ■ ADACHK FSTPRINT: The FSTPRINT utility function is run for the database. ■ ADACHK FCBPRINT FILE=1: The FCBPRINT utility function is run for file 1.

10 ACCHECK: Check Address Converter against Data Storage

- Essential Parameters 62
- Optional Parameters and Subparameters 63
- Examples 65

The ADACHK ACCHECK function can be used to check the address converter for a specific file or range of files or for a specific ISN or range of ISNs. It can run while concurrent updates are running. If spanned records are used, ACCHECK assumes any ISNs passed to it are primary ISNs and performs its processing accordingly. The ranges can encompass all files or all ISNs.

If the file being checked has spanned records enabled, the secondary address converter used to map the secondary ISNs to the RABNs of the secondary records is automatically checked as well.

ACCHECK checks each address converter element to determine whether the Data Storage RABN is within the used portion of the Data Storage extents specified in the file control block (FCB).

ACCHECK checks the ISN for each record in each Data Storage block (within the specified ISN range) to ensure that the address converter element for that ISN contains the correct Data Storage RABN.

The syntax of the ADACHK ACCHECK utility function is:

```
ADACHK  ACCHECK  [ABEND34]
                 [ERRLIM = { error-threshold-count | 100 } ]
                 [FILE= { file | file1 - filex | all-files } ]
                 [ISN= { isn | isn1 - isnx | all-ISNs } ]
                 [LAYOUT = { SHORT | MEDIUM | LONG } ]
                 [LWP = { workpoolsize | 10240K } ]
                 [MAXCALLS = { count | [ 200 | ( TOPISN-value/100 ) ] } ]
                 [NOSYNC]
                 [NOUSERABEND]
                 [SORTTYPE = { INTERNAL | EXTERNAL } ]
                 [TEST]
                 [UTYPE = { ACC | EXF | EXU } ]
```

This chapter describes the syntax and parameters of the ACCHECK function.

Essential Parameters

There are no required parameters for this utility function.

Optional Parameters and Subparameters

The following optional parameters can be specified.

ABEND34: Change User Abend 35 to 34

This optional parameter can be used to change a user abend 35 to user abend 34 if a utility error occurs. This ensures that a dump is produced when the utility terminates abnormally.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

ERRLIM: Error Threshold

The maximum number of errors that this ADACHK utility function will tolerate before terminating. Valid values are any positive integer <=5000. If no valid value is specified for this parameter, a default of "100" is used.

If a value less than 0 or greater than 5000 is specified, the following error will result and the default will be used:

```
CHK413E, ERROR: Parameter ERRLIM is incorrect.
```

FILE: Files To Be Checked

The file or single range of files to be checked. If no file or range of files are specified, all files in the database are checked.

ISN: ISN Range To Be Checked

The ISN or range of ISNs to be checked. If no ISN or range of ISNs are specified, the entire range of ISNs (from MINISN through TOPISN) is checked.

LAYOUT: Report Detail Level

The level of data produced for the report. Valid values are described in the following table:

Valid Values	Description
SHORT	Specify this value to produce the minimum output. This is the default.
MEDIUM	Specify this value to produce medium-level output (several interpreted lines).
LONG	Specify this value to produce extensively-detailed output.

LWP: Sort Work Pool Size

LWP specifies the size of the work pool to be used for sorting, if SORTTYPE=INTERNAL. It cannot be specified with SORTTYPE=EXTERNAL. The LWP value can be specified in bytes or kilobytes (value followed by a "K"). If no value is specified, the default is 10485760 bytes (or 10240K).

If ADARUN parameter V64BIT=YES has been specified, the work pool is allocated in 64-bit addressable storage.

The minimum value for LWP is 102,400 (100K). The maximum value is limited by how much 31-bit or 64-bit addressable virtual storage ADACHK can obtain from the system.

When the amount of data to be sorted exceeds the space declared by the LWP parameter, one or two sort data sets defined in the ADACHK job may be used as temporary storage. For more information, read ***Sort Data Set Considerations***, elsewhere in this chapter. The sort data set(s) should be at least as large as the combined ASSO and DATA extents of the largest file to be processed by ADACHK.

If a sort data set is provided and large enough, the default value of 10 megabytes is sufficient for checking Adabas files of about three gigabytes. For checking larger files, LWP should be increased. As a rule of thumb, an LWP of size `n*10` MB supports files sizes of about `n*n*3` GB. For example, LWP=102400K (100 megabytes = `10*10` MB) is sufficient for processing files of about 300 gigabytes (= `10*10*3` GB).

It is generally recommended you make LWP larger than the required minimum to reduce the number of I/Os for sorting and thus to improve the performance of ADACHK.

MAXCALLS: Maximum Number of Nucleus Calls

The maximum number of nucleus calls allowed per file. Once this limit is hit, no more nucleus calls can be made and a warning message that the limit has been reached is printed. Valid values are any positive integer, with no upper limit.

The default for MAXCALLS is either the value "200" or the result of the TOPISN setting divided by 100 (TOPISN/100), whichever is greater.

NOSYNC: Transient Error Confirmation

Note: Use of this parameter is not recommended and is only intended for users who want to limit calls to their nucleus.

Use this parameter to control whether transient errors are confirmed via the active nucleus. If this parameter is specified, no nucleus calls are made and the errors reported may be due to concurrent updates. A warning message is generated informing you.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

SORTTYPE: Sort Type Used

The type of sort to use. Valid values are "INTERNAL" (the default) and "EXTERNAL".

Value	Meaning
EXTERNAL	An external sort facility will be used to sort the data. For information about operating system considerations related to this option, read *SORTTYPE=EXTERNAL Operating System Considerations*, elsewhere in this section.
INTERNAL	An internal sort facility will be used to sort the data. This is the default. For information about the use of sort data sets with this setting, read *Sort Data Set Considerations*, elsewhere in this chapter.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See *Syntax Checking with the TEST Parameter* for more information about using the TEST parameter in ADACHK functions.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

UTYPE: User Type

The user type in effect for the ADACHK run. Valid values are:

Value	Identifies
ACC	An access-only user. This is the default. When ACC is specified, parallel updates are allowed on the file by normal users (but not by other utility processes).
EXF	An exclusive file control user. When EXF is specified, only the ADACHK process can use the file; no other users can read or write to the file in parallel with the ADACHK run.
EXU	An exclusive control user. When EXU is specified, parallel reads are allowed on the file by normal users. This option might provide better performance if there are a lot of concurrent updates ongoing.

Examples

The following example checks the address converter for all files in the database. It also checks all related structures in the database for consistency.

```
ADACHK ACCHECK
```

The following example checks ISNs 100-200 for file 2. The default INMEMORY sort is used.

```
ADACHK ACCHECK FILE=2,ISN=100-200
```

11 ACSCAN: Scan the address converter to identify reusable ISNs

- Essential parameters 69
- Optional Parameters and Subparameters 69

The ADACHK ACSCAN function can be used to scan the AC (using multi block I/O) looking for reusable ISNs. The user can specify one file, multiple files, or a range of files and ADACHK will

1. Read through the AC for the file using multi block I/O.
2. Identify the first and largest range of reusable ISNs per file.
3. Call the nucleus to reset the rotating ISN pointer if the parameter RESET is specified.

As with the other ADACHK functions, if multiple files are specified ADACHK will process one file at a time.

ACSCAN will not be a “CHECK” function and therefore will not check the physical layout of the entire database, which includes checks for overlaps, gaps, and the integrity of expanded file chains as ACCHECK, DSCHECK, ICHECK, CHECK, and VALIDATE do currently. This also means that when ADACHK CHECK is specified, ACSCAN will not be included.

ACSCAN can also be run online while concurrent updates are running or when the nucleus is inactive. The nucleus must be active for the RESET to take effect but if the nucleus is down and the ACSCAN with RESET is requested, the function will still print the recommended rotating ISN and also print an informational message that the nucleus must be up to RESET to rotating ISN pointer.

The only interaction with the nucleus will be to issue an ACC open at start (allowing normal file activity but no UTI/UTL utilities), close at the completion, set a DIB entry, and send the RESET ISN call to the nucleus if RESET is specified. No interactive error validations are necessary or will with occur with this function.

Only the primary rotating ISN will be affected by this function. There is no need at this point to scan the secondary AC or to reset the pointer to a specific secondary ISN.

The syntax of the ADACHK ACSCAN utility function is:

```
ADACHK ACSCAN FILE = file-number | file1-filex
                                   [RESET]
                                   [ABEND34]
                                   [ERRLIM={error-threshold-count|100}]
                                   [LAYOUT={SHORT|MEDIUM|LONG}|
                                   [NOUSERABEND]
                                   [TEST]
```

This document describes the syntax and parameters of the ACSCAN function.

Essential parameters

FILE: Files To Be Checked

The file or single range of files to be checked.

Optional Parameters and Subparameters

The following optional parameters can be specified.

- RESET:
- ABEND34: Change User Abend 35 to 34
- ERRLIM: Error Threshold
- LAYOUT: Report Detail Level
- NOUSERABEND: Termination without Abend
- TEST: Test Syntax
- Examples

RESET:

The `RESET` parameter causes the rotating ISN pointer in the FCB to point to the optimal AC block where reusable ISNs exists as identified by the ACSCAN function. The nucleus must be active for this to be effective.

The RESET parameter cannot be issued against a file which other operations or utilities have requested exclusive usage.

ABEND34: Change User Abend 35 to 34

This optional parameter can be used to change a user abend 35 to user abend 34 if a utility error occurs. This ensures that a dump is produced when the utility terminates abnormally.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

ERRLIM: Error Threshold

The maximum number of errors that this ADACHK utility function will tolerate before terminating. Valid values are any positive integer <=5000. If no valid value is specified for this parameter, a default of “100” is used.

If a value less than 0 or greater than 5000 is specified, the following error will result and the default will be used:

```
CHK413E, ERROR: Parameter ERRLIM is incorrect.
```

LAYOUT: Report Detail Level

The level of data produced for the report. Valid values are described in the following table:

Valid Values	Description
SHORT	Specify this value to produce the minimum output. This is the default.
MEDIUM	Provides same output as LAYOUT=SHORT
LONG	Specify this value to produce extensively-detailed output.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will not abend after printing the error message. Instead, the message “utility TERMINATED DUE TO ERROR CONDITION” is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

TEST: Test Syntax

The `TEST` parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables cannot be tested: only the syntax of the specified parameters can be tested. See **Syntax Checking with the TEST Parameter** for more information about using the `TEST` parameter in ADACHK functions.

Examples

The following example scans the address converter for file 1 in the database identifying the reusable ISNs. Since RESET is specified, the rotating AC pointer will be set to the first largest range of re-usable ISNs.

```
ADACHK ACSCAN FILE=1,LAYOUT=LONG,RESET
```

Sample output:

```
*****************************************************************

 ACSCAN function for File 1

*****************************************************************


CHK025I, The Adabas nucleus is currently INACTIVE

Allocated AC Extents:

           From RABN          = x'00000A7C' (2684)
           To   RABN          = x'00000A7D' (2685)
           Highest ISN        = x'000004F7' (1271)
           ISNs per Block     =     x'027C' (636)

Unused ISN-ISN          AC RABN      Unused ISNs      Used ISNs
---------------------------------------------------------------
00000001-0000027B  2684 (x'00000A7C')          0            636
0000027C-000004F7  2685 (x'00000A7D')        164            472
---------------------------------------------------------------
Totals                                       164           1108

Total unused ISNs:                           164
Total used ISNs  :                                        1108
Rotating ISN in AC will be set to 636 (x'0000027C')
```

12 ASSOPRINT: Print/Dump Associator Blocks

- Essential Parameters 74
- Optional Parameters and Subparameters 74
- Example 76

The ADACHK ASSOPRINT function can be used to print the contents of any Associator (ASSO) block in the database while concurrent updates are running. If ADACHK can determine the type of information stored in the block (for example, NI or UI), it produces a formatted output.

```
ADACHK ASSOPRINT RABN = { rabn | rabn1 - rabn2 } [ ,{ rabn | rabn1 - rabn2 } ]...
                 [ABEND34]
                 [DEVICE = device ]
                 [ERRLIM = { error-threshold-count | 100 } ]
                 [LAYOUT = { SHORT | MEDIUM | LONG } ]
                 [NOUSERABEND]
                 [TEST]
```

This chapter describes the syntax, parameters, and output of the ASSOPRINT function.

Essential Parameters

RABN: RABNs to Be Printed or Dumped
The RABNs or ranges of RABNs to be printed or dumped.

Optional Parameters and Subparameters

ABEND34: Change User Abend 35 to 34
This optional parameter can be used to change a user abend 35 to user abend 34 if a utility error occurs. This ensures that a dump is produced when the utility terminates abnormally.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

DEVICE: Device Type
The device type that contains the data set to be printed. This parameter is required if the device type is different from the standard device type assigned by the ADARUN DEVICE parameter.

ERRLIM: Error Threshold
The maximum number of errors that this ADACHK utility function will tolerate before terminating. Valid values are any positive integer <=5000. If no valid value is specified for this parameter, a default of "100" is used.

If a value less than 0 or greater than 5000 is specified, the following error will result, and the default value will be used:

```
CHK413E, ERROR: Parameter ERRLIM is incorrect.
```

LAYOUT: Report Detail Level

The level of data produced for the report or dump.Valid values are described in the following table. For more information about the output sections described below, read *ADACHK Print Function Output Format*, elsewhere in this section.

Valid Values	Description
SHORT	Specify this value to produce the minimum output, printing sections 1 and 2 of the possible output. This is the default.
MEDIUM	Specify this value to produce medium-level output, printing sections 1 and 3 of the possible output.
LONG	Specify this value to produce extensively-detailed output, printing all three sections of the possible output. **Note:** The results from the LAYOUT=LONG setting are what you get when you run the equivalent ADAICK utility function.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See *Syntax Checking with the TEST Parameter* for more information about using the TEST parameter in ADACHK functions.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

Example

In the following example, Associator RABNs 1, 2, 30-31, and 2000 are printed.

```
ADACHK ASSOPRINT RABN=1,2,30-31,2000
```

13 CHECK: Perform all ADACHK Checks

- Essential Parameters 79
- Optional Parameters and Subparameters 79
- Example 83

The ADACHK CHECK function can be used to perform all ADACHK utility check functions while concurrent updates are running. This includes the **ACCHECK**, **DSCHECK**, **ICHECK**, and **VALIDATE** functions.

Before Adabas 8.3 introduced the ADACHK CHECK utility function, different utilities (ADAACK, ADADCK, ADAICK, and ADAVAL) were used to check the integrity of the entire database. Consequently, several reads of each database block were necessary to check the whole database. With ADACHK CHECK, each block is processed only once (except for rare occasions) thus greatly improving the performance of the integrity checks. Multiblock I/Os are also used whenever possible, also enhancing performance.

ADACHK CHECK can be performed on the file level, a range of files, or the entire database. It is the default function if ADACHK is run with no specified function.

The syntax of the ADACHK CHECK utility function is:

```
ADACHK  CHECK     [ABEND34]
                  [CODE = cipherkey ]
                  [DESCRIPTOR = { desc1 [ ,desc2 ] ... | all-descriptors } ]
                  [ERRLIM = { error-threshold-count | 100 } ]
                  [FILE= { file | file1 - filex | all-files } ]
                  [LAYOUT = { SHORT | MEDIUM | LONG } ]
                  [LWP = { workpoolsize | 10240K } ]
                  [MAXCALLS = { count | [ 200 | ( TOPISN-value/100 ) ] } ]
                  [MAXDESCLEN = length | FDT-largest-descriptor ]
                  [NOSYNC]
                  [NOUSERABEND]
                  [NOVALIDATE]
                  [SORTTYPE = { INTERNAL | EXTERNAL } ]
                  [TEST]
                  [UTYPE = { ACC | EXF | EXU } ]
```

This chapter describes the syntax and parameters of the CHECK function.

Essential Parameters

There are no required parameters for this utility function.

Optional Parameters and Subparameters

The following optional parameters can be specified.

ABEND34: Change User Abend 35 to 34

This optional parameter can be used to change a user abend 35 to user abend 34 if a utility error occurs. This ensures that a dump is produced when the utility terminates abnormally.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

CODE: Cipher Code

The CODE parameter is required if the file or file(s) being processed by the ADACHK utility are enciphered. Specify the required cipher key.

If the data in a file processed by the ADACHK CHECK run is in ciphered form, the cipher key must be specified using the CODE parameter. In addition, you must also include the cipher file in the FILE parameter. Only one cipher file can be specified in the run. For more information, read *Cipher File Considerations*, earlier in this section.

For more information on the use of ciphering in Adabas databases, contact your technical support representative for more information about Adabas Security documentation.

DESCRIPTOR: List of Descriptors

The DESCRIPTOR parameter restricts ADACHK processing to one descriptor field, providing a way to limit the run in cases in which the Associator is very large or there is a need to evaluate a specific descriptor. If DESCRIPTOR is not specified, ADACHK validates all qualifying descriptor fields.

The following is an example of DESCRIPTOR use:

```
ADACHK CHECK FILE=4,DESCRIPTOR='AA,BB'
```

ERRLIM: Error Threshold

The maximum number of errors that this ADACHK utility function will tolerate before terminating. Valid values are any positive integer <=5000. If no valid value is specified for this parameter, a default of "100" is used.

If a value less than 0 or greater than 5000 is specified, the following error will result, and the default value will be used:

```
CHK413E, ERROR: Parameter ERRLIM is incorrect.
```

FILE: Files To Be Checked

The file or single range of files to be checked. If no file or range of files are specified, all files in the database are checked.

If the file listed in the FILE parameter is enciphered, the CODE parameter is also required. You must specify the cipher key in the CODE parameter. For more information, read *Cipher File Considerations*, earlier in this section.

ISN: ISN Range To Be Checked

The ISN or range of ISNs to be checked. If no ISN or range of ISNs are specified, the entire range of ISNs (from MINISN through TOPISN) is checked.

LAYOUT: Report Detail Level

The level of data produced for the report. Valid values are described in the following table:

Valid Values	Description
SHORT	Specify this value to produce the minimum output. This is the default.
MEDIUM	Specify this value to produce medium-level output (several interpreted lines).
LONG	Specify this value to produce extensively-detailed output.

LWP: Sort Work Pool Size

LWP specifies the size of the work pool to be used for sorting, if SORTTYPE=INTERNAL. It cannot be specified with SORTTYPE=EXTERNAL. The LWP value can be specified in bytes or kilobytes (value followed by a "K"). If no value is specified, the default is 10485760 bytes (or 10240K).

If ADARUN parameter V64BIT=YES has been specified, the work pool is allocated in 64-bit addressable storage.

The minimum value for LWP is 102,400 (100K). The maximum value is limited by how much 31-bit or 64-bit addressable virtual storage ADACHK can obtain from the system.

When the amount of data to be sorted exceeds the space declared by the LWP parameter, one or two sort data sets defined in the ADACHK job may be used as temporary storage. For more information, read *Sort Data Set Considerations*, elsewhere in this chapter. The sort data set(s) should be at least as large as the combined ASSO and DATA extents of the largest file to be processed by ADACHK.

If a sort data set is provided and large enough, the default value of 10 megabytes is sufficient for checking Adabas files of about three gigabytes. For checking larger files, LWP should be increased. As a rule of thumb, an LWP of size `n*10` MB supports files sizes of about `n*n*3` GB. For example, LWP=102400K (100 megabytes = `10*10` MB) is sufficient for processing files of about 300 gigabytes (= `10*10*3` GB).

It is generally recommended you make LWP larger than the required minimum to reduce the number of I/Os for sorting and thus to improve the performance of ADACHK.

MAXCALLS: Maximum Number of Nucleus Calls

The maximum number of nucleus calls allowed per file. Once this limit is hit, no more nucleus calls can be made and a warning message that the limit has been reached is printed. Valid values are any positive integer, with no upper limit.

The default for MAXCALLS is either the value "200" or the result of the TOPISN setting divided by 100 (TOPISN/100), whichever is greater.

MAXDESCLEN: Maximum Length of Descriptor Value

Use the MAXDESCLEN parameter to specify the maximum length of a descriptor value present in the file during compressed record extraction. If the MAXDESCLEN parameter is not specified, the default is to extract the largest default descriptor length from the FDT.

NOSYNC: Transient Error Confirmation



Note: Use of this parameter is not recommended and is only intended for users who want to limit calls to their nucleus.

Use this parameter to control whether transient errors are confirmed via the active nucleus. If this parameter is specified, no nucleus calls are made and the errors reported may be due to concurrent updates. A warning message is generated informing you.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

NOVALIDATE: No Validation

Specify the NOVALIDATE parameter when you want to run the ADACHK utility without validation (the VALIDATE function).

SORTTYPE: Sort Type Used

The type of sort to use. Valid values are "INTERNAL" (the default) and "EXTERNAL".

Value	Meaning
EXTERNAL	An external sort facility will be used to sort the data. For information about operating system considerations related to this option, read *SORTTYPE=EXTERNAL Operating System Considerations*, elsewhere in this section.
INTERNAL	An internal sort facility will be used to sort the data. This is the default. For information about the use of sort data sets with this setting, read *Sort Data Set Considerations*, elsewhere in this chapter.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See *Syntax Checking with the TEST Parameter* for more information about using the TEST parameter in ADACHK functions.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

Note: If your database contains cipher files, note that the TEST parameter does not check the types of files in the database. This means that the TEST run will not show any ADACHK errors related to cipher files in the run; these errors will only become apparent when you perform the ADACHK run without the TEST parameter. For more information, read *Cipher File Considerations*, earlier in this section.

UTYPE: User Type

The user type in effect for the ADACHK run. Valid values are:

Value	Identifies
ACC	An access-only user. This is the default. When ACC is specified, parallel updates are allowed on the file by normal users (but not by other utility processes).
EXF	An exclusive file control user. When EXF is specified, only the ADACHK process can use the file; no other users can read or write to the file in parallel with the ADACHK run.
EXU	An exclusive control user. When EXU is specified, parallel reads are allowed on the file by normal users. This option might provide better performance if there are a lot of concurrent updates ongoing.

Example

The following example runs the ACCHECK, DSCHECK, ICHECK, and VALIDATE functions of ADACHK:

```
ADACHK CHECK
```

14 CLOGPRINT: Print/Dump Command Log Blocks

- Essential Parameters 86
- Optional Parameters and Subparameters 86
- Example 88

The ADACHK CLOGPRINT function can be used to print the contents of any block or range of blocks in the command log (CLOG) data set while concurrent updates are running.

```
ADACHK CLOGPRINT RABN = { rabn | rabn1 - rabn2 } [ ,{ rabn | rabn1 - rabn2 } ]...
                 [ABEND34]
                 [DEVICE = device ]
                 [ERRLIM = { error-threshold-count | 100 } ]
                 [LAYOUT = { SHORT | MEDIUM LONG } ]
                 [NOUSERABEND]
                 [NUMBER = { number | 1 } ]
                 [TEST]
```

This chapter describes the syntax and parameters of the CLOGPRINT function.

Essential Parameters

RABN: RABNs to Be Processed

The RABNs or ranges of RABNs to be printed or dumped.

Optional Parameters and Subparameters

ABEND34: Change User Abend 35 to 34

This optional parameter can be used to change a user abend 35 to user abend 34 if a utility error occurs. This ensures that a dump is produced when the utility terminates abnormally.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

DEVICE: Device Type

The device type that contains the data set to be printed. This parameter is required if the device type is different from the standard device type assigned by the ADARUN DEVICE parameter.

ERRLIM: Error Threshold

The maximum number of errors that this ADACHK utility function will tolerate before terminating. Valid values are any positive integer <=5000. If no valid value is specified for this parameter, a default of "100" is used.

If a value less than 0 or greater than 5000 is specified, the following error will result, and the default value will be used:

```
CHK413E, ERROR: Parameter ERRLIM is incorrect.
```

NUMBER: Command Log Data Set Number

The number of the multiple (two through eight) command log (CLOG) data set from which the blocks are to be printed. If NUMBER is not specified, the blocks are taken from DDCLOGR1 (the default).

LAYOUT: Report Detail Level

The level of data produced for the report or dump.Valid values are described in the following table. For more information about the output sections described below, read *ADACHK Print Function Output Format*, elsewhere in this section.

Valid Values	Description
SHORT	Specify this value to produce the minimum output, printing sections 1 and 2 of the possible output. This is the default.
MEDIUM	Specify this value to produce medium-level output, printing sections 1 and 3 of the possible output.
LONG	Specify this value to produce extensively-detailed output, printing all three sections of the possible output. **Note:** The results from the LAYOUT=LONG setting are what you get when you run the equivalent ADAICK utility function.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See *Syntax Checking with the TEST Parameter* for more information about using the TEST parameter in ADACHK functions.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

Example

In the following example, command log RABNs 1, 2, 30-31, and 2000 are printed from DDCLOG1.

```
ADACHK CLOGPRINT RABN=1,2,30-31,2000
```

15 DATAPRINT: Print/Dump Data Storage Blocks

- Essential Parameters 90
- Optional Parameters and Subparameters 90
- Example 92

The ADACHK DATAPRINT function can be used to print the contents of any block or range of blocks in Data Storage while concurrent updates are running.

```
ADACHK DATAPRINT  RABN = { rabn | rabn1 - rabn2 } [ ,{ rabn | rabn1 - rabn2 } ]...
                  [ABEND34]
                  [DEVICE = device ]
                  [ERRLIM = { error-threshold-count | 100 } ]
                  [LAYOUT = { SHORT | MEDIUM | LONG } ]
                  [NOUSERABEND]
                  [TEST]
```

This chapter describes the syntax and parameters of the DATAPRINT function.

Essential Parameters

RABN: RABNs to Be Processed

The RABNs or ranges of RABNs to be printed or dumped.

Optional Parameters and Subparameters

ABEND34: Change User Abend 35 to 34

This optional parameter can be used to change a user abend 35 to user abend 34 if a utility error occurs. This ensures that a dump is produced when the utility terminates abnormally.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

DEVICE: Device Type

The device type that contains the data set to be printed. This parameter is required if the device type is different from the standard device type assigned by the ADARUN DEVICE parameter.

ERRLIM: Error Threshold

The maximum number of errors that this ADACHK utility function will tolerate before terminating. Valid values are any positive integer <=5000. If no valid value is specified for this parameter, a default of "100" is used.

If a value less than 0 or greater than 5000 is specified, the following error will result, and the default value will be used:

```
CHK413E, ERROR: Parameter ERRLIM is incorrect.
```

LAYOUT: Report Detail Level

The level of data produced for the report or dump.Valid values are described in the following table. For more information about the output sections described below, read *ADACHK Print Function Output Format*, elsewhere in this section.

Valid Values	Description
SHORT	Specify this value to produce the minimum output, printing sections 1 and 2 of the possible output. This is the default.
MEDIUM	Specify this value to produce medium-level output, printing sections 1 and 3 of the possible output.
LONG	Specify this value to produce extensively-detailed output, printing all three sections of the possible output. **Note:** The results from the LAYOUT=LONG setting are what you get when you run the equivalent ADAICK utility function.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See *Syntax Checking with the TEST Parameter* for more information about using the TEST parameter in ADACHK functions.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

Example

In the following example, Data Storage RABNs 6, 7, 8, 9, 2222, and 5000 are printed.

```
ADACHK DATAPRINT RABN=6-9,5000,2222
```

16 DSCHECK: Check Data Storage

- Essential Parameters 95
- Optional Parameters and Subparameters 95
- Examples 100

Use the ADACHK DSCHECK utility function to check Data Storage and the Data Storage space table (DSST) of a specific file (or files) in the database. It can be run while concurrent updates are running. ADACHK reads each used Data Storage block (according to the Data Storage extents in the file control block) and performs consistency checks. These include verifying:

- That the block length is within the permitted range.
- That the sum of the length of all records in the Data Storage block plus four is equal to the block length.
- Whether any record exists with a record length greater than the maximum compressed record length for the file or with a length of zero.
- Whether there are any duplicate ISNs within a single block.
- That the ISN headers are valid if spanned records are used. In addition, checks are performed to ensure that each header contains the ISN of the primary record in the chain and the ISN of the next spanned record in the chain.
- That the primary and secondary spanned records are correctly identified, if spanned records are used.
- Whether the associated DSST element contains the correct value. If not, a repair of the DSST is necessary.
- The compressed record and printing it.
- Large object (LB) field value reference structures in a base file's compressed record. The relationship of an LB field value reference to the LB field value stored in the associated large object (LOB) file is *not* verified. LOB and base files are processed independently.

The syntax of the ADACHK DSCHECK utility function is:

```
ADACHK  DSCHECK  [ABEND34]
                 [CODE = cipherkey ]
                 [COMPREC]
                 [ERRLIM = { error-threshold-count | 100 } ]
                 [FILE= { file | file1 - filex | all-files } ]
                 [FROMRABN = from-rabn ]
                 [ISN= { isn | isn1 - isnx | all-ISNs } ]
                 [LAYOUT = { SHORT | MEDIUM | LONG } ]
                 [LWP = { workpoolsize | 10240K } ]
                 [MAXCALLS = { count | [ 200 | ( TOPISN-value/100 ) ] } ]
                 [NOSYNC]
                 [NOUSERABEND]
                 [REPAIR]
                 [SORTTYPE = { INTERNAL | EXTERNAL } ]
                 [TEST]
                 [TORABN = to-rabn ]
                 [USAGE]
                 [UTYPE = { ACC | EXF | EXU } ]
```

This chapter describes the syntax and parameters of the DSCHECK function.

Essential Parameters

There are no required parameters for this utility function.

Optional Parameters and Subparameters

The following optional parameters can be specified.

ABEND34: Change User Abend 35 to 34

This optional parameter can be used to change a user abend 35 to user abend 34 if a utility error occurs. This ensures that a dump is produced when the utility terminates abnormally.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

CODE: Cipher Code

The CODE parameter is required if the file or file(s) being processed by the ADACHK utility are enciphered. Specify the required cipher key.

If the data in a file processed by the ADACHK DSCHECK run is in ciphered form, the cipher key must be specified using the CODE parameter. In addition, you must also include the cipher file in the FILE parameter. Only one cipher file can be specified in the run. For more information, read *Cipher File Considerations*, earlier in this section.

For more information on the use of ciphering in Adabas databases, contact your technical support representative for more information about Adabas Security documentation.

COMPREC: Check and Print Compressed Record

Specify this parameter to check and print the compressed Data Storage records.

This is a comparable function to the ADAICK DSCHECK function. The FILE and ISN parameters must also be specified when this parameter is specified. A spanned record will be printed as a single logical record.

ERRLIM: Error Threshold

The maximum number of errors that this ADACHK utility function will tolerate before terminating. Valid values are any positive integer <=5000. If no valid value is specified for this parameter, a default of "100" is used.

If a value less than 0 or greater than 5000 is specified, the following error will result, and the default value will be used:

```
CHK413E, ERROR: Parameter ERRLIM is incorrect.
```

FILE: Files To Be Checked

The file or single range of files to be checked. If no file or range of files are specified, all files in the database are checked.

If the file listed in the FILE parameter is enciphered, the CODE parameter is also required. You must specify the cipher key in the CODE parameter. For more information, read *Cipher File Considerations*, earlier in this section.

FROMRABN: Starting RABN to be Checked

The RABN of the Data Storage block where the check should start. This parameter is applicable only if a single file is to be checked; this parameter cannot be specified in conjunction with the ISN parameter. In other words, only one FROMRABN/TORABN range can be specified in a single ADACHK run. If more than one FROMRABN/TORABN range is specified in an ADACHK request for multiple files, only the last range is used in the run. In addition, if the range specified by the FROMRABN/TORABN parameters is outside the range for any given file Data Storage extent, ADACHK will not check the blocks in the extent. Consequently, if more than one FROMRABN/TORABN range is specified, or if the range is outside the range for any given file Data Storage extent, Adabas issues a warning message.

If this parameter is omitted and if the ISN parameter is omitted, the check starts at the beginning of the first allocated Data Storage extent for the file.

ISN: ISN Range to be Checked

The ISN or range of ISNs to be checked. If no ISN or range of ISNs are specified, the entire range of ISNs (from MINISN through TOPISN) is checked.

You cannot specify the ISN parameter in conjunction with the FROMRABN parameter; these two parameters are mutually exclusive.

LAYOUT: Report Detail Level

The level of data produced for the report. Valid values are described in the following table:

Valid Values	Description
SHORT	Specify this value to produce the minimum output. This is the default.
MEDIUM	Specify this value to produce medium-level output (several interpreted lines).
LONG	Specify this value to produce extensively-detailed output.

LWP: Sort Work Pool Size

LWP specifies the size of the work pool to be used for sorting, if SORTTYPE=INTERNAL. It cannot be specified with SORTTYPE=EXTERNAL. The LWP value can be specified in bytes or kilobytes (value followed by a "K"). If no value is specified, the default is 10485760 bytes (or 10240K).

If ADARUN parameter V64BIT=YES has been specified, the work pool is allocated in 64-bit addressable storage.

The minimum value for LWP is 102,400 (100K). The maximum value is limited by how much 31-bit or 64-bit addressable virtual storage ADACHK can obtain from the system.

When the amount of data to be sorted exceeds the space declared by the LWP parameter, one or two sort data sets defined in the ADACHK job may be used as temporary storage. For more information, read *Sort Data Set Considerations*, elsewhere in this chapter. The sort data set(s) should be at least as large as the combined ASSO and DATA extents of the largest file to be processed by ADACHK.

If a sort data set is provided and large enough, the default value of 10 megabytes is sufficient for checking Adabas files of about three gigabytes. For checking larger files, LWP should be increased. As a rule of thumb, an LWP of size `n*10` MB supports files sizes of about `n*n*3` GB. For example, LWP=102400K (100 megabytes = `10*10` MB) is sufficient for processing files of about 300 gigabytes (= `10*10*3` GB).

It is generally recommended you make LWP larger than the required minimum to reduce the number of I/Os for sorting and thus to improve the performance of ADACHK.

MAXCALLS: Maximum Number of Nucleus Calls

The maximum number of nucleus calls allowed per file. Once this limit is hit, no more nucleus calls can be made and a warning message that the limit has been reached is printed. Valid values are any positive integer, with no upper limit.

The default for MAXCALLS is either the value "200" or the result of the TOPISN setting divided by 100 (TOPISN/100), whichever is greater.

NOSYNC: Transient Error Confirmation



Note: Use of this parameter is not recommended and is only intended for users who want to limit calls to their nucleus.

Use this parameter to control whether transient errors are confirmed via the active nucleus. If this parameter is specified, no nucleus calls are made and the errors reported may be due to concurrent updates. A warning message is generated informing you.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

REPAIR: Repair the Data Storage Space Table

Specify this parameter if you would like ADACHK to repair the Data Storage Space Table (DSST) if it finds any invalid DSST elements.

SORTTYPE: Sort Type Used

The type of sort to use. Valid values are "INTERNAL" (the default) and "EXTERNAL".

Value	Meaning
EXTERNAL	An external sort facility will be used to sort the data. For information about operating system considerations related to this option, read ***SORTTYPE=EXTERNAL Operating System Considerations***, elsewhere in this section.
INTERNAL	An internal sort facility will be used to sort the data. This is the default. For information about the use of sort data sets with this setting, read ***Sort Data Set Considerations***, elsewhere in this chapter.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See ***Syntax Checking with the TEST Parameter*** for more information about using the TEST parameter in ADACHK functions.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

Note: If your database contains cipher files, note that the TEST parameter does not check the types of files in the database. This means that the TEST run will not show any ADACHK errors related to cipher files in the run; these errors will only become apparent when you perform the ADACHK run without the TEST parameter. For more information, read *Cipher File Considerations*, earlier in this section.

TORABN: Ending RABN to be Checked

Use this parameter to specify the RABN of the Data Storage block where the ADACHK check should end. This parameter is applicable only if a single file is to be checked. In other words, only one FROMRABN/TORABN range can be specified in a single ADACHK run. If more than one FROMRABN/TORABN range is specified in an ADACHK request for multiple files, only the last range is used in the run. In addition, if the range specified by the FROMRABN/TORABN parameters is outside the range for any given file Data Storage extent, ADACHK will not check the blocks in the extent. Consequently, if more than one FROMRABN/TORABN range is specified, or if the range is outside the range for any given file Data Storage extent, Adabas issues a warning message.

If this parameter is omitted, the check ends at the end of the last allocated Data Storage extent for the file.

USAGE: Print Bar Graph of Data Storage Block Usage

If the USAGE is specified, ADACHK prints a bar graph that shows the number of bytes used in each Data Storage block, the block size, and the percentage of blocks used.

UTYPE: User Type

The user type in effect for the ADACHK run. Valid values are:

Value	Identifies
ACC	An access-only user. This is the default. When ACC is specified, parallel updates are allowed on the file by normal users (but not by other utility processes).
EXF	An exclusive file control user. When EXF is specified, only the ADACHK process can use the file; no other users can read or write to the file in parallel with the ADACHK run.
EXU	An exclusive control user. When EXU is specified, parallel reads are allowed on the file by normal users. This option might provide better performance if there are a lot of concurrent updates ongoing.

Examples

The following example checks Data Storage and the Data Storage Space Table (DSST) for file 5 in the database. It will produce a bar graph showing the number of bytes used in each Data Storage block, the block size, and the percentage of blocks used. In addition, if DSST errors are identified during the check, they will be repaired.

```
ADACHK DSCHECK FILE=5, USAGE,REPAIR
```

The following example checks Data Storage and the DSST for file 1 RABNs ranging from 412 to 420.

```
ADACHK DSCHECK FILE=1, FROMRABN=412,TORABN=420
```

The following example checks Data Storage and the DSST for file 1, ISN 4. The compressed record for file 1, ISN 4 is also checked and printed.

```
ADACHK DSCHECK FILE=1, ISN=4, COMPREC
```

17 DSIMPRINT: Print Delta Save Image Data Set Blocks

- Essential Parameters 102
- Optional Parameters and Subparameters 102
- Example 104

The ADACHK DSIMPRINT function can be used to print the contents of any block or range of blocks in the Delta Save image (DSIM) data set. It can be run while concurrent updates are running.

The syntax of the ADACHK DSIMPRINT function is:

```
ADACHK DSIMPRINT  RABN = { rabn | rabn1 - rabn2 } [ ,{ rabn | rabn1 - rabn2 } ]...
                  [ABEND34]
                  [DEVICE = device ]
                  [ERRLIM = { error-threshold-count | 100 } ]
                  [LAYOUT = { SHORT | MEDIUM | LONG } ]
                  [NOUSERABEND]
                  [TEST]
```

This chapter describes the syntax and parameters of the DSIMPRINT function.

Essential Parameters

RABN: RABNs to Be Processed

The RABNs or ranges of RABNs to be printed or dumped.

Optional Parameters and Subparameters

ABEND34: Change User Abend 35 to 34

This optional parameter can be used to change a user abend 35 to user abend 34 if a utility error occurs. This ensures that a dump is produced when the utility terminates abnormally.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

DEVICE: Device Type

The device type that contains the data set to be printed. This parameter is required if the device type is different from the standard device type assigned by the ADARUN DEVICE parameter.

ERRLIM: Error Threshold

The maximum number of errors that this ADACHK utility function will tolerate before terminating. Valid values are any positive integer <=5000. If no valid value is specified for this parameter, a default of "100" is used.

If a value less than 0 or greater than 5000 is specified, the following error will result, and the default value will be used:

```
CHK413E, ERROR: Parameter ERRLIM is incorrect.
```

LAYOUT: Report Detail Level

The level of data produced for the report or dump.Valid values are described in the following table. For more information about the output sections described below, read *ADACHK Print Function Output Format*, elsewhere in this section.

Valid Values	Description
SHORT	Specify this value to produce the minimum output, printing sections 1 and 2 of the possible output. This is the default.
MEDIUM	Specify this value to produce medium-level output, printing sections 1 and 3 of the possible output.
LONG	Specify this value to produce extensively-detailed output, printing all three sections of the possible output. **Note:** The results from the LAYOUT=LONG setting are what you get when you run the equivalent ADAICK utility function.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See *Syntax Checking with the TEST Parameter* for more information about using the TEST parameter in ADACHK functions.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

Example

In the following example, Delta Save image data set RABNs 6, 7, 8, 9, 2222, and 5000 are printed.

```
ADACHK DSIMPRINT RABN=6-9,5000,2222
```

18 DSSTPRINT: Print Data Storage Space Table

- Essential Parameters 106
- Optional Parameters and Subparameters 106
- Example 108

The ADACHK DSSTPRINT function can be used to print the contents of the Data Storage Space Table (DSST) while concurrent updates are running.

The syntax of the ADACHK DSSTPRINT is:

```
ADACHK DSSTPRINT  [ABEND34]
                  [ERRLIM = { error-threshold-count | 100 } ]
                  [LAYOUT = { SHORT | MEDIUM | LONG } ]
                  [NOUSERABEND]
                  [TEST]
```

This chapter describes the syntax and parameters of the DSSTPRINT function.

Essential Parameters

There are no required parameters for this utility function.

Optional Parameters and Subparameters

ABEND34: Change User Abend 35 to 34

This optional parameter can be used to change a user abend 35 to user abend 34 if a utility error occurs. This ensures that a dump is produced when the utility terminates abnormally.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

ERRLIM: Error Threshold

The maximum number of errors that this ADACHK utility function will tolerate before terminating. Valid values are any positive integer <=5000. If no valid value is specified for this parameter, a default of "100" is used.

If a value less than 0 or greater than 5000 is specified, the following error will result, and the default value will be used:

```
CHK413E, ERROR: Parameter ERRLIM is incorrect.
```

LAYOUT: Report Detail Level

The level of data produced for the report or dump.Valid values are described in the following table. For more information about the output sections described below, read *ADACHK Print Function Output Format*, elsewhere in this section.

Valid Values	Description
SHORT	Specify this value to produce the minimum output, printing sections 1 and 2 of the possible output. This is the default.
MEDIUM	Specify this value to produce medium-level output, printing sections 1 and 3 of the possible output.
LONG	Specify this value to produce extensively-detailed output, printing all three sections of the possible output. **Note:** The results from the LAYOUT=LONG setting are what you get when you run the equivalent ADAICK utility function.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See *Syntax Checking with the TEST Parameter* for more information about using the TEST parameter in ADACHK functions.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

Example

In the following example, the contents of the DSST are printed.

```
ADACHK DSSTPRINT
```

19 FCBPRINT: Print/Dump File Control Block (FCB)

- Essential Parameters 110
- Optional Parameters and Subparameters 110
- Example 112

The ADACHK FCBPRINT function can be used to print the contents of the FCB for any file in the database while concurrent updates are running.

```
ADACHK FCBPRINT   [ABEND34]
                  [ERRLIM = { error-threshold-count | 100 } ]
                  [FILE = { file | file1 - file2 [ ,{ file | file1 - file2 } ]... | all-files } ]
                  [LAYOUT = { SHORT | MEDIUM |LONG } ]
                  [NOUSERABEND]
                  [TEST]
```

This chapter describes the syntax and parameters of the FCBPRINT function.

Essential Parameters

There are no required parameters for this utility function.

Optional Parameters and Subparameters

ABEND34: Change User Abend 35 to 34

This optional parameter can be used to change a user abend 35 to user abend 34 if a utility error occurs. This ensures that a dump is produced when the utility terminates abnormally.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

ERRLIM: Error Threshold

The maximum number of errors that this ADACHK utility function will tolerate before terminating. Valid values are any positive integer <=5000. If no valid value is specified for this parameter, a default of "100" is used.

If a value less than 0 or greater than 5000 is specified, the following error will result, and the default value will be used:

```
CHK413E, ERROR: Parameter ERRLIM is incorrect.
```

FILE: File Number

The file numbers or file range for which the FCB is to be printed or dumped. If no file number is specified, the FCB for all files in the database are printed.

LAYOUT: Report Detail Level

The level of data produced for the report or dump.Valid values are described in the following table. For more information about the output sections described below, read ***ADACHK Print Function Output Format***, elsewhere in this section.

Valid Values	Description
SHORT	Specify this value to produce the minimum output, printing sections 1 and 2 of the possible output. This is the default.
MEDIUM	Specify this value to produce medium-level output, printing sections 1 and 3 of the possible output.
LONG	Specify this value to produce extensively-detailed output, printing all three sections of the possible output. **Note:** The results from the LAYOUT=LONG setting are what you get when you run the equivalent ADAICK utility function.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "`utility` TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See ***Syntax Checking with the TEST Parameter*** for more information about using the TEST parameter in ADACHK functions.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

Example

In the following example, the FCBs for files 1, 2, and 3 are printed. Medium-level output is produced.

```
ADACHK FCBPRINT FILE=1-3,LAYOUT=MEDIUM
```

20 FDTPRINT: Print/Dump Field Definition Table (FDT)

- Essential Parameters 114
- Optional Parameters and Subparameters 114
- Example 116

The ADACHK FDTPRINT function can be used to print the contents of the FDT for any file in the database while concurrent updates are running.

```
ADACHK FDTPRINT   [ABEND34]
                  [ERRLIM = { error-threshold-count | 100 } ]
                  [FILE = { file | file1 - file2 [ ,{ file | file1 - file2 } ]... | all-files } ]
                  [LAYOUT = { SHORT | MEDIUM | LONG } ]
                  [NOUSERABEND]
                  [TEST]
```

This chapter describes the syntax and parameters of the FDTPRINT function.

Essential Parameters

There are no required parameters for this utility function.

Optional Parameters and Subparameters

ABEND34: Change User Abend 35 to 34

This optional parameter can be used to change a user abend 35 to user abend 34 if a utility error occurs. This ensures that a dump is produced when the utility terminates abnormally.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

ERRLIM: Error Threshold

The maximum number of errors that this ADACHK utility function will tolerate before terminating. Valid values are any positive integer <=5000. If no valid value is specified for this parameter, a default of "100" is used.

If a value less than 0 or greater than 5000 is specified, the following error will result, and the default value will be used:

```
CHK413E, ERROR: Parameter ERRLIM is incorrect.
```

FILE: File Number

The file numbers or file range for which the FDT is to be printed or dumped. If no file number is specified, the FDT for all files in the database are printed.

LAYOUT: Report Detail Level

The level of data produced for the report or dump.Valid values are described in the following table. For more information about the output sections described below, read ***ADACHK Print Function Output Format***, elsewhere in this section.

Valid Values	Description
SHORT	Specify this value to produce the minimum output, printing sections 1 and 2 of the possible output. This is the default.
MEDIUM	Specify this value to produce medium-level output, printing sections 1 and 3 of the possible output.
LONG	Specify this value to produce extensively-detailed output, printing all three sections of the possible output. **Note:** The results from the LAYOUT=LONG setting are what you get when you run the equivalent ADAICK utility function.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See ***Syntax Checking with the TEST Parameter*** for more information about using the TEST parameter in ADACHK functions.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

Example

In the following example, the FDTs for all files will be printed. The most extensive output is produced (LAYOUT=LONG).

```
ADACHK FDTPRINT LAYOUT=LONG
```

21 FSTPRINT: Print/Dump Free Space Table (FST)

- Essential Parameters 118
- Optional Parameters and Subparameters 118
- Example 120

The ADACHK FSTPRINT function can be used to print the contents of the free space table (FST) for the database while concurrent updates are running.

```
ADACHK FSTPRINT   [ABEND34]
                  [ERRLIM = { error-threshold-count | 100 } ]
                  [LAYOUT = { SHORT | MEDIUM | LONG } ]
                  [NOUSERABEND]
                  [TEST]
```

This chapter describes the syntax and parameters of the FSTPRINT function.

Essential Parameters

There are no required parameters for this utility function.

Optional Parameters and Subparameters

ABEND34: Change User Abend 35 to 34

This optional parameter can be used to change a user abend 35 to user abend 34 if a utility error occurs. This ensures that a dump is produced when the utility terminates abnormally.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

ERRLIM: Error Threshold

The maximum number of errors that this ADACHK utility function will tolerate before terminating. Valid values are any positive integer <=5000. If no valid value is specified for this parameter, a default of "100" is used.

If a value less than 0 or greater than 5000 is specified, the following error will result, and the default value will be used:

```
CHK413E, ERROR: Parameter ERRLIM is incorrect.
```

LAYOUT: Report Detail Level

The level of data produced for the report or dump.Valid values are described in the following table. For more information about the output sections described below, read *ADACHK Print Function Output Format*, elsewhere in this section.

Valid Values	Description
SHORT	Specify this value to produce the minimum output, printing sections 1 and 2 of the possible output. This is the default.
MEDIUM	Specify this value to produce medium-level output, printing sections 1 and 3 of the possible output.
LONG	Specify this value to produce extensively-detailed output, printing all three sections of the possible output. **Note:** The results from the LAYOUT=LONG setting are what you get when you run the equivalent ADAICK utility function.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See *Syntax Checking with the TEST Parameter* for more information about using the TEST parameter in ADACHK functions.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

Example

In the following example, the free space table (FST) will be printed. The most extensive output is produced (LAYOUT=LONG).

```
ADACHK FSTPRINT LAYOUT=LONG
```

22 GCBPRINT: Print/Dump General Control Blocks (GCBs)

- Essential Parameters 122
- Optional Parameters and Subparameters 122
- Example 124

The ADACHK GCBPRINT function can be used to print the contents of the general control blocks (GCBs) for the database while concurrent updates are running.

```
ADACHK GCBPRINT   [ABEND34]
                  [ERRLIM = { error-threshold-count | 100 } ]
                  [LAYOUT = { SHORT | MEDIUM | LONG } ]
                  [NOUSERABEND]
                  [TEST]
```

This chapter describes the syntax and parameters of the GCBPRINT function.

Essential Parameters

There are no required parameters for this utility function.

Optional Parameters and Subparameters

ABEND34: Change User Abend 35 to 34

This optional parameter can be used to change a user abend 35 to user abend 34 if a utility error occurs. This ensures that a dump is produced when the utility terminates abnormally.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

ERRLIM: Error Threshold

The maximum number of errors that this ADACHK utility function will tolerate before terminating. Valid values are any positive integer <=5000. If no valid value is specified for this parameter, a default of "100" is used.

If a value less than 0 or greater than 5000 is specified, the following error will result, and the default value will be used:

```
CHK413E, ERROR: Parameter ERRLIM is incorrect.
```

LAYOUT: Report Detail Level

The level of data produced for the report or dump.Valid values are described in the following table. For more information about the output sections described below, read *ADACHK Print Function Output Format*, elsewhere in this section.

Valid Values	Description
SHORT	Specify this value to produce the minimum output, printing sections 1 and 2 of the possible output. This is the default.
MEDIUM	Specify this value to produce medium-level output, printing sections 1 and 3 of the possible output.
LONG	Specify this value to produce extensively-detailed output, printing all three sections of the possible output. **Note:** The results from the LAYOUT=LONG setting are what you get when you run the equivalent ADAICK utility function.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See *Syntax Checking with the TEST Parameter* for more information about using the TEST parameter in ADACHK functions.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

Example

In the following example, the general control blocks will be printed. The minimum output is produced (LAYOUT=SHORT is the default).

```
ADACHK GCBPRINT
```

23 ICHECK: Check Index Physical Structure

- Essential Parameters 126
- Optional Parameters and Subparameters 126
- Example 128

Use the ADACHK ICHECK utility function to check the physical structure of the index while concurrent updates are running. This utility function validates the index based upon the descriptor value structures and the Associator extents defined by the general control blocks (GCBs) and file control blocks (FCBs). It also checks the check index and address converter for specific files, invoking the ACCHECK function to provide this functionality.

The syntax of the ADACHK ICHECK utility function is:

```
ADACHK  ICHECK      [ABEND34]
                    [ERRLIM = { error-threshold-count | 100 } ]
                    [FILE= { file | file1 - filex | all-files } ]
                    [LAYOUT = { SHORT | MEDIUM | LONG } ]
                    [MAXCALLS = { count | [ 200 | ( TOPISN-value/100 ) ] } ]
                    [NOSYNC]
                    [NOUSERABEND]
                    [TEST]
                    [UTYPE = { ACC | EXF | EXU } ]
```

This chapter describes the syntax and parameters of the ICHECK function.

Essential Parameters

There are no required parameters for this utility function.

Optional Parameters and Subparameters

The following optional parameters can be specified.

ABEND34: Change User Abend 35 to 34

This optional parameter can be used to change a user abend 35 to user abend 34 if a utility error occurs. This ensures that a dump is produced when the utility terminates abnormally.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

ERRLIM: Error Threshold

The maximum number of errors that this ADACHK utility function will tolerate before terminating. Valid values are any positive integer <=5000. If no valid value is specified for this parameter, a default of "100" is used.

If a value less than 0 or greater than 5000 is specified, the following error will result, and the default value will be used:

```
CHK413E, ERROR: Parameter ERRLIM is incorrect.
```

FILE: Files To Be Checked

The file or single range of files to be checked. If no file or range of files are specified, all files in the database are checked.

LAYOUT: Report Detail Level

The level of data produced for the report. Valid values are described in the following table:

Valid Values	Description
SHORT	Specify this value to produce the minimum output. This is the default.
MEDIUM	Specify this value to produce medium-level output (several interpreted lines).
LONG	Specify this value to produce extensively-detailed output.

MAXCALLS: Maximum Number of Nucleus Calls

The maximum number of nucleus calls allowed per file. Once this limit is hit, no more nucleus calls can be made and a warning message that the limit has been reached is printed. Valid values are any positive integer, with no upper limit.

The default for MAXCALLS is either the value "200" or the result of the TOPISN setting divided by 100 (TOPISN/100), whichever is greater.

NOSYNC: Transient Error Confirmation



Note: Use of this parameter is not recommended and is only intended for users who want to limit calls to their nucleus.

Use this parameter to control whether transient errors are confirmed via the active nucleus. If this parameter is specified, no nucleus calls are made and the errors reported may be due to concurrent updates. A warning message is generated informing you.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See ***Syntax Checking with the TEST Parameter*** for more information about using the TEST parameter in ADACHK functions.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

UTYPE: User Type

The user type in effect for the ADACHK run. Valid values are:

Value	Identifies
ACC	An access-only user. This is the default. When ACC is specified, parallel updates are allowed on the file by normal users (but not by other utility processes).
EXF	An exclusive file control user. When EXF is specified, only the ADACHK process can use the file; no other users can read or write to the file in parallel with the ADACHK run.
EXU	An exclusive control user. When EXU is specified, parallel reads are allowed on the file by normal users. This option might provide better performance if there are a lot of concurrent updates ongoing.

Example

The following example checks the index structure for file 2.

```
ADACHK ICHECK FILE=2
```

24 NIPRINT: Print/Dump Normal Index

- Essential Parameters 130
- Optional Parameters and Subparameters 130
- Example 132

The ADACHK NIPRINT function can be used to print the contents of the normal index of files or ranges of files in the database while concurrent updates are running.

```
ADACHK NIPRINT      [ABEND34]
                    [ERRLIM = { error-threshold-count | 100 } ]
                    [FILE = { file | file1 - file2 [ ,{ file | file1 - file2 } ]... | all-files } ]
                    [LAYOUT = { SHORT | MEDIUM | LONG } ]
                    [NOUSERABEND]
                    [TEST]
```

This chapter describes the syntax and parameters of the NIPRINT function.

Essential Parameters

There are no required parameters for this utility function.

Optional Parameters and Subparameters

ABEND34: Change User Abend 35 to 34

This optional parameter can be used to change a user abend 35 to user abend 34 if a utility error occurs. This ensures that a dump is produced when the utility terminates abnormally.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

ERRLIM: Error Threshold

The maximum number of errors that this ADACHK utility function will tolerate before terminating. Valid values are any positive integer <=5000. If no valid value is specified for this parameter, a default of "100" is used.

If a value less than 0 or greater than 5000 is specified, the following error will result, and the default value will be used:

```
CHK413E, ERROR: Parameter ERRLIM is incorrect.
```

FILE: File Number

The file numbers or file range for which the normal index is to be printed or dumped. If no file number is specified, the normal index for all files in the database are printed.

LAYOUT: Report Detail Level

The level of data produced for the report or dump.Valid values are described in the following table. For more information about the output sections described below, read ***ADACHK Print Function Output Format***, elsewhere in this section.

Valid Values	Description
SHORT	Specify this value to produce the minimum output, printing sections 1 and 2 of the possible output. This is the default.
MEDIUM	Specify this value to produce medium-level output, printing sections 1 and 3 of the possible output.
LONG	Specify this value to produce extensively-detailed output, printing all three sections of the possible output. **Note:** The results from the LAYOUT=LONG setting are what you get when you run the equivalent ADAICK utility function.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "`utility` TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See ***Syntax Checking with the TEST Parameter*** for more information about using the TEST parameter in ADACHK functions.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

Example

In the following example, the normal index for all files in the database will be printed. The most extensive output is produced (LAYOUT=LONG).

```
ADACHK NIPRINT LAYOUT=LONG
```

25 PLOGPRINT: Print/Dump Protection Log Blocks

- Essential Parameters 134
- Optional Parameters and Subparameters 134
- Example 136

The ADACHK PLOGPRINT function can be used to print the contents of any block or range of blocks in the protection log (PLOG) data set. It can be run while concurrent updates are running.

```
ADACHK PLOGPRINT RABN = { rabn | rabn1 - rabn2 } [ ,{ rabn | rabn1 - rabn2 } ]...
                 [ABEND34]
                 [DEVICE = device ]
                 [ERRLIM = { error-threshold-count | 100 } ]
                 [NUMBER = { number | 1 } ]
                 [LAYOUT = { SHORT | MEDIUM | LONG } ]
                 [NOUSERABEND]
                 [TEST]
```

This chapter describes the syntax and parameters of the PLOGPRINT function.

Essential Parameters

RABN: RABNs to Be Processed

The RABNs or ranges of RABNs to be printed or dumped.

Optional Parameters and Subparameters

ABEND34: Change User Abend 35 to 34

This optional parameter can be used to change a user abend 35 to user abend 34 if a utility error occurs. This ensures that a dump is produced when the utility terminates abnormally.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

DEVICE: Device Type

The device type that contains the data set to be printed. This parameter is required if the device type is different from the standard device type assigned by the ADARUN DEVICE parameter.

ERRLIM: Error Threshold

The maximum number of errors that this ADACHK utility function will tolerate before terminating. Valid values are any positive integer <=5000. If no valid value is specified for this parameter, a default of "100" is used.

If a value less than 0 or greater than 5000 is specified, the following error will result, and the default value will be used:

```
CHK413E, ERROR: Parameter ERRLIM is incorrect.
```

NUMBER: Command/ Protection Log Data Set Number

The number of the multiple (two through eight) protection log (PLOG) data set from which the blocks are to be printed. If NUMBER is not specified, the blocks are taken from DDPLOGR1 (the default).

LAYOUT: Report Detail Level

The level of data produced for the report or dump.Valid values are described in the following table. For more information about the output sections described below, read *ADACHK Print Function Output Format*, elsewhere in this section.

Valid Values	Description
SHORT	Specify this value to produce the minimum output, printing sections 1 and 2 of the possible output. This is the default.
MEDIUM	Specify this value to produce medium-level output, printing sections 1 and 3 of the possible output.
LONG	Specify this value to produce extensively-detailed output, printing all three sections of the possible output. **Note:** The results from the LAYOUT=LONG setting are what you get when you run the equivalent ADAICK utility function.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See *Syntax Checking with the TEST Parameter* for more information about using the TEST parameter in ADACHK functions.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

Example

In the following example, protection log RABNs 1, 2, 30-31, and 2000 are printed from the DDPLOG provided in the JCL.

```
ADACHK PLOGPRINT RABN=1,2,30-31,2000
```

26 PPTPRINT: Print/Dump Parallel Participant Table (PPT)

- Essential Parameters 138
- Optional Parameters and Subparameters 138
- Example 140

The ADACHK PPTPRINT function can be used to print the contents of the parallel participant table (PPT) for the database. It can be run while concurrent updates are running.

```
ADACHK PPTPRINT   [ABEND34]
                  [ERRLIM = { error-threshold-count | 100 } ]
                  [LAYOUT = { SHORT | MEDIUM | LONG } ]
                  [NOUSERABEND]
                  [TEST]
```

This chapter describes the syntax and parameters of the PPTPRINT function.

Essential Parameters

There are no required parameters for this utility function.

Optional Parameters and Subparameters

ABEND34: Change User Abend 35 to 34

This optional parameter can be used to change a user abend 35 to user abend 34 if a utility error occurs. This ensures that a dump is produced when the utility terminates abnormally.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

ERRLIM: Error Threshold

The maximum number of errors that this ADACHK utility function will tolerate before terminating. Valid values are any positive integer <=5000. If no valid value is specified for this parameter, a default of "100" is used.

If a value less than 0 or greater than 5000 is specified, the following error will result, and the default value will be used:

```
CHK413E, ERROR: Parameter ERRLIM is incorrect.
```

LAYOUT: Report Detail Level

The level of data produced for the report or dump.Valid values are described in the following table. For more information about the output sections described below, read *ADACHK Print Function Output Format*, elsewhere in this section.

Valid Values	Description
SHORT	Specify this value to produce the minimum output, printing sections 1 and 2 of the possible output. This is the default.
MEDIUM	Specify this value to produce medium-level output, printing sections 1 and 3 of the possible output.
LONG	Specify this value to produce extensively-detailed output, printing all three sections of the possible output. **Note:** The results from the LAYOUT=LONG setting are what you get when you run the equivalent ADAICK utility function.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See *Syntax Checking with the TEST Parameter* for more information about using the TEST parameter in ADACHK functions.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

Example

In the following example, the PPT will be printed. The minimum output is produced (LAYOUT=SHORT is the default).

```
ADACHK PPTPRINT
```

27 RLOGPRINT: Print/Dump Recovery Log

- Essential Parameters 142
- Optional Parameters and Subparameters 142
- Example 144

The ADACHK RLOGPRINT function can be used to print the contents of any block or range of blocks in the recovery log (RLOG) data set. It can be run while concurrent updates are running.

```
ADACHK RLOGPRINT RABN = { rabn | rabn1 - rabn2 } [ ,{ rabn | rabn1 - rabn2 } ]...
                 [ABEND34]
                 [DEVICE = device ]
                 [ERRLIM = { error-threshold-count | 100 } ]
                 [LAYOUT = { SHORT | MEDIUM | LONG } ]
                 [NOUSERABEND]
                 [TEST]
```

This chapter describes the syntax and parameters of the RLOGPRINT function.

Essential Parameters

RABN: RABNs to Be Processed

The RABNs or ranges of RABNs to be printed or dumped.

Optional Parameters and Subparameters

ABEND34: Change User Abend 35 to 34

This optional parameter can be used to change a user abend 35 to user abend 34 if a utility error occurs. This ensures that a dump is produced when the utility terminates abnormally.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

DEVICE: Device Type

The device type that contains the data set to be printed. This parameter is required if the device type is different from the standard device type assigned by the ADARUN DEVICE parameter.

ERRLIM: Error Threshold

The maximum number of errors that this ADACHK utility function will tolerate before terminating. Valid values are any positive integer <=5000. If no valid value is specified for this parameter, a default of "100" is used.

If a value less than 0 or greater than 5000 is specified, the following error will result, and the default value will be used:

```
CHK413E, ERROR: Parameter ERRLIM is incorrect.
```

LAYOUT: Report Detail Level

The level of data produced for the report or dump.Valid values are described in the following table. For more information about the output sections described below, read *ADACHK Print Function Output Format*, elsewhere in this section.

Valid Values	Description
SHORT	Specify this value to produce the minimum output, printing sections 1 and 2 of the possible output. This is the default.
MEDIUM	Specify this value to produce medium-level output, printing sections 1 and 3 of the possible output.
LONG	Specify this value to produce extensively-detailed output, printing all three sections of the possible output. **Note:** The results from the LAYOUT=LONG setting are what you get when you run the equivalent ADAICK utility function.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See *Syntax Checking with the TEST Parameter* for more information about using the TEST parameter in ADACHK functions.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

Example

In the following example, recovery log RABNs 1, 2, 30-31, and 2000 are printed. The most extensive output is produced (LAYOUT=LONG).

```
ADACHK RLOGPRINT RABN=1,2,30-31,2000 LAYOUT=LONG
```

28 SORTPRINT: Print/Dump Sort Data Set

- Essential Parameters 146
- Optional Parameters and Subparameters 146
- Example 148

The ADACHK SORTPRINT function can be used to print the contents of any block or range of blocks in the sort (SORT) data set. It can be run while concurrent updates are running.

```
ADACHK SORTPRINT  RABN = { rabn | rabn1 - rabn2 } [ ,{ rabn | rabn1 - rabn2 } ]...
                  [ABEND34]
                  [DEVICE = device ]
                  [ERRLIM = { error-threshold-count | 100 } ]
                  [LAYOUT = { SHORT | MEDIUM | LONG } ]
                  [NOUSERABEND]
                  [TEST]
```

This chapter describes the syntax and parameters of the SORTPRINT function.

Essential Parameters

RABN: RABNs to Be Processed

The RABNs or ranges of RABNs to be printed or dumped.

Optional Parameters and Subparameters

ABEND34: Change User Abend 35 to 34

This optional parameter can be used to change a user abend 35 to user abend 34 if a utility error occurs. This ensures that a dump is produced when the utility terminates abnormally.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

DEVICE: Device Type

The device type that contains the data set to be printed. This parameter is required if the device type is different from the standard device type assigned by the ADARUN DEVICE parameter.

ERRLIM: Error Threshold

The maximum number of errors that this ADACHK utility function will tolerate before terminating. Valid values are any positive integer <=5000. If no valid value is specified for this parameter, a default of "100" is used.

If a value less than 0 or greater than 5000 is specified, the following error will result, and the default value will be used:

```
CHK413E, ERROR: Parameter ERRLIM is incorrect.
```

LAYOUT: Report Detail Level

The level of data produced for the report or dump.Valid values are described in the following table. For more information about the output sections described below, read *ADACHK Print Function Output Format*, elsewhere in this section.

Valid Values	Description
SHORT	Specify this value to produce the minimum output, printing sections 1 and 2 of the possible output. This is the default.
MEDIUM	Specify this value to produce medium-level output, printing sections 1 and 3 of the possible output.
LONG	Specify this value to produce extensively-detailed output, printing all three sections of the possible output. **Note:** The results from the LAYOUT=LONG setting are what you get when you run the equivalent ADAICK utility function.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See *Syntax Checking with the TEST Parameter* for more information about using the TEST parameter in ADACHK functions.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

Example

In the following example, sort data set RABNs 1, 2, 30-31, and 2000 are printed. Medium output is produced (LAYOUT=MEDIUM).

```
ADACHK SORTPRINT RABN=1,2,30-31,2000, LAYOUT=MEDIUM
```

29 TEMPPRINT: Print/Dump Temporary Data Set

- Essential Parameters 150
- Optional Parameters and Subparameters 150
- Example 152

The ADACHK TEMPPRINT function can be used to print the contents of any block or range of blocks in the temporary (TEMP) data set. It can be run while concurrent updates are running.

```
ADACHK TEMPPRINT  RABN = { rabn | rabn1 - rabn2 } [ ,{ rabn | rabn1 - rabn2 } ]...
                  [ABEND34]
                  [DEVICE = device ]
                  [ERRLIM = { error-threshold-count | 100 } ]
                  [LAYOUT = { SHORT | MEDIUM | LONG } ]
                  [NOUSERABEND]
                  [TEST]
```

This chapter describes the syntax and parameters of the TEMPPRINT function.

Essential Parameters

RABN: RABNs to Be Processed

The RABNs or ranges of RABNs to be printed or dumped.

Optional Parameters and Subparameters

ABEND34: Change User Abend 35 to 34

This optional parameter can be used to change a user abend 35 to user abend 34 if a utility error occurs. This ensures that a dump is produced when the utility terminates abnormally.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

DEVICE: Device Type

The device type that contains the data set to be printed. This parameter is required if the device type is different from the standard device type assigned by the ADARUN DEVICE parameter.

ERRLIM: Error Threshold

The maximum number of errors that this ADACHK utility function will tolerate before terminating. Valid values are any positive integer <=5000. If no valid value is specified for this parameter, a default of "100" is used.

If a value less than 0 or greater than 5000 is specified, the following error will result, and the default value will be used:

```
CHK413E, ERROR: Parameter ERRLIM is incorrect.
```

LAYOUT: Report Detail Level

The level of data produced for the report or dump.Valid values are described in the following table. For more information about the output sections described below, read *ADACHK Print Function Output Format*, elsewhere in this section.

Valid Values	Description
SHORT	Specify this value to produce the minimum output, printing sections 1 and 2 of the possible output. This is the default.
MEDIUM	Specify this value to produce medium-level output, printing sections 1 and 3 of the possible output.
LONG	Specify this value to produce extensively-detailed output, printing all three sections of the possible output. **Note:** The results from the LAYOUT=LONG setting are what you get when you run the equivalent ADAICK utility function.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See *Syntax Checking with the TEST Parameter* for more information about using the TEST parameter in ADACHK functions.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

Example

In the following example, TEMP data set RABNs 1, 2, 30-31, and 2000 are printed. Medium output is produced (LAYOUT=MEDIUM).

```
ADACHK TEMPPRINT RABN=1,2,30-31,2000, LAYOUT=MEDIUM
```

30 UIPRINT: Print/Dump Upper Index

- Essential Parameters 154
- Optional Parameters and Subparameters 154
- Example 156

The ADACHK UIPRINT function can be used to print the contents of the upper index of files or ranges of files in the database. It can be run while concurrent updates are running.

```
ADACHK UIPRINT      [ABEND34]
                    [ERRLIM = { error-threshold-count | 100 } ]
                    [FILE = { file | file1 - file2 [ ,{ file | file1 - file2 } ]... | all-files } ]
                    [LAYOUT = { SHORT | MEDIUM | LONG } ]
                    [NOUSERABEND]
                    [TEST]
```

This chapter describes the syntax and parameters of the UIPRINT function.

Essential Parameters

There are no required parameters for this utility function.

Optional Parameters and Subparameters

ABEND34: Change User Abend 35 to 34

This optional parameter can be used to change a user abend 35 to user abend 34 if a utility error occurs. This ensures that a dump is produced when the utility terminates abnormally.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

ERRLIM: Error Threshold

The maximum number of errors that this ADACHK utility function will tolerate before terminating. Valid values are any positive integer <=5000. If no valid value is specified for this parameter, a default of "100" is used.

If a value less than 0 or greater than 5000 is specified, the following error will result, and the default value will be used:

```
CHK413E, ERROR: Parameter ERRLIM is incorrect.
```

FILE: File Number

The file numbers or file range for which the upper index is to be printed or dumped. If no file number is specified, the upper index for all files in the database are printed.

LAYOUT: Report Detail Level

The level of data produced for the report or dump.Valid values are described in the following table. For more information about the output sections described below, read ***ADACHK Print Function Output Format***, elsewhere in this section.

Valid Values	Description
SHORT	Specify this value to produce the minimum output, printing sections 1 and 2 of the possible output. This is the default.
MEDIUM	Specify this value to produce medium-level output, printing sections 1 and 3 of the possible output.
LONG	Specify this value to produce extensively-detailed output, printing all three sections of the possible output. **Note:** The results from the LAYOUT=LONG setting are what you get when you run the equivalent ADAICK utility function.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "`utility` TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See ***Syntax Checking with the TEST Parameter*** for more information about using the TEST parameter in ADACHK functions.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

Example

In the following example, the upper index for all files in the database will be printed. The minimum output is produced (LAYOUT=SHORT is the default).

```
ADACHK UIPRINT
```

31 VALIDATE: Validate All Database Files

- Essential Parameters 158
- Optional Parameters and Subparameters 158
- Output 162
- Examples 165

Use the ADACHK VALIDATE utility function to validate any or all files within an Adabas database, except the checkpoint and security files. It can be run while concurrent updates are running.

The VALIDATE function validates the contents of Data Storage against values in the Associator. This is done by reading compressed records from Data Storage and creating a DVT that is validated against each corresponding value in the indices. ADACHK prints a list of all fields compared and the ISNs rejected during validation on DDDRUCK.

The syntax of the ADACHK VALIDATE utility function is:

```
ADACHK  VALIDATE  [ABEND34]
                  [CODE = cipherkey ]
                  [DESCRIPTOR = desc1 [ ,desc2 ] ... ]
                  [ERRLIM = { error-threshold-count | 100 } ]
                  [FILE= { file | file1 - filex | all-files } ]
                  [LAYOUT = { SHORT | MEDIUM | LONG } ]
                  [LWP = { workpoolsize | 10240K } ]
                  [MAXCALLS = { count | [ 200 | ( TOPISN-value/100 ) ] } ]
                  [MAXDESCLEN = length | FDT-largest-descriptor ]
                  [NOSYNC]
                  [NOUSERABEND]
                  [SORTTYPE = { INTERNAL | EXTERNAL } ]
                  [TEST]
                  [UTYPE = { ACC | EXF | EXU } ]
```

This chapter describes the syntax and parameters of the VALIDATE function.

Essential Parameters

There are no required parameters for this utility function.

Optional Parameters and Subparameters

The following optional parameters can be specified.

ABEND34: Change User Abend 35 to 34

This optional parameter can be used to change a user abend 35 to user abend 34 if a utility error occurs. This ensures that a dump is produced when the utility terminates abnormally.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

CODE: Cipher Code

The CODE parameter is required if the file or file(s) being processed by the ADACHK utility are enciphered. Specify the required cipher key.

If the data is in a file is in ciphered form, the cipher code must be specified using the COMPREC parameter. For more information on the use of ciphering in Adabas databases, contact your technical support representative for more information about Adabas Security documentation.

DESCRIPTOR: List of Descriptors

The DESCRIPTOR parameter restricts ADACHK processing to one descriptor field, providing a way to limit the run in cases in which the Associator is very large or there is a need to evaluate a specific descriptor. If DESCRIPTOR is not specified, ADACHK validates all qualifying descriptor fields.

The following is an example of DESCRIPTOR use:

```
ADACHK CHECK FILE=4,DESCRIPTOR='AA,BB'
```

ERRLIM: Error Threshold

The maximum number of errors that this ADACHK utility function will tolerate before terminating. Valid values are any positive integer <=5000. If no valid value is specified for this parameter, a default of "100" is used.

If a value less than 0 or greater than 5000 is specified, the following error will result, and the default value will be used:

```
CHK413E, ERROR: Parameter ERRLIM is incorrect.
```

FILE: Files To Be Checked

The file or single range of files to be checked. If no file or range of files are specified, all files in the database are checked.

LAYOUT: Report Detail Level

The level of data produced for the report. Valid values are described in the following table:

Valid Values	Description
SHORT	Specify this value to produce the minimum output. This is the default.
MEDIUM	Specify this value to produce medium-level output (several interpreted lines).
LONG	Specify this value to produce extensively-detailed output.

LWP: Sort Work Pool Size

LWP specifies the size of the work pool to be used for sorting, if SORTTYPE=INTERNAL. It cannot be specified with SORTTYPE=EXTERNAL. The LWP value can be specified in bytes or kilobytes (value followed by a "K"). If no value is specified, the default is 10485760 bytes (or 10240K).

If ADARUN parameter V64BIT=YES has been specified, the work pool is allocated in 64-bit addressable storage.

The minimum value for LWP is 102,400 (100K). The maximum value is limited by how much 31-bit or 64-bit addressable virtual storage ADACHK can obtain from the system.

When the amount of data to be sorted exceeds the space declared by the LWP parameter, one or two sort data sets defined in the ADACHK job may be used as temporary storage. For more information, read ***Sort Data Set Considerations***, elsewhere in this chapter. The sort data set(s) should be at least as large as the combined ASSO and DATA extents of the largest file to be processed by ADACHK.

If a sort data set is provided and large enough, the default value of 10 megabytes is sufficient for checking Adabas files of about three gigabytes. For checking larger files, LWP should be increased. As a rule of thumb, an LWP of size `n*10` MB supports files sizes of about `n*n*3` GB. For example, LWP=102400K (100 megabytes = `10*10` MB) is sufficient for processing files of about 300 gigabytes (= `10*10*3` GB).

It is generally recommended you make LWP larger than the required minimum to reduce the number of I/Os for sorting and thus to improve the performance of ADACHK.

MAXCALLS: Maximum Number of Nucleus Calls

The maximum number of nucleus calls allowed per file. Once this limit is hit, no more nucleus calls can be made and a warning message that the limit has been reached is printed. Valid values are any positive integer, with no upper limit.

The default for MAXCALLS is either the value "200" or the result of the TOPISN setting divided by 100 (TOPISN/100), whichever is greater.

MAXDESCLEN: Maximum Length of Descriptor Value

Use the MAXDESCLEN parameter to specify the maximum length of a descriptor value present in the file during compressed record extraction. If the MAXDESCLEN parameter is not specified, the default is to extract the largest default descriptor length from the FDT.

The output produced by ADACHK VALIDATE varies based on the status of the nucleus (whether it is up or down) and the setting of the MAXDESCLEN parameter. For more information, read ***Output Anomalies Based on Nucleus Status and MAXDESCLEN Setting***, elsewhere in this section.

NOSYNC: Transient Error Confirmation



Note: Use of this parameter is not recommended and is only intended for users who want to limit calls to their nucleus.

Use this parameter to control whether transient errors are confirmed via the active nucleus. If this parameter is specified, no nucleus calls are made and the errors reported may be due to concurrent updates. A warning message is generated informing you.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

SORTTYPE: Sort Type Used

The type of sort to use. Valid values are "INTERNAL" (the default) and "EXTERNAL".

Value	Meaning
EXTERNAL	An external sort facility will be used to sort the data. For information about operating system considerations related to this option, read ***SORTTYPE=EXTERNAL Operating System Considerations***, elsewhere in this section.
INTERNAL	An internal sort facility will be used to sort the data. This is the default. For information about the use of sort data sets with this setting, read ***Sort Data Set Considerations***, elsewhere in this chapter.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See ***Syntax Checking with the TEST Parameter*** for more information about using the TEST parameter in ADACHK functions.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

UTYPE: User Type

The user type in effect for the ADACHK run. Valid values are:

Value	Identifies
ACC	An access-only user. This is the default. When ACC is specified, parallel updates are allowed on the file by normal users (but not by other utility processes).
EXF	An exclusive file control user. When EXF is specified, only the ADACHK process can use the file; no other users can read or write to the file in parallel with the ADACHK run.
EXU	An exclusive control user. When EXU is specified, parallel reads are allowed on the file by normal users. This option might provide better performance if there are a lot of concurrent updates ongoing.

Output

This section describes various aspects of the output produced by the ADACHK VALIDATE function.

- Output Anomalies Based on Nucleus Status and MAXDESCLEN Setting
- Directing Rejected ISNs to a Sequential Data Set
- Example of ADACHK VALIDATE Output

Output Anomalies Based on Nucleus Status and MAXDESCLEN Setting

The output produced by ADACHK VALIDATE varies based on the status of the nucleus (whether it is up or down) and the setting of the MAXDESCLEN parameter.

In the following example, the nucleus is not active (it is down). Zero (0) transient errors were found but 32 calls were attempted. Since the nucleus was down and truncation occurred, ADACHK VALIDATE reports these as errors but provides a warning message (CHK416W) and a condition code of 8. It also provides the length of the longest truncated field so you can avoid the truncation error on the next run.

```
CHK025I, The Adabas nucleus is currently INACTIVE
MAXDESCLEN = 10
--------------------------------------------------------------------------------
File DE F ISN DE Value
---- -- - ------------ -------- -------- -------- -------- ------------------
CHK416W, WARNING: Value for field AA has been
truncated. Errors for this field may be due to truncated values. Specify a
larger MAXDESCLEN to avoid truncation of data storage record field values.

26 AA - 5 19C1C1C1 C1C14040 40404040 40404040 *.AAAAA * 40404040 40404040 4040 * *

CHK417I, The longest truncated field value was 50 bytes. This value belongs to
descriptor AA. Specify MAXDESCLEN at least this size on next run.

*****ADACHK Nucleus Statistics - Error Verifications*****
Number of transient errors = 0
Number of nucleus verification calls = 32 ↩
```

In the following example, the nucleus is active (it is up). ADACHK VALIDATE performs the truncation that resulted in an error. It then made a call to the nucleus to confirm whether the error was real or transient. In this case, it was transient. In this case, with the help of the nucleus, ADACHK was able to determine that the errors were transient due to the truncation and provides an informative message so that the setting of MAXDESCLEN can be increased to avoid the nucleus call in future invocations.

```
CHK025I, The Adabas nucleus is currently ACTIVE
MAXDESCLEN = 10 CHK416W,

WARNING: Value for field AA has been truncated. Errors for this
field may be due to truncated values. Specify a larger MAXDESCLEN to avoid
truncation of data storage record field values.

26 AA *** No Inconsistencies ***

CHK417I, The longest truncated field value was 50 bytes. This value belongs
to descriptor AA. Specify MAXDESCLEN at least this size on next run.

*****ADACHK Nucleus Statistics - Error Verifications*****
Number of transient errors = 32
Number of nucleus verification calls = 32
```

Directing Rejected ISNs to a Sequential Data Set

If you choose, rejected ISNs can be output to a sequential data set (DDFEHL). The first record on DDFEHL always has the following structure:

Bytes	Description
0-1	Record length in binary format (example: X'0012')
2-3	Set to zero (example: X'0000')
4-9	Program ID (example: C'ADACHK')
10-13	Four-byte packed Julian date in format, YYYYDDDF ("F" = B'1111')
14-17	Four-byte packed time in format, hhmmssth (t = tenths of a second, h = hundredths of a second)

All remaining DDFEHL records have the following structure (items shown with an asterisk (*) are also in the normal DDDRUCK output):

Bytes	Description
0-1	Record length in binary format (example: X'0012')
2-3	Set to zero (example: X'0000')
4-5	Adabas file number in binary format
6	Flag byte: ■ `C'-'`: A value is missing. ■ `C'+'` : A value is incorrect.
7	Set to zero.
8-11	ISN in binary format.
12-13	Descriptor name as stored in the field definition table (FDT).
14	Descriptor value length in binary format.

Bytes	Description
15-	Descriptor value

Example of ADACHK VALIDATE Output

ADACHK VALIDATE output provides a DDDRUCK table listing, by file and descriptor, of all Data Storage and Associator entries and their status. The following is an example of ADACHK VALIDATE output:

```
FILE  DE  F      ISN    DE-VALUE
-------------------------------------------------------------
  1   AA    *** NO INCONSISTENCIES ***
  1   BA  -        35  07C6D935 C5D4C1D5          *.FREEMAN*
  1   BA  -       173  07C6D935 C5D4C1D5          *.FREEMAN*
  1   BA  -       471  07C6D935 C5D4C1D5          *.FREEMAN*

  1   BA  -       534  07C6D935 C5D4C1D5          *.FREEMAN*
  1   BA  -       597  07C6D935 C5D4C1D5          *.FREEMAN*
  1   BA  -       622  07C6D935 C5D4C1D5          *.FREEMAN*
  1   BA  -       658  07C6D935 C5D4C1D5          *.FREEMAN*
  1   BA  -       717  07C6D935 C5D4C1D5          *.FREEMAN*
  1   BA  -       152  05D4C5E8 C5D9              *.MEYER*
  1   BA  +       153  05D4C5E8 C5D9              *.MEYER*
  1   BB    *** NO INCONSISTENCIES ***
  1   CA    *** NO INCONSISTENCIES ***
  1   CB    *** NO INCONSISTENCIES ***
  1   CC    *** NO INCONSISTENCIES ***
  1   CD    *** NO INCONSISTENCIES ***
  1   PA    *** NO INCONSISTENCIES ***
  1   S2    *** NO INCONSISTENCIES ***            DE DELETED
```

where:

- In the F (flag) column, a dash (--) indicates that an inverted list entry is missing for the specified Data Storage descriptor; and a plus symbol (+) indicates that the inverted list entry in the Associator is incorrect.
- The DE-VALUE column provides the compressed descriptor value, first in hexadecimal and then in alphanumeric.

Notes:

1. The "*** NO INCONSISTENCIES ***" entry occurs for every successful descriptor validation.
2. "DE DELETED" will be printed for logically deleted descriptors.

Examples

In the following example, the first VALIDATE function will validate files 1 through 3 using an external sort. The maximum length of a descriptor value present in files 1 through 3 is 15. The second VALIDATE function will only validate descriptor fields AA and BB in file 4.

```
ADACHK VALIDATE FILE=1-3,MAXDESCLEN=15,SORTTYPE=EXTERNAL
ADACHK VALIDATE FILE=4,DESCRIPTOR='AA,BB' ↩
```

32 WORKPRINT: Print/Dump Work Data Set

- Essential Parameters 168
- Optional Parameters and Subparameters 168
- Examples 170

The ADACHK WORKPRINT function can be used to print the contents of any block or range of blocks in the work (WORK) data set. It can be run while concurrent updates are running.

```
ADACHK WORKPRINT   RABN = { rabn | rabn1 - rabn2 } [ ,{ rabn | rabn1 - rabn2 } ]...
                   [ABEND34]
                   [DEVICE = device ]
                   [ERRLIM = { error-threshold-count | 100 } ]
                   [LAYOUT = { SHORT | MEDIUM | LONG } ]
                   [NOUSERABEND]
                   [TEST]
```

This chapter describes the syntax and parameters of the WORKPRINT function.

Essential Parameters

RABN: RABNs to Be Processed

The RABNs or ranges of RABNs to be printed or dumped.

Optional Parameters and Subparameters

ABEND34: Change User Abend 35 to 34

This optional parameter can be used to change a user abend 35 to user abend 34 if a utility error occurs. This ensures that a dump is produced when the utility terminates abnormally.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

ERRLIM: Error Threshold

The maximum number of errors that this ADACHK utility function will tolerate before terminating. Valid values are any positive integer <=5000. If no valid value is specified for this parameter, a default of "100" is used.

If a value less than 0 or greater than 5000 is specified, the following error will result, and the default value will be used:

```
CHK413E, ERROR: Parameter ERRLIM is incorrect.
```

DEVICE: Device Type

The device type that contains the data set to be printed. This parameter is required if the device type is different from the standard device type assigned by the ADARUN DEVICE parameter.

LAYOUT: Report Detail Level

The level of data produced for the report or dump.Valid values are described in the following table. For more information about the output sections described below, read ***ADACHK Print Function Output Format***, elsewhere in this section.

Valid Values	Description
SHORT	Specify this value to produce the minimum output, printing sections 1 and 2 of the possible output. This is the default.
MEDIUM	Specify this value to produce medium-level output, printing sections 1 and 3 of the possible output.
LONG	Specify this value to produce extensively-detailed output, printing all three sections of the possible output. **Note:** The results from the LAYOUT=LONG setting are what you get when you run the equivalent ADAICK utility function.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "`utility` TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See ***Syntax Checking with the TEST Parameter*** for more information about using the TEST parameter in ADACHK functions.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADACHK run.

Examples

In the following example, work data set RABNs 1, 2, 30-31, and 2000 are printed. The most extensive output is produced (LAYOUT=LONG).

```
ADACHK WORKPRINT RABN=1,2,30-31,2000,LAYOUT=LONG
```

33 JCL Requirements and Examples

- Requirements to Run ADACHK JCL 172

This section describes the job control information required to run ADACHK on the z/OS operating system and shows examples of the job stream.

Requirements to Run ADACHK JCL

Data Set	DD Name	Storage	More Information
Data set containing blocks to print. If you are running an ADACHK print function (such as ADACHK PLOGPRINT, ADACHK SORTPRINT, or ADACHK ASSOPRINT), include the corresponding data set specification in the job for the run. For information about which data sets to specify for each ADACHK function, read *Disk Accesses by Utility and Function*, in the *Adabas Operations Manual*.	DDASSORn	disk	Associator
	DDDATARn	disk	Data Storage
	DDWORKR1	disk	Work
	DDWORKR4	disk	
	DDTEMPR1	disk	Temp
	DDSORTRn	disk	Sort
	DDCLOGRn	disk	Multiple command log
	DDPLOGRn	disk	Multiple protection log
	DDDSIMR1	disk	DSIM data set
Associator	DDASSOR*n* or DDASSO*nn*	disk	
Data Storage	DDDATAR*n* or DDDATA*nn*	disk	
Sort work data sets	DDSORTR1 or DDSORTR2		Used by the ACCHECK, CHECK, DSCHECK, and VALIDATE utility functions only when SORTTYPE=INTERNAL (the default) is specified and when the amount of data to be sorted exceeds the space declared by the LWP parameter in these utility function runs.
ADARUN parameters	DDCARD	reader	*Operations*
ADACHK parameters	DDKARTE	reader	
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADACHK messages	DDDRUCK	printer	*Messages and Codes*

ADACHK JCL Examples

This section provides samples of z/OS ADACHK JCL:

Example Checking the Entire Database

```
//ADACHKC   JOB
//*
//*   ADACHK: CHECK
//*      Check of entire database
//*
//CHK       EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD    <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DBYYYYY.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DBYYYYY.DATAR1 <=== DATA
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADACHK,SVC=XXX,DEVICE=3390,DBID=YYYYY
/*
//DDKARTE   DD   *
ADACHK CHECK
/*
//
```

Example Specifying Sort Data Sets

```
//ADACHKS   JOB
//*
//*   ADACHK: Execute ADACHK CHECK specifying sort datasets
//*           to be used in the event that LWP is insufficient.
//*
//CHK       EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD    <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DBYYYYY.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DBYYYYY.DATAR1 <=== DATA
//DDSORTR1  DD   DISP=NEW,DSN=EXAMPLE.SORTR1,SPACE=(CYL,(100,500))
//DDSORTR2  DD   DISP=NEW,DSN=EXAMPLE.SORTR2,SPACE=(CYL,(100,500))
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADACHK,SVC=XXX,DEVICE=3390,DBID=YYYYY
/*
//DDKARTE   DD   *
ADACHK CHECK LWP=256000
/*
//
```

Example of ADACHK Print Functions

```
//ADACHKP   JOB
//*
//*   ADACHK: PRINT GCB followed by PPTPRINT and ASSO and DATA
//*      prints of various RABNs
//*
//CHK       EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD    <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DBYYYYY.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DBYYYYY.DATAR1 <=== DATA
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADACHK,SVC=XXX,DEVICE=3390,DBID=YYYYY
/*
//DDKARTE   DD   *
ADACHK GCBPRINT
ADACHK PPTPRINT
ADACHK ASSOPRINT RABN=1000-1002
ADACHK ASSOPRINT RABN=2006
ADACHK DATAPRINT RABN=25006-25007
ADACHK DATAPRINT RABN=17000-17999
/*
//
```

Example of ADACHK JCL Running Both Print and Check Functions

```
//ADACHKV   JOB
//*
//*   ADACHK: Execute multiple print and check functions
//*
//*
//CHK       EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD    <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DBYYYYY.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DBYYYYY.DATAR1 <=== DATA
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADACHK,SVC=XXX,DEVICE=3390,DBID=YYYYY
/*
//DDKARTE   DD   *
ADACHK ICHECK
ADACHK DSCHECK FILE=1,2
ADACHK DSCHECK FILE=17
ADACHK VALIDATE FILE=3
```

```
ADACHK ACCHECK
ADACHK CHECK FILE=21
ADACHK FSTPRINT
ADACHK FCBPRINT FILE=1
/*
//
```

IV ADACMP Utility: Compress-Decompress Data

This chapter covers the following topics:

Functional Overview

Input Data Requirements

Processing (including, among other things, *Segmented Record Considerations*, *Restart Considerations*, and *User Exit 6* explanations)

COMPRESS: Compress an Adabas File

DECOMPRESS: Decompress an Adabas File

Field Definition Statements

JCL Requirements and Examples

COMPRESS Function Output

DECOMPRESS Function Output

34 Functional Overview

- COMPRESS Function Overview 180
- DECOMPRESS Function Overview 181

This chapter provides an overview of the ADACMP utility.

COMPRESS Function Overview

The COMPRESS function edits and compresses data records that are to be loaded into the database:



Input can be data records from:

- a physical sequential data set (fixed- or variable-length records) supplied by the user; or
- an existing Adabas file (that is, from ADACMP DECOMPRESS or ADAULD UNLOAD).

Caution: ADACMP COMPRESS utility runs that specify an FDT (via the FDT parameter) but do not specify a FORMAT parameter and that run against a file with logically deleted fields require that the data include the values for the logically deleted fields. Failure to include these values could lead to incorrectly compressed records.

The logical structure and characteristics of the input data are described with *field definition statements*:

- The FNDEF statement is used to define a field (or group of fields).
- The SUBFN and SUPFN statements are used to define a subfield and a superfield, respectively.
- The COLDE, HYPDE, PHONDE, SUBDE, and SUPDE statements are used to define a collation descriptor, hyperdescriptor, phonetic descriptor, subdescriptor and superdescriptor, respectively.

The field definitions provided are used to create the Adabas field definition table (FDT) for the file. Alternatively, you can use an existing Adabas FDT instead of providing field definitions (read about the **FDT parameter**).

If the fields in the input record are to be processed in an order that is different from their position in the input record, or if one or more fields are to be skipped, the **FORMAT parameter** may be used to indicate the order and location of the input fields.

The ADACMP COMPRESS function processes the input data as follows:

- Checks numeric data for validity.
- Removes trailing blanks from alphanumeric fields.
- Removes leading zeros from numeric fields.
- Packs numeric unpacked fields.

Fields defined with the **fixed (FI) option** are not compressed.

A user exit can be used to further edit the input data. For more information, read ***User Exit 6***, elsewhere in this section.

The output of the ADACMP COMPRESS function that is used as input to the ADALOD utility includes the FDT, compressed records, and on the utility report, the Data Storage space requirement (for the ADALOD DSSIZE parameter) and the Temp and Sort data set size estimates (TEMPSIZE and SORTSIZE).

The ADACMP COMPRESS function report also indicates:

- the number of records processed;
- the number of records rejected; and
- the compression rate percentage.

A data set containing rejected records is also produced.

DECOMPRESS Function Overview

The DECOMPRESS function decompresses individual files:



Input data can be decompressed from data records in existing Adabas files:

- unloaded using the ADAULD (file unload) utility; or
- directly (without separate file unloading).

The INFILE parameter of ADACMP DECOMPRESS is used for Adabas files that are directly decompressed. As part of the decompression process, the target file is unloaded without FDT information, which can save time when decompressing larger files.

The output of the ADACMP DECOMPRESS function includes ISNs if the ISN parameter is specified. The DECOMPRESS output may be used as input to a non-Adabas program or as input to the COMPRESS function, once any desired changes to the data structure or field definitions for the file are completed.

35 Input Data Requirements

- Input Data Structure 184
- Multiple-Value Field Count 184
- Periodic Group Count 186
- System Field Requirements 189
- Variable-Length Field Size 190

This chapter describes the input data requirements of the ADACMP utility.

Input Data Structure

ADACMP input data must be in a sequential data set or file. Indexed sequential and VSAM input cannot be used.

The records may be fixed, variable, or of undefined length. The maximum input record length permitted depends on the operating system. The maximum compressed record length is restricted by the Data Storage block size in use and the maximum compressed record length set for the file (see the **MAXRECL parameter** of the ADALOD utility). The input records can be in either blocked or unblocked format.

The fields in each record must be structured according to the field definition statements provided (or the specified FDT if an existing Adabas FDT is being used). If a user exit routine is used, the structure following user exit processing must agree with the field definitions. Any trailing information contained in an input record for which a corresponding field definition statement is not present is ignored and is not included in the ADACMP output.

Data sets that contain no records are also permitted.

The input data set can be omitted if the parameter NUMREC=0 is supplied.

Multiple-Value Field Count

The number of values in each record's multiple-value field must either be specified in the field definition statement, or the value count must precede the values in each input record. When specified in the field definition statement, the minimum multiple-value count is 1, and the maximum is 65534 or 191, depending on the setting of the **MUPEX** and **MUPECOUNT** parameters of ADACMP COMPRESS. When the minimum count is specified in the input record, zero (0) can be specified to indicate a multiple-value field containing no values.

If the number of values is the same for each record, this number may be specified with the field definition statement for the multiple-value field (in the **occurrences** specification). In this case, the count byte in the input record must be omitted. If the record definitions are from an existing FDT or if the input data is from an earlier DECOMPRESS operation, the data already contains the length values; the count must not be specified in the **field definition statements**.

The count you specify may be changed by ADACMP if the **NU** option is specified for the field. ADACMP suppresses null values, and changes the count field accordingly. This is true whether you specify the value count before each series of values, or in the field definition statement. Refer to the section ***MU: Multiple-Value Field***.

Example 1: Multiple-Value Field Count with Varying Number of Occurrences

Field Definition:

```
ADACMP FNDEF='01,MF,5,A,MU,NU'
```

Each record contains a different number of values for MF, and the count comes before each series of occurrences.

	Before ADACMP	After ADACMP
Input Record 1 (3 values)	MF count=3	MF count=3
	AAAA	AAAA
	BBBB	BBBB
	CCCC	CCCC
Input Record 2 (2 values)	MF count=2	MF count=2
	AAAA	AAAA
	BBBB	BBBB
Input Record 3 (3 values)	MF count=3	MF count=2
	AAAA	AAAA
	bbbb	CCCC
	CCCC	
Input Record 4 (no values)	MF count=0	MF count=0
Input Record 5 (1 value)	MF count=1	MF count=0
	bbbb	

Example 2: Multiple-Value Field Count with Same Number of Occurrences

Field Definition:

```
ADACMP FNDEF='01,MF,4,A,MU(3),NU'
```

Each record contains 3 values for MF, as specified in the field definition statement.

	Before ADACMP	After ADACMP
Input Record 1		MF count=3
	AAAA	AAAA
	BBBB	BBBB
	CCCC	CCCC
Input Record 2		MF count=2
	AAAA	AAAA
	BBBB	BBBB

	Before ADACMP	After ADACMP
	bbbb	
Input Record 3		MF count=2
	AAAA	AAAA
	bbbb	CCCC
	CCCC	
Input Record 4		MF count=0
	bbbb	
	bbbb	
	bbbb	

Periodic Group Count

Each periodic group must specify a count of field iterations (occurrences) in the record. The count is specified either within the field definition statement for all records, or as a one- or two-byte binary value (depending on the value of the **MUPECOUNT** parameter) before each occurrence group in every record. If the count is in the field definition statement, the count byte must be omitted from the input records. When specified in the field definition statement, the minimum count allowed is 1, and the maximum is 65534 or 191, depending on the setting of the **MUPEX** and **MUPECOUNT** parameters of ADACMP COMPRESS.. When the minimum count is specified in the record, the value can be zero (0) for a periodic group with no occurrences.

The occurrence count provided may be modified by ADACMP if all the fields contained in the periodic group are defined with the **NU** option. If all the fields within a given occurrence contain null values and there are no following occurrences that contain non-null values, the occurrence will be suppressed and the periodic group occurrence count will be adjusted accordingly.

Example 1: Periodic Group Count with Varying Number of Occurrences

Field Definitions:

```
ADACMP COMPRESS
ADACMP FNDEF='01,GA,PE'
ADACMP FNDEF='02,A1,4,A,NU'
ADACMP FNDEF='02,A2,4,A,NU'
```

The input records contain a variable number of occurrences for GA (up to 191 occurrences are permitted as the **MUPEX** parameter is not specified causing the **MUPECOUNT** parameter to default to "1"). The count of occurrences comes before each occurrence group in the input records.

	Before ADACMP	After ADACMP
Input Record 1	GA count=2	GA count=2
	GA (1st occurrence)	
	A1=AAAA	A1=AAAA
	A2=BBBB	A2=BBBB
	GA (2nd occurrence)	
	A1=CCCC	A1=CCCC
	A2=DDDD	A2=DDDD
Input Record 2	GA count=1	GA count=0
	GA (1st occurrence)	
	A1=bbbb	suppressed
	A2=bbbb	suppressed
Input Record 3	GA count=3	GA count=3
	GA (1st occurrence)	
	A1=AAAA	A1=AAAA
	A2=bbbb	A2=suppressed
	GA (2nd occurrence)	
	A1=BBBB	A1=BBBB
	A2=bbbb	A2=suppressed
	GA (3rd occurrence)	
	A1=CCCC	A1=CCCC
	A2=bbbb	A2=suppressed
Input Record 4	GA count=0	GA count=0

Example 2: Periodic Group Count with Same Number of Occurrences

Field Definitions:

```
ADACMP FNDEF='01,GA,PE(3)'
ADACMP FNDEF='02,A1,4,A,NU'
ADACMP FNDEF='02,A2,4,A,NU'
```

All input records contain 3 occurrences for GA, as specified in the field definition statement.

	Before ADACMP	After ADACMP
Input Record 1		GA count=3
	GA (1st occurrence)	
	A1=AAAA	A1=AAAA
	A2=bbbb	A2 suppressed
	GA (2nd occurrence)	
	A1=BBBB	A1=BBBB
	A2=bbbb	A2 suppressed
	GA (3rd occurrence)	
	A1=CCCC	A1=CCCC
	A2=bbbb	A2 suppressed
Input Record 2		GA count=2 (see note)
	GA (1st occurrence)	
	A1=bbbb	A1=suppressed
	A2=bbbb	A2=suppressed
	GA (2nd occurrence)	
	A1=BBBB	A1=BBBB
	A2=bbbb	A2=suppressed
	GA (3rd occurrence)	
	A1=bbbb	A1=suppressed
	A2=bbbb	A2=suppressed
Input Record 3	All occurrences contain null values	GA count=0 All occurrences are suppressed

Note: The first occurrence is included in the count since occurrences follow that contain non-null values. The third occurrence is not included in the count since there are no non-null values in the occurrences that follow.

Example 3: Adding a Field to a PE-Group

In the PE named AW, the field AY should be added:

Old FDT	New FDT
01 AA,8,A,DE,UQ	01 AA,8,A,DE,UQ
01 AW,PE	01 AW,PE
02 AX,8,U,NU	02 AX,8,U,NU
02 AT,8,U,NU	02 AT,8,U,NU
01,AZ,3,A,DE,MU,NU	02 AY,8,U,NU

Old FDT	New FDT
	01,AZ,3,A,DE,MU,NU

Note: All of the currently existing fields in the PE must be specified.

1. Determine the maximum occurrence of the PE (for example, a result of 2).
2. Decompress the file with the format parameter.
3. Decompress INFILE=xx,FORMAT='AA,AX1-2,AT1-2,AZ'
4. Compress again:

```
ADACMP COMPRESS FILE=32
ADACMP FORMAT='AA,AX1-2,AT1-2,AZ'
ADACMP FNDEF='01,AA,8,A,DE,UQ'
ADACMP FNDEF='01,AW,PE(2)'
ADACMP FNDEF='02,AX,8,U,NU'
ADACMP FNDEF='02,AT,8,U,NU'
ADACMP FNDEF='02,AY,8,U,NU'
ADACMP FNDEF='01,AZ,3,A,DE,MU,NU'
```

System Field Requirements

The ADACMP utility treats system fields like normal fields and compresses data according to the **FNDEF definition** of the field. This means that system field values must be provided.

If the records are to be initially loaded into the database you may find it easiest to set the number of MU occurrences of system fields to "1". In the following example, this is accomplished by the `MU(1)` settings in the FNDEF definitions for the job name (`SY=JOBNAME`) and time (`SY=TIME`) system fields:

```
//DDKARTE DD *
ADACMP COMPRESS
ADACMP TZ='America/New_York'
ADACMP FNDEF='01,PA,8,A,NU,DE'
ADACMP FNDEF='01,JB,8,A,NU,MU(1),SY=JOBNAME'
ADACMP FNDEF='01,TZ,14,U,NU,MU(1),DT=E(DATETIME),TZ,SY=TIME'
ADACMP FNDEF='01,PL,8,A,NU'

//DDEBAND  DD  *                                                    ↩

CHAIR   CMPJOB  20100518142915Wood
DESK    CMPJOB  20100518142915Plastic
LAMP    CMPJOB  20100518142915Metall              ↩
```

Alternatively, you can use the FORMAT keyword to omit system fields in the format buffer. In the following example, any system field will get the default empty value.

```
ADACMP FORMAT='PA,PL.'
//DDEBAND  DD  *
CHAIR   Wood
DESK    Plastic
LAMP    Metall
//                                                              ↩
```

Variable-Length Field Size

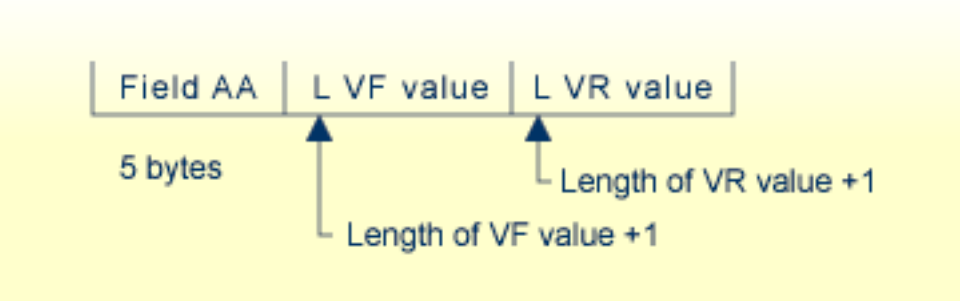
Each value of a variable-length field (length parameter not specified in the field definition) must be preceded by a one-byte binary count indicating the value length (including the length byte itself). An LA field specified with variable length (i.e., length 0), must be preceded by a two-byte inclusive length. An LB field specified with variable length must be preceded by a four-byte inclusive length.

Example of Variable-Length Field Size

Field Definitions:

```
ADACMP FNDEF='01,AA,5,A,DE'
ADACMP FNDEF='01,VF,0,A'
ADACMP FNDEF='01,VR,0,A'
```

Input record:



36 Processing

- Segmented Record Considerations 192
- Logically Deleted Fields 198
- Data Verification 198
- Data Compression 198
- Representation of LOB Values and Value References in Uncompressed Data 200
- Identifying MU and PE Occurrences Greater Than 191 in Compressed Records 201
- Restart Considerations 201
- User Exit 6 201

This chapter describes ADACMP utility processing.

Segmented Record Considerations

If a decompressed record from ADACMP is too long to fit into the longest record length allowed for a sequential data set (32KB or less), ADACMP can segment it into multiple physical records. A single logical decompressed record can span one or more physical decompressed records.

In addition, the ADACMP utility allows you to create special headers, ADAH and ADAC, in the decompressed output. These special headers are used only with ADACMP processing. They identify the position of the payload data in the logical record as well as the relation between the physical record and the other physical records in the same logical record. When created, an ADAH header is used for the first physical record of a logical record; ADAC headers are used for the second and subsequent physical records that comprise the logical record. If a decompressed logical record does not need to be segmented (if it fits into one physical record), only an ADAH header is created; there is no need for ADAC headers.

Do not confuse the record segmentation that occurs with ADACMP decompression logic and *record spanning*. Spanned records also consist of multiple physical records (one primary record and multiple secondary records), but they are compressed records. In addition, each spanned record is automatically assigned a standard spanned record header that is not the same as the ADAH and ADAC headers you can create for decompressed records using ADACMP; segmented records produced by ADACMP do not contain the standard spanned record header. For complete information about spanned records, read *Spanned Records* in *Adabas Concepts Manual*.

This section covers the following topics:

- Creating and Supporting ADACMP Headers
- ADAH and ADAC Header Descriptions

Creating and Supporting ADACMP Headers

The HEADER parameter of ADACMP DECOMPRESS controls whether the decompression logic produces the headers in its *output*. The HEADER parameter of ADACMP COMPRESS controls whether the compression logic will accept the ADACMP headers as part of the uncompressed *input*.

ADAH and ADAC Header Descriptions

This section covers the following topics:

- ADAH Headers
- ADAC Headers
- Example

ADAH Headers

When ADACMP headers are used, the first physical record of a logical record begins with an ADAH header containing the following information:

- The characters "ADAH".
- The length of the ADAH header.
- A continuation indicator that indicates whether this is the last physical record in the logical record or whether another physical record for the same logical record will follow this one.
- The total length of the record (with the headers). The value may be zero if the total record length is not known when the first physical record is written.
- The length of the payload data (a segment of the logical record) in this physical record. This refers to the length of the payload data; it does not include the length of the ADAH header. The length must be less than or equal to the length of the physical record minus the length of the header. If it is less than this value, any extra data in the physical record (not covered by the payload data length) is ignored.

The payload data follows the ADAH header.

The ADAH DSECT can be found in the ADAH member of the distributed Adabas source library.

ADAC Headers

When ADACMP decompressed record segmenting occurs and when ADACMP headers are requested, the second and every subsequent physical record for a logical record begins with an ADAC header containing the following information:

- The characters "ADAC".
- The length of the ADAC header.
- A continuation indicator that indicates whether this is the last physical record in the logical record or whether another physical record for the same logical record will follow this one.
- The sequence number of this secondary record in the logical record. The second physical record of a logical record is the first secondary record and therefore has a sequence number of "1". The sequence numbers are in ascending order, without gaps.

- The offset within the logical record of the payload data (segment) contained in this physical record. This offset is the sum of the payload data lengths of each prior physical record in the logical record.
- The length of the payload data (segment) in this physical record. This refers to the length of the payload data; it does not include the length of the ADAC header. The length must be less than or equal to the length of the physical record minus the length of the header. If it is less than this value, any extra data in the physical record (not covered by the payload data length) is ignored.

The payload data follows the ADAC header.

The ADAC DSECT can be found in the ADAC member of the distributed Adabas source library.

Example

The following table depicts three logical records spanning seven physical records of uncompressed data.



Note: DSECTs for the ADAH and ADAC headers can be found in members ADAH and ADAC in the distributed Adabas source library.

Logical Record	Physical Record Headers	Header Fields		Description
		Field	Value	
1	ADAH	ADAHEYE	ADAH	ADAH header eyecatcher
		ADAHLEN	32	ADAH header length
		ADAHIND	C	Continuation indicator. Valid values are: C: Continuation record segment to follow E: End of logical record (last segment)
		Reserved	0	Must contain binary zeros.
		ADAHTLEN	50000	Total length of logical record. This value may be zero if the total length is not known when the first segment is written.
		Reserved	0	Reserved
		ADAHSLEN	27962	Length of this segment (length of the payload data). The sum of the values of ADAHLEN and ADAHSLEN is the minimum length of the physical record. The physical record can be longer than this; in this case, the excess data has no meaning and is ignored.
		ADAHDATA	'Record 1 - payload data part 1'	Payload data
	ADAC	ADACEYE	ADAC	ADAC header eyecatcher
		ADACLEN	32	ADAC header length

Logical Record	Physical Record Headers	Header Fields		Description
		Field	Value	
		ADACIND	E	Continuation indicator. Valid values are: C: Continuation record segment to follow E: End of logical record (last segment)
		Reserved	0	Reserved
		ADACSEQ	1	Continuation record sequence number within the logical record (the first ADAC record has a sequence number of "1").
		ADACOFFS	27962	Segment offset within the logical record (the first payload data byte is at offset "0").
		Reserved	0	Reserved
		ADACSLEN	22038	Length of this segment (length of the payload data). The sum of the values of ADACLEN and ADACSLEN is the minimum length of the physical record. The physical record can be longer than this; in this case, the excess data has no meaning and is ignored.
		ADACDATA	'Record 1 - payload data part 2'	Continuation record payload data
2	ADAH	ADAHEYE	ADAH	ADAH header eyecatcher
		ADAHLEN	32	ADAH header length
		ADAHIND	E	Continuation indicator. Valid values are: C: Continuation record segment to follow E: End of logical record (last segment)
		Reserved	0	Must contain binary zeros.
		ADAHTLEN	25000	Total length of logical record. This value may be zero if the total length is not known when the first segment is written.
		Reserved	0	Reserved
		ADAHSLEN	25000	Length of this segment (length of the payload data). The sum of the values of ADAHLEN and ADAHSLEN is the minimum length of the physical record. The physical record can be longer than this; in this case, the excess data has no meaning and is ignored.
		ADAHDATA	'Record 2 - payload data'	Payload data
3	ADAH	ADAHEYE	ADAH	ADAH header eyecatcher
		ADAHLEN	32	ADAH header length

Logical Record	Physical Record Headers	Header Fields		Description
		Field	Value	
		ADAHIND	C	Continuation indicator. Valid values are: C: Continuation record segment to follow E: End of logical record (last segment)
		Reserved	0	Must contain binary zeros.
		ADAHTLEN	0	Total length of logical record. This value may be zero if the total length is not known when the first segment is written (as is the case for logical record 3).
		Reserved	0	Reserved
		ADAHSLEN	27962	Length of this segment (length of the payload data). The sum of the values of ADAHLEN and ADAHSLEN is the minimum length of the physical record. The physical record can be longer than this; in this case, the excess data has no meaning and is ignored.
		ADAHDATA	'Record 3 - payload data part 1'	Payload data
	ADAC	ADACEYE	ADAC	ADAC header eyecatcher
		ADACLEN	32	ADAC header length
		ADACIND	C	Continuation indicator. Valid values are: C: Continuation record segment to follow E: End of logical record (last segment)
		Reserved	0	Reserved
		ADACSEQ	1	Continuation record sequence number within the logical record (the first ADAC record has a sequence number of "1").
		ADACOFFS	27962	Segment offset within the logical record (the first payload data byte is at offset "0").
		Reserved	0	Reserved
		ADACSLEN	27962	Length of this segment (length of the payload data). The sum of the values of ADACLEN and ADACSLEN is the minimum length of the physical record. The physical record can be longer than this; in this case, the excess data has no meaning and is ignored.
		ADACDATA	'Record 3 - payload data part 2'	Continuation record payload data
	ADAC	ADACEYE	ADAC	ADAC header eyecatcher
		ADACLEN	32	ADAC header length

Logical Record	Physical Record Headers	Header Fields		Description
		Field	Value	
		ADACIND	C	Continuation indicator. Valid values are: C: Continuation record segment to follow E: End of logical record (last segment)
		Reserved	0	Reserved
		ADACSEQ	2	Continuation record sequence number within the logical record (the first ADAC record has a sequence number of "1").
		ADACOFFS	55924	Segment offset within the logical record (the first payload data byte is at offset "0").
		Reserved	0	Reserved
		ADACSLEN	27962	Length of this segment (length of the payload data). The sum of the values of ADACLEN and ADACSLEN is the minimum length of the physical record. The physical record can be longer than this; in this case, the excess data has no meaning and is ignored.
		ADACDATA	'Record 3 - payload data part 3'	Continuation record payload data
	ADAC	ADACEYE	ADAC	ADAC header eyecatcher
		ADACLEN	32	ADAC header length
		ADACIND	E	Continuation indicator. Valid values are: C: Continuation record segment to follow E: End of logical record (last segment)
		Reserved	0	Reserved
		ADACSEQ	3	Continuation record sequence number within the logical record (the first ADAC record has a sequence number of "1").
		ADACOFFS	83886	Segment offset within the logical record (the first payload data byte is at offset "0").
		Reserved	0	Reserved
		ADACSLEN	16114	Length of this segment (length of the payload data). The sum of the values of ADACLEN and ADACSLEN is the minimum length of the physical record. The physical record can be longer than this; in this case, the excess data has no meaning and is ignored.
		ADACDATA	'Record 3 - payload data part 4'	Continuation record payload data

Logically Deleted Fields

ADACMP COMPRESS utility runs that specify an FDT (via the FDT parameter) but do not specify a FORMAT parameter and that run against a file with logically deleted fields (see the ADADBS DELFN utility function) require that the data include the values for the logically deleted fields. Failure to include these values could lead to incorrectly compressed records.

Data Verification

ADACMP checks each field defined with format P (packed) or U (unpacked) to ensure that the field value is numeric and in the correct format. If a value is empty, the null characters must correspond to the format specified for the field (see *Representing SQL Null Values* in the *Field Definition Statements* section.

Alphanumeric (A)	blanks (hex '40')
Binary (B)	binary zeros (hex '00')
Fixed (F)	binary zeros (hex '00')
Floating Point (G)	binary zeros (hex '00')
Packed (P)	decimal packed zeros with sign (hex '00' followed by '0F', '0C', or '0D' in the rightmost, low-order byte)
Unpacked (U)	decimal unpacked zeros with sign (hex 'F0' followed by 'C0' or 'D0' in the rightmost, low-order byte)

Any record that contains invalid data is written to the ADACMP error (DDFEHL) data set and is not written to the compressed data set.

Data Compression

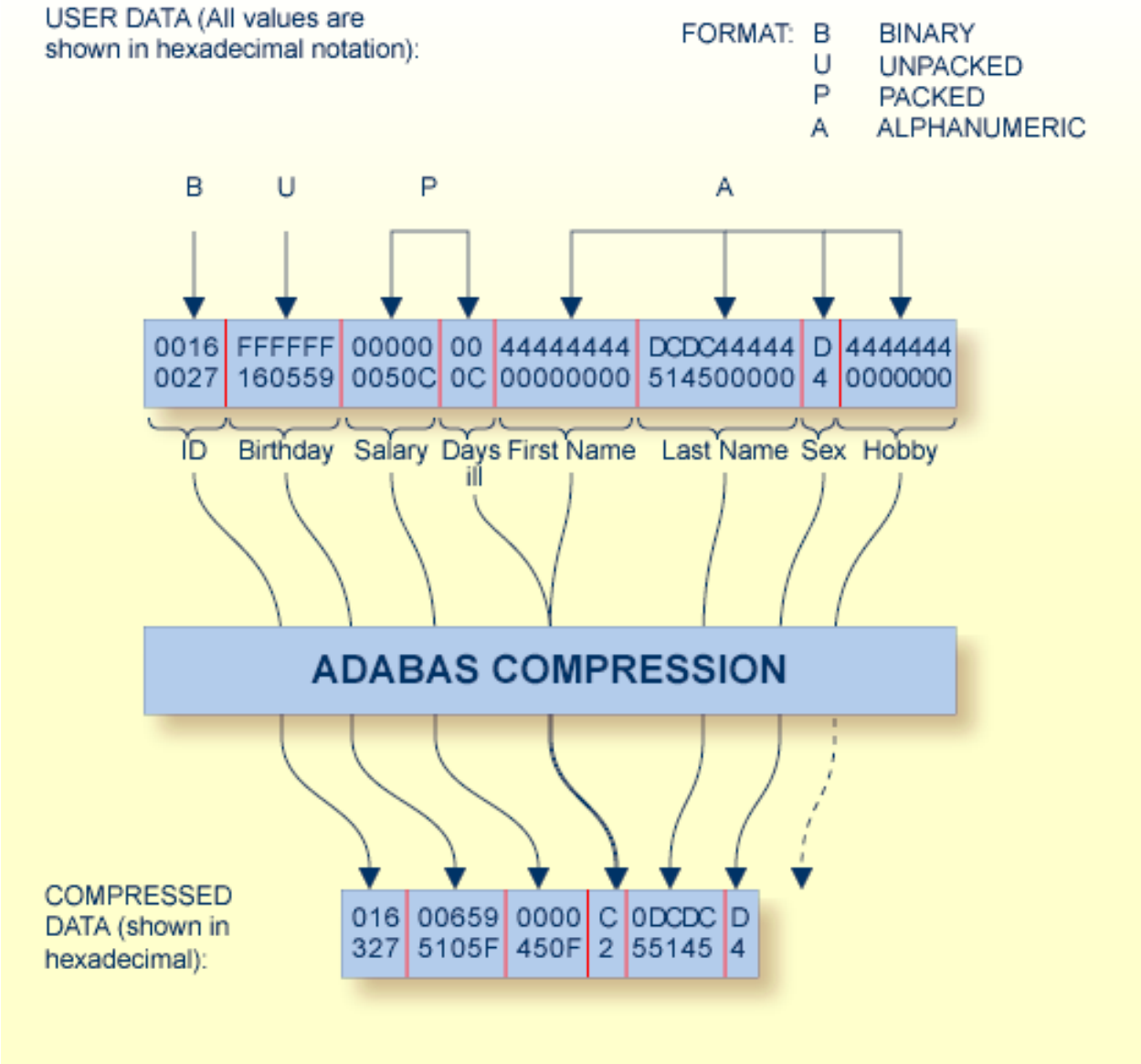
The value for each field is compressed (unless the **FI option** is specified) as follows:

- Trailing blanks are removed for fields defined with A format.
- Leading zeros are removed for numeric fields (fields defined with B, F, P or U format).
- If the field is defined with U (unpacked) format, the value is converted to packed (P) format.
- Trailing zeros in floating-point (G format) fields are removed.
- If the field is defined with the NU option and the value is a null value, a one-byte indicator is stored. Hexadecimal 'C1' indicates one empty field follows, 'C2' indicates that two empty fields follow, and so on, up to a maximum of 63 before the indicator byte is repeated. For SQL null

value (NC option field) compression, see *Representing SQL Null Values* in the *Field Definition Statements* section.

- Empty fields located at the end of the record are not stored, and therefore not compressed.

Example of Data Compression



The graphic shows how the following field definitions and corresponding values would be processed by ADACMP:

```
FNDEF='01,ID,4,B,DE'
FNDEF='01,BD,6,U,DE,NU'
FNDEF='01,SA,5,P'
```

```
FNDEF='01,DI,2,P,NU'
FNDEF='01,FN,9,A,NU'
FNDEF='01,LN,10,A,NU'
FNDEF='01,SE,1,A,FI'
FNDEF='01,HO,7,A,NU'
```

Representation of LOB Values and Value References in Uncompressed Data

This section describes how large object (LB) field values, LB field value references, and logical records that are longer than 32 KB must be represented in the input data set for the ADACMP COMPRESS function and how these items are represented in the output data set of the ADACMP DECOMPRESS function.

- Large Object (LB) Field Values
- Large Object (LB) Field Value References

Large Object (LB) Field Values

If ADACMP is run without the FORMAT parameter, each large object (LB) field value in the uncompressed data is preceded by a 4-byte length field. The length value includes the length of the LB field value proper plus four bytes for the length field itself. An empty LB field value for a field defined *without* the NB option consists of the length field with a value of 5 and a single blank; for a field defined *with* the NB option, an empty LB field value consists only of the length field with a value of 4.

If ADACMP COMPRESS is used to define an FDT with LB fields, each LB field value in the uncompressed input must be less than or equal to 253 bytes.

Large Object (LB) Field Value References

When the ADACMP DECOMPRESS function is run with LOBVALUES=NO to decompress only the records from the *base file* of a *LOB file group*, omitting all LB field values stored in the associated *LOB file*, each reference in a base file record to a LB field value in the LOB file is represented in the uncompressed output as follows:

- The four-byte length field for the LB field value contains X'FFFFFFFF' (high value) to indicate the presence of the reference to an LB field value.
- The indicator is followed by a two-byte inclusive length field for the LB field value reference. The length value includes the length of the LB field value reference proper plus two bytes for the length field itself.
- The length field is followed by the LB field value reference proper.

The same structure is expected by the ADACMP COMPRESS function with LOBVALUES=NO in the place of an LB field value that is stored in the *LOB file* associated with the *base file* that is being compressed.

LB field value references that are input to ADACMP COMPRESS must originate from ADACMP DECOMPRESS. There is no sensible way to introduce new LB field value references using COMPRESS, as they would not properly refer to existing LB field values in a *LOB file*.

Identifying MU and PE Occurrences Greater Than 191 in Compressed Records

MU and PE occurrences greater than 191 are indicated in compressed records by a x'C0' byte at the beginning of the occurrence count. This byte is set by the ADACMP utility or the nucleus when the records are compressed. The x'C0' indicator byte is followed by a byte indicating the number of count bytes used for the MU or PE occurrence count that follows. For example, consider the following indicator:

```
X'C0020204'
```

In this example, x'C0' indicates this is an extended count; x'02' indicates that there are two count bytes, and x'0204' indicates that there are 516 occurrences of the field.

Restart Considerations

ADACMP has no restart capability. An interrupted ADACMP execution must be reexecuted from the beginning.

User Exit 6

A user-written routine called user exit 6 can be used for editing during ADACMP COMPRESS processing. The routine may be written in Assembler or COBOL. It must be assembled or compiled and then linked into the Adabas load library (or any library concatenated with it).

User exit 6 is invoked by specifying:

```
ADARUN UEX6=program
```

where *program* is the routine name in the load library.

For specific information about the user exit 6 structure and parameters, read *User Exits and Hyperdescriptor Exits* in *Adabas User, Hyperdescriptor, Collation Descriptor, and SMF Exits Manual*.

37 COMPRESS: Compress an Adabas File

- Essential Parameters and Subparameters 204
- Optional Parameters and Subparameters 206
- ADACMP COMPRESS Examples 214

```
ADACMP  COMPRESS  {field-definition-statements | FDT = file-number}
                  [CODE = cipher-code ]
                  [DATADEVICE = device-type ]
                  [DEVICE = device-type-list ]
                  [FACODE = file-alpha-EBCDIC-key ]
                  [FILE = {file-number | 0 } ]
                  [FWCODE = file-wide-key ]
                  [FUWCODE = wide-key ]
                  [FORMAT = format ]
                  [HEADER = { YES | NO } ]
                  [LOBDEVICE = device-type-list ]
                  [LOBVALUES = { YES | NO } ]
                  [MAXLOGRECLEN = buffer-size ]
                  [MUPECOUNT = 1 | 2 ]
                  [MUPEX]
                  [NOUSERABEND]
                  [NUMREC = number-of-records]
                  [PASSWORD = "password" ]
                  [RECFM = { F | FB | V | VB | U } ]
                        [,LRECL = record-length } ]
                  [SPAN]
                  [TZ = ' timezone-name ' [,DST] ]
                  [ { USERISN | MINISN = { start-isn | 1 } } ]
                  [UACODE = userdata-alpha-key ]
                  [UARC = {userdata-architecture-key | 2 } ]
                  [UWCODE = userdata-wide-key ]
```

This chapter describes the syntax and parameters of the ADACMP COMPRESS function.

Essential Parameters and Subparameters

field-definition-statements

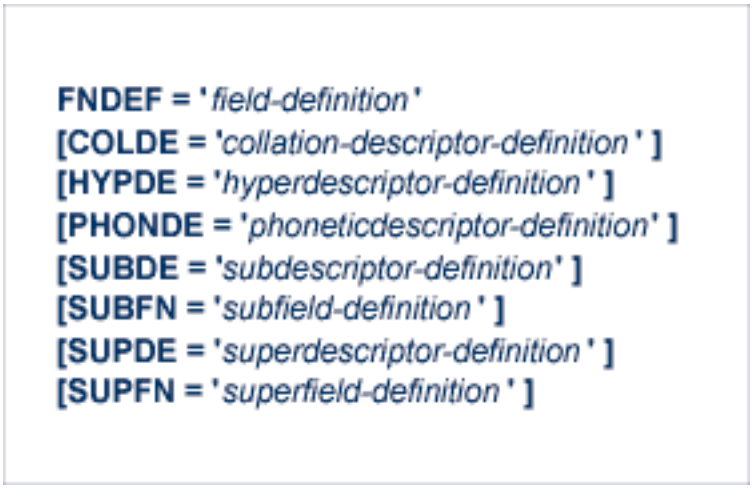
Field definition statements, when provided as input to ADACMP, are used to:

- provide the length and format of each field contained in the input record. This enables ADACMP to determine the correct field length and format during editing and compression.

- create the Field Definition Table (FDT) for the file. This table is used by Adabas during the execution of Adabas commands to determine the logical structure and characteristics of any given field (or group) in the file.

Either an FDT parameter or field definition statements must be supplied for ADACMP COMPRESS. If both are supplied, the field definition statements are ignored.

The field definition statements that can be included in this syntax:



For complete information on field definition statements, including their syntax, read *Field Definition Statements*, elsewhere in this section.

FDT: Use Existing Adabas Field Definition Table

Specifies an existing Adabas FDT to be used. The FDT may be that of an existing file or a file that has been deleted with the KEEPFDT option of the ADADBS utility.

Either an FDT parameter or field definition statements must be supplied for ADACMP COMPRESS. If both are supplied, the field definition statements are ignored.

If the FDT parameter is specified, the input data must be consistent with the structure as defined in the specified FDT, unless the FORMAT parameter is used. When the FDT defines multiple-value fields or periodic groups, length values must be defined or already included in the FDT. Read sections *Multiple-Value Field Count* and *Periodic Group Field Count*.

If the FDT parameter is used, any field definitions specified will be ignored.

Caution: ADACMP COMPRESS utility runs that specify an FDT (via the FDT parameter) but do not specify a FORMAT parameter and that run against a file with logically deleted fields require that the data include the values for the logically deleted fields. Failure to include these values could lead to incorrectly compressed records.

Optional Parameters and Subparameters

CODE: Cipher Code

If the data is to be loaded into the database in ciphered form, the cipher code must be specified with this parameter. See the *Adabas Security* documentation for additional information on the use of ciphering.



Note: You cannot specify the CODE parameter in a ADACMP COMPRESS function if the file contains LB fields.

DATADEVICE: Device Type

The DATADEVICE parameter specifies the Data Storage device type to be used for the segmentation of spanned records. If the SPAN parameter is specified, ADACMP will break long, spanned, compressed records into segments that are just a bit smaller than the Data Storage block size implied by the DATADEVICE parameter.

If the SPAN parameter is not specified, no value for DATADEVICE is required. However, it can be specified to limit the size of compressed records. In this case, all records that exceed the given storage device block size will be written to the DDFEHL error data set.

DEVICE: Device Type

If the DEVICE parameter is specified, ADACMP calculates and displays a report of this run's space requirements for each specified device type. This report includes an indication of whether or not the `MUPEX` parameter has been set for a file.

If no device types are listed on the DEVICE parameter, the ADARUN device type is used as the default.

DST: Daylight Savings Indicator

The DST parameter can be specified to indicate that date-time data includes a daylight savings time indicator. If a time zone uses daylight savings time, you must be sure to store and retrieve the daylight savings indicator with your date-time data or there will be no way to distinguish date-time values in the hour before the time is switched back to standard time. The two-byte daylight savings indicator directly follows the date-time value in uncompressed input and specifies the binary integer value of the daylight saving time offset from standard time in seconds.

You must specify a DST parameter for files containing date-time fields defined with option TZ, when the time zone includes a daylight savings time indicator. If the DST parameter is not specified, date-time data is stored without a daylight savings time indicator. The default is store the date-time data without a daylight savings time indicator.

The DST parameter requires that the **TZ parameter** be set in the same ADACMP run. However, it should *not* be specified when the FORMAT parameter is specified in the run – use format buffer element `fnD` to define the position of a daylight savings indicator for a given field in the input data.

FACODE: Alphanumeric Field Encoding

FACODE must be specified if you want to define UES file encoding for alphanumeric fields in the file. The alphanumeric encoding must belong to the EBCDIC encoding family; that is, the space character is X'40'.

FILE: File Number

If the FDT contains a hyperdescriptor, this parameter must be specified. The specified file number becomes input for the related hyperdescriptor exit. For more information about hyperdescriptor exits, refer to the *Adabas DBA Reference* documentation.

User exit 6 is always supplied with this file number. If FILE is not specified, a value of zero is assumed.

FORMAT: Input Record Format Definition

Use this parameter to provide a format definition that indicates the location, format, and length of fields in the input record. The format provided must follow the rules for format buffer entries for update commands as described in the *Adabas Command Reference Guide* documentation.

Conversion rules are those described for Adabas update commands in the *Adabas Command Reference Guide* documentation. For conversion of SQL null (NC option) field values, see ***NC: SQL Null Value Option***. If a field is omitted in the FORMAT parameter, that field is assigned no value.

If the FORMAT parameter is omitted, the input record is processed in the order of the field definition statements provided or, if the FDT parameter is used, according to an existing Adabas field definition table.

LB (large object) fields may be specified in the definition supplied in the FORMAT parameter if the **LOBVALUES** parameter is set to NO. This functionality is most useful for changing the FDT of a base file in a LOB (large object) file group. In this case, the LB field values stored in the LOB file are not modified.

The following restrictions apply to LB fields specified in a FORMAT parameter:

- An LB field cannot be specified more than once.
- An LB field cannot be specified with a length or asterisk (*) length notation.
- LOB segment notation is not allowed.
- Length indicator (L) notations are not allowed for LB fields



Caution: Care must be taken when specifying LB fields in a FORMAT parameter if the LOBVALUES parameter is set to NO. Failure to specify all LB fields could lead to LB values existing in the LOB file without a corresponding reference in the base file.

FUWCODE: Wide-Character Field Default User Encoding

FUWCODE defines the default user encoding for wide-character fields for the file when loaded in the database. If this parameter is omitted, the encoding is taken from the UWCODE definition of the database.

FWCODE: Wide-Character Field Encoding

If fields with format W (wide-character) exist in the compressed file, you *must* specify FWCODE to define the file encoding for them.

FWCODE also determines the maximum byte length of the wide-character field.

HEADER

This optional parameter indicates whether or not the ADACMP compression logic should expect segmented ADACMP record headers in the uncompressed input records. Valid values are YES or NO; the default is NO.

HEADER=NO is the format accepted and produced by ADACMP in Adabas versions prior to Adabas 8. When it is specified, the input records must contain only the uncompressed data for the fields of the file being processed. Each data record must fit into one physical record of the sequential input data set (less than 32 KB). If the data exceeds this size, the records in error are written to the DDFEHL error data set.

HEADER=YES can only be specified if you are running Adabas 8 or later Adabas versions. If HEADER=YES is specified, each input record must begin with either an ADAH or ADAC header, relating the physical segmented record with a logical record to be processed by ADACMP. Each logical record can be larger than 32 KB. The header in each physical record defines the position of the data following it within the logical record. DSECTs for the ADAH and ADAC headers can be found in members ADAH and ADAC of the distributed Adabas SRCE data set.

If HEADER=YES is specified, an error may occur while segmented uncompressed records are being assembled into a logical record. If the ADAH header is in error, the ADAH record is written and subsequent ADAC records are not written until the next ADAH record is processed. If an ADAC header is in error, the preceding ADAH header will be written without its payload data. The ADAC record in error will be written in its entirety. Subsequent ADAC records are not written until the next ADAH record is processed. For more information about rejected records and possible response codes resulting from them, read ***COMPRESS Function Output***, elsewhere in this section.

Do not confuse the HEADER parameter with the SPAN parameter. The SPAN parameter controls whether the compressed records themselves should be spanned if they exceed the Data Storage block size of the device. Spanned records contain a standard spanned record header that differs from the ADAH or ADAC headers expected by the HEADER parameter. For complete information about spanned records, read *Spanned Records* in *Adabas Concepts Manual*.

LOBDEVICE: Device Type for LOB File

This optional parameter specifies the data storage device type that will be used for loading the *LOB file* produced by the ADACMP COMPRESS function. ADACMP will divide the LB field values into segments based on the block size of the specified device. This parameter is only valid if the FDT includes one or more large object (LB option) fields.

If LOBDEVICE is not specified, the device specified for the ADACMP COMPRESS DEVICE parameter is used. If no device is specified for the DEVICE parameter either, the value of the ADARUN DEVICE parameter is used.

If the LOBDEVICE parameter is specified, ADACMP calculates and displays a report of the space requirements for each specified device type. Since the LOB fields are split into segments with the maximum possible size for the first device specified in LOBDEVICE, you should not use the output for loading with any other device specified in LOBDEVICE. Here is an example of the output produced for LOB files:

```
Large Objects Statistics:
-------------------------

Number of bytes for largest LOB              567,041
Total number of LOBs                           2,572
Total number of outsourced LOBs                   60
Number of LOB file records                       571


DATASTORAGE Space requirements for Base File
--------------------------------------------

I  Device  I  Padding I  Blocksize I             Number of         I
I          I  Factor  I   (bytes)  I         Blocks         Cyls   I
I----------I----------I-----------I--------------------------------I
I   8391   I          I   10,796   I                                I
I          I     5%   I   10,255   I             31            1   I
I          I    10%   I    9,715   I             33            1   I
I          I    15%   I    9,175   I             35            1   I
I          I    20%   I    8,637   I             37            1   I
I          I    25%   I    8,097   I             39            1   I
I          I    30%   I    7,556   I             41            1   I
I          I          I            I                                I
I----------I----------I-----------I--------------------------------I


DATASTORAGE Space requirements for LOB File
-------------------------------------------


I  Device  I   Blocksize I             Number of        I
I          I    (bytes)  I         Blocks          Cyls I
I----------I------------I-----------------------------I
I   8391   I    10,796   I            571             8 I
```

```
I   3390   I     5,064   I          1,185             8 I
I   8393   I    27,644   I            241             9 I
I----------I-------------I------------------------------I

Note: LOB fields were split into segments with the maximum possible size for
      the first LOBDEVICE. Do not use the output for loading with the
      other LOBDEVICEs.


TEMP Space Estimation
---------------------


I  Device  I   Blocksize I             Number of        I
I          I    (bytes)  I         Blocks          Cyls I
I----------I-------------I------------------------------I
I   8391   I    18,452   I             22             1 I
I----------I-------------I------------------------------I
```

LOBVALUES: LB Field Size Indicator

This optional parameter indicates whether long LB field values (larger than 253 bytes) or short LB field values (up to 253 bytes) are expected in the ADACMP COMPRESS input data. Valid values for this parameter are "YES" and "NO"; the default is "NO".

If "YES" is specified for this parameter, the uncompressed input data may contain LB field values larger than 253 bytes. In this case, a second sequential output data set must also be supplied in the JCL for the run. This second data set is identified in the JCL using the DD control statement DDAUSB1. It is used to store the compressed LB segment records for LB field values that are larger than 253 bytes.

If "NO" is specified for this parameter, the uncompressed input data may contain only LB fields up to 253 bytes long and references to LB field values stored in a *LOB file*. In this case, you cannot specify an LB field in the ADACMP COMPRESS FORMAT parameter. During processing, ADACMP writes any short LB field values and LB field value references contained in the input to the output.

Note: An ADACMP DECOMPRESS function with LOBVALUES=NO followed by an ADACMP COMPRESS function with LOBVALUES=NO can be used to modify the FDT of the *base file* in the *LOB file group*.

MAXLOGRECLEN: Buffer Size

This optional parameter can be used to specify the size, in bytes, of a buffer used by ADACMP to assemble any segmented, uncompressed, physical records into a compressed logical record. This buffer is allocated only if HEADER=YES is also specified. Otherwise, the setting of MAXLOGRECLEN is ignored. The default value of MAXLOGRECLEN is 1048576 bytes (1 MB).

If the value specified by MAXLOGRECLEN is appended by the letter "K", it is multiplied by 1024. The minimum value is 32768 bytes.

Note: MAXLOGRECLEN pertains to the spanning of uncompressed input data and should not be confused with the SPAN parameter which pertains to the spanning of compressed records.

MINISN: Starting ISN

For automatic ISN assignment, MINISN defines the lowest ISN to be used. If MINISN is not specified, the default is 1. If USERISN is specified, MINISN cannot be specified.

MUPECOUNT: Specify Value Count Field Size

The `MUPECOUNT` parameter specifies the size of the value count field in the input record for the COMPRESS function . Its syntax is:

```
MUPECOUNT={1 | 2}
```

If "1" is specified, each value count field preceding the MU or PE values in the input data must be one byte with a value of no more than "191". If "2" is specified, each value count field preceding the MU or PE values in the input data must be two bytes. A value count may exceed 191 only if the `MUPEX` parameter is also specified. When `MUPEX` is specified, the maximum count is "65534".

If the `MUPEX` parameter has been set, the default for `MUPECOUNT` is "2"; if the `MUPEX` parameter has not been set, the default for `MUPECOUNT` is "1".

Note: This option is not compatible with releases prior to Adabas 8; therefore, backward conversion to prior versions is not possible once records with more than 191 PE group occurrences have been loaded. However, ADACMP data sets created by versions of Adabas prior to Adabas 8 will load successfully using the Version 8 ADALOD utility.

For information on how to identify MU and PE occurrences greater than 191 in the compressed record, read ***Identifying MU and PE Occurrences Greater Than 191 in Compressed Records***, elsewhere in this section.

MUPEX: Enable Extended Periodic Group Count

The `MUPEX` parameter indicates whether extended MU/PE limits (greater than 191) are allowed for the file. If this option is *not* specified, the maximum number of MU fields and the maximum number of PE fields that can be specified is 191. Otherwise, the maximum is 65534.

If you set the `MUPEX` parameter, consider setting the `SPAN` parameter as well to avoid compression errors if the compressed record size is exceeded when compressing the additional MU and PE fields.

Note: This option is not compatible with releases prior to Adabas 8; therefore, backward conversion to prior versions is not possible once records with more than 191 PE group occurrences have been loaded. However, ADACMP data sets created by versions of Adabas prior to Adabas 8 will load successfully using the Version 8 ADALOD utility.

For information on how to identify MU and PE occurrences greater than 191 in the compressed record, read *Identifying MU and PE Occurrences Greater Than 191 in Compressed Records*, elsewhere in this section.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

NUMREC: Number of Records to Be Processed

Specifies the number of input records to be processed. If this parameter is omitted, all input records contained on the input data set are processed.

we recommend using this parameter for the initial ADACMP execution if a large number of records are contained on the input data set. This avoids unneeded processing of all records when a field definition error or invalid input data results in a large number of rejected records. This parameter is also useful for creating small files for test purposes.

Setting NUMREC to zero (0) prevents the input data set from being opened.

PASSWORD: Password for FDT File

If the FDT parameter is specified and the file is password-protected, this parameter must be used to provide a valid password for that file.

SPAN: Enabling Spanned Records

The `SPAN` parameter allows the record to be spanned after compression if its compressed size exceeds the Data Storage block size of the device.

TZ: Time Zone

The TZ parameter can be used to specify the local time zone. As records are compressed and loaded into the file, the date-time data is converted from the specified time zone to UTC time (Coordinated Universal Time, also known as Greenwich Mean Time). Date-time field data is always stored in UTC time in an Adabas file. Valid time zone names are listed in the TZINFO member of the Adabas source library and are specified in single quotes. The following is an example of a valid TZ specification:

```
TZ='America/New_York'
```

Note: Time zone names are case-sensitive.

This example sets the time zone to Eastern US. When compressing input, local time is assumed to be Eastern US and will be converted to UTC time for storage on the database.

Adabas uses the time zone data taken from the **tz database**, which is also called the *zoneinfo* or *Olson* database. The specific list of time zone names that Adabas supports in any given release can be found in the TZINFO member of the Adabas source library (ADA*vrs*.SRCE). For more information about the TZINFO member of the time zone library, read *Supported Time Zones*, in the *Adabas DBA Tasks Manual*.

Note: Review important information about the **daylight savings time (DST)** parameter.

If a time zone uses daylight savings time, you must be sure to store the daylight savings indicator with your date-time data or there will be no way to distinguish date-time values in the hour before the time is switched back to standard time.

UACODE: User Encoding for Input Alphanumeric Fields

UACODE defines the user encoding of the sequential input of alphanumeric fields. If you specify UACODE, you *must* also specify FACODE.

UARC: Architecture for Input Uncompressed User Data

The UARC parameter specifies the architecture of the sequential input of the uncompressed user data. The "userdata-architecture-key" is an integer which is of the sum of the following numbers:

byte order	b=0	high-order byte first
	b=1	low-order byte first
encoding family	e=0	ASCII encoding family
	e=2	EBCDIC encoding family (default)
floating-point format	f=0	IBM370 floating-point format
	f=4	VAX floating-point format
	f=8	IEEE floating-point format

The default is ARC = b + e + f = 2; that is, high-order byte first; EBCDIC encoding family; and IBM370 floating-point format (b=0; e=2; f=0).

User data from an Intel386 PC provides the example: b=1; e=0; f=8; or ARC=9.

USERISN: User ISN Assignment

The ISN for each record is to be user-defined. If this parameter is omitted, the ISN for each record is assigned by Adabas.

If USERISN is specified, you must provide the ISN to be assigned to each record as a four-byte binary number immediately preceding each data record. If the MINISN parameter is specified, USERISN cannot be specified.

If USERISN is specified with HEADER=YES, the ISN immediately follows the ADAH header as part of the logical record.

The format for fixed or undefined length input records with user-defined ISNs is:

```
userisn/data
```

The format for variable-length input records with user-defined ISNs is

```
length/xx/userisn/data
```

where

length	is a two-byte binary physical record length (length of record data, plus 8 bytes).
xx	is a two-byte field containing binary zeros.
userisn	is a four-byte binary ISN to be assigned to the record.
data	is input record data.

ISNs may be assigned in any order, must be unique (for the file), and must not exceed the MAXISN setting specified for the file (see the **ADALOD utility** documentation).

ADACMP does not check for unique ISNs or for ISNs that exceed MAXISN. These checks are performed by the ADALOD utility.

UWCODE: User Encoding for Input Wide-Character Fields

UWCODE defines the user encoding of the sequential input of wide-character fields. If you specify UWCODE, you *must* also specify FWCODE.

For user input, all wide-character fields are encoded in the same code page. It is not possible to select different encodings for different fields in the same ADACMP run.

ADACMP COMPRESS Examples

Example 1:

```
ADACMP   COMPRESS
ADACMP   FNDEF='01,AA,7,A,DE,FI'                    Field AA
ADACMP   FNDEF='01,AB,15,A,DE,MU,NU'                Field AB
ADACMP   FNDEF='01,GA'                              Group GA
ADACMP   FNDEF='02,AC,15,A,NU'                      Field AC
ADACMP   FNDEF='02,AD,2,P,FI'                       Field AD
```

```
ADACMP   FNDEF='02,AE,5,P,NU'                        Field AE
ADACMP   FNDEF='02,AF,6,W'                           Field AF
ADACMP   FNDEF='01,DT,8,P,DT=E(DATETIME),TZ'         Field DT
ADACMP   COLDE='7,Y1=AF'                             Collation descriptor Y1
ADACMP   SUBDE='BB=AA(1,4)'                          Subdescriptor BB
ADACMP   SUPDE='CC=AA(1,4),AD(1,1)'                  Superdescriptor CC
ADACMP   HYPDE='1,DD,4,A,MU=AB,AC,AD'                Hyperdescriptor DD
ADACMP   PHONDE='EE(AA)'                             Phonetic descriptor EE
ADACMP   SUBFN='FF=AA(1,2)'                          Subfield FF
ADACMP   SUPFN='GG=AA(1,4),AD(1,1)'                  Superfield GG
ADACMP   TZ='Europe/Berlin',DST                      Time zone specification
```

The time zone in this example is set to "Europe/Berlin", with a day light savings indicator. The following fields are defined:

Field	Is defined as ...
AA	Level 1, seven bytes alphanumeric, descriptor, fixed storage option.
AB	Level 1, 15 bytes alphanumeric, descriptor, multiple value field, null value suppression.
GA	A group containing fields AC, AD, AE, and AF.
AC	Level 2, 15 bytes alphanumeric, null value suppression.
AD	Level 2, two bytes packed, fixed storage option.
AE	Level 2, five bytes packed, null value suppression.
AF	Level 2, six bytes wide-character format.
DT	Level 1, 8 bytes, packed, using a DATETIME edit mask and for which time zone conversion is requested. **Note:** The time zone for compression is specified as "Europe/Berlin" time and includes a daylight savings indicator.
Y1	A collation descriptor for AF and is assigned to collation descriptor user exit 7 (CDX07).
BB	A subdescriptor (positions 1-4 of field AA).
CC	A superdescriptor (positions 1-4 of field AA and position 1 of field AD).
DD	A hyperdescriptor consisting of fields AB, AC and AD. DD is assigned hyperdescriptor exit 1.
EE	A phonetic descriptor derived from field AA.
FF	A subfield (positions 1-2 of field AA).
GG	A superfield (positions 1-4 of AA and position 1 of AD).

Example 2: With TZ field using TZ='Europe/Berlin' and DST option

```
ADACMP  COMPRESS
ADACMP  FNDEF='01,T3,14,U,NU,TZ,DT=E(DATETIME)'
ADACMP  TZ='Europe/Berlin',DST
```

As example we look at a local datetime value around the transition time from daylight saving time to standard time on October 29, 2017, i.e. when the local clock is wound back by 1 hour at 01:00 UTC:

```
2017-03-26 01:00:00 UTC = 2017-03-26 03:00:00 CEST  +02:00:00 dst=+01:00:00
2017-10-29 01:00:00 UTC = 2017-10-29 02:00:00 CET   +01:00:00 dst=+00:00:00
```

We take the local time value `2017-10-29 02:59:00` for timezone Berlin. The value is ambiguous without specifiying DST and adding the DST indicator to the datetime value makes it clear. For the Berlin time `2017-10-29 02:59:00` possible DST values are:

- `+01, DST value = 3600 or X'0E10'` is UTC time 2`017-10-29 00:59:00`. This is one minute before the local clock is set 1 hour back.

  Input `'20171029025900',X'0E10'` is compressed to the 8 bytes packed value `02017102 9005900F`.
- `+00, DST value = 0` is UTC time `2017-10-29 01:59:00`. This is 59 minutes after the local clock was set 1 hour back.

  Input `'20171029025900',X'0000'` is compressed to the 8 bytes packed value `02017102 9015900F`.
- When the daylight savings indicator is not given in the input, there is ambiguity and compression will select the lower absolute time value (UTC):

  ```
  ADACMP  TZ='Europe/Berlin'       * DST not specified
  ```

  Input `'20171029025900'` is compressed to the 8 bytes packed value `02017102 9005900F`.

Example 3:

```
ADACMP   COMPRESS
ADACMP   FORMAT='AG,6,U,AF,4X,AA,'    input record format
ADACMP   FORMAT='AB,AC'               continuation of FORMAT statement
ADACMP   FNDEF='01,AA,10,A,NU'        field definitions
ADACMP   FNDEF='01,AB,7,U,NU'
ADACMP   FNDEF='01,AF,5,P,NU'
ADACMP   FNDEF='01,AG,12,P,NU,DE'
ADACMP   FNDEF='01,AC,3,A,NU,DE'
```

The input record format is provided explicitly using the FORMAT parameter. ADACMP uses this format as the basis for processing fields from the input record. The FDT for the file corresponds to the structure specified in the FNDEF statements.

Example 4:

```
ADACMP COMPRESS
ADACMP FORMAT='AG,AF,4X,AA,AB,AC'     input record format
ADACMP FDT=8                          FDT same as file 8
```

The input record format is provided explicitly using the FORMAT parameter. The FDT to be used is the same as that currently defined for Adabas file 8.

Example 5:

```
ADACMP   COMPRESS NUMREC=2000,USERISN
ADACMP   FNDEF='01,AA,7,A,DE,FI'            Field AA
ADACMP   FNDEF='01,AB,15,A,DE,MU,NU'        Field AB
```

The number of input records to be processed is limited to 2000. The ISN for each record is to be provided by the user.

38 DECOMPRESS: Decompress an Adabas File

- Optional Parameters and Subparameters 240
- Decompressing Multiclient Files 226
- ADACMP DECOMPRESS Examples 227

The DECOMPRESS function decompresses data either

- from output unloaded by the ADAULD UNLOAD utility function; or
- directly from a single compressed Adabas file when the file number is specified with the INFILE parameter.

When decompressing data directly from the INFILE file, DECOMPRESS first performs an ADAULD UNLOAD/MODE=SHORT function. This can save time over separate ADAULD and ADACMP DECOMPRESS operations.

```
ADACMP  DECOMPRESS  [CODE =  cipher-code ]
                    [FORMAT =  output-record-format-definition ]
                    [HEADER = { YES | NO } ]
                    [INFILE = file-number ]
                       [ETID = owner-id ]
                       [LPB = prefetch-buffer-size ]
                       [PASSWORD = ' password ' ]
                       [SORTSEQ = { descriptor [,NU] | ISN } ]
                       [UTYPE = { EXF | EXU } ]
                    [ISN]
                    [LOBVALUES = { YES | NO } ]
                    [MAXLOGRECLEN = buffer-size ]
                    [NOUSERABEND]
                    [NUMREC = number-of-records ]
                    [TRUNCATE]
                    [TZ = ' timezone-name ' [,DST] ]
                    [UACODE = userdata-alpha-key ]
                    [UARC = {architecture-key |  2 } ]
                    [UWCODE = userdata-wide-key ]
```

This chapter describes the syntax and parameters of the ADACMP DECOMPRESS function.

Optional Parameters and Subparameters

CODE: Cipher Code

If the file to be decompressed is ciphered, the cipher code that was used when the file was compressed must be specified with this parameter. See the *Adabas Security* documentation for additional information on the use of ciphering.

DST: Daylight Savings Indicator

The DST parameter can be specified to indicate that date-time data includes a daylight savings time indicator. If a time zone uses daylight savings time, you must be sure to store and retrieve the daylight savings indicator with your date-time data or there will be no way to distinguish date-time values in the hour before the time is switched back to standard time. The two-byte daylight savings indicator directly follows the date-time value in uncompressed input and specifies the binary integer value of the daylight saving time offset from standard time in seconds.

The DST parameter requires that the **TZ parameter** be set in the same ADACMP run. However, it should *not* be specified when the FORMAT parameter is specified in the run.

You must specify a DST parameter for files containing date-time fields defined with option TZ, when the time zone includes a daylight savings time indicator. If the DST parameter is not specified, date-time data is stored without a daylight savings time indicator. The default is store the date-time data without a daylight savings time indicator.

ETID: Multiclient File Owner ID

ETID specifies an owner ID for a multiclient file specified by INFILE. ADACMP DECOMPRESS selectively decompresses only those records in the multiclient file assigned to the owner ID specified by ETID. The ETID value must be the same as that assigned to the records when they were loaded into the multiclient file.

FORMAT: Output Record Format Definition

FORMAT allows decompression to a format other than that specified by the FDT. It can be used to change the FDT of an existing file and, in particular, the structure of a periodic (PE) group.

The FORMAT parameter syntax is the same as the format buffer syntax used for read commands except that text cannot be inserted (text is not compressible/decompressible); see the *Adabas Command Reference Guide* documentation for more information.



Note: The FORMAT parameter does not check whether all related data fields have been processed during decompression.

For example, if a multiple-value (MU) field defined as `01,AA,8,A,MU` has five occurrences, and the ADACMP DECOMPRESS FORMAT parameter specifies `AA1-4`, then only the first four AA field values are decompressed; no indication is given regarding the fifth field value. This also applies to PE field occurrences and length overrides.

LB (large object) fields may be specified in the definition supplied in the FORMAT parameter if the **LOBVALUES** parameter is set to NO. This functionality is most useful for changing the FDT of a base file in a LOB (large object) file group. In this case, the LB field values stored in the LOB file are not modified.

The following restrictions apply to LB fields specified in a FORMAT parameter:

- An LB field cannot be specified more than once.
- An LB field cannot be specified with a length or asterisk (*) length notation.
- LOB segment notation is not allowed.
- Length indicator (L) notations are not allowed for LB fields



Caution: Care must be taken when specifying LB fields in a FORMAT parameter if the LOBVALUES parameter is set to NO. Failure to specify all LB fields could lead to LB values existing in the LOB file without a corresponding LB value reference in the base file.

HEADER

This optional parameter indicates whether or not the ADACMP decompression logic should produce the ADACMP segmented record headers (ADAH and ADAC) as part of the decompressed output. Valid values are YES or NO; the default is NO.

HEADER=NO is the format accepted and produced by ADACMP in Adabas versions prior to Adabas 8. When it is specified, the decompressed output records produced by ADACMP will contain only the data for the fields of the file being processed. Each data record must fit into one physical record of the sequential input data set (less than 32 KB). If the data exceeds this size, the records in error are written to the DDFEHL error data set.

HEADER=YES can only be specified if you are running Adabas 8 or later Adabas versions. If HEADER=YES is specified, each output decompressed record produced by ADACMP begins with either an ADAH or ADAC header, relating the physical record with a logical record. Each logical record can be larger than 32 KB. The header in each physical record defines the position of the data following it within the logical record. For complete information about segmented records in ADACMP, read *Segmented Record Considerations*, elsewhere in this guide. DSECTs for the ADAH and ADAC headers can be found in members ADAH and ADAC of the distributed Adabas SRCE data set.

INFILE: Number of File to Be Decompressed

The INFILE parameter allows you to decompress a file without first unloading it with the ADAULD utility. If the INFILE parameter is not specified, the input is read from a sequential (DDEBAND) file. With the ETID parameter, INFILE permits selectively decompressing records from a multiclient file. When decompressing multiclient files, refer to the section *Decompressing Multiclient Files*, elsewhere in this section.

ISN: Include ISN in Decompressed Output

The ISN of each record is to be included with each decompressed record output. If this parameter is omitted, the ISN will not be included with each record.

If ISN is specified with HEADER=YES, the ISN immediately follows the ADAH header as part of the logical record. The DSECT for the ADAH header can be found in member ADAH of the distributed Adabas SRCE data set.

LPB: Prefetch Buffer Size

LPB specifies the size, in bytes, of the internal prefetch buffer for the ADACMP DECOMPRESS INFILE function. The maximum value is 32760 bytes. The default is calculated by Adabas, depending on the ADARUN LU value in effect for the nucleus.

LOBVALUES: LB Field Size Indicator

The LOBVALUES parameter should only be specified for files containing large object (LB) fields.

This optional parameter indicates whether long LB field values (larger than 253 bytes) or short LB field values (up to 253 bytes) are expected in the ADACMP DECOMPRESS output data. Valid values for this parameter are "YES" and "NO"; the default is "NO".

If "NO" is specified for this parameter, the uncompressed output data may contain only LB fields up to 253 bytes long and references to LB field values stored in a *LOB file*. In this case, you cannot specify an LB field in the ADACMP DECOMPRESS FORMAT parameter. During processing, ADACMP DECOMPRESS reads only the base file records as input; if the base file contains references to LB field values in a LOB file, ADACMP DECOMPRESS does not read them, but only reproduces the references in the output.

If "YES" is specified for this parameter, the uncompressed output data will contain the LB field values present for the record. In this case, the INFILE parameter must also be specified to identify the file number of the *base file* of a *LOB file group* whose data is to be decompressed. During processing, as ADACMP DECOMPRESS reads and decompresses the records from the base file, it reads any referenced LB field values from the *LOB file*.



Note: An ADACMP DECOMPRESS function with LOBVALUES=NO followed by an ADACMP COMPRESS function with LOBVALUES=NO can be used to modify the FDT of the *base file* in the *LOB file group*.

MAXLOGRECLEN: Buffer Size

This optional parameter can be used to specify the size, in bytes, of a buffer used by ADACMP to assemble logical records that span one or more physical records of uncompressed output data. This buffer is allocated only if HEADER=YES is also specified. Otherwise, the setting of MAXLOGRECLEN is ignored. The default value of MAXLOGRECLEN is 1048576 bytes (1 MB).

If the value specified by MAXLOGRECLEN is appended by the letter "K", it is multiplied by 1024. The minimum value is 32768 bytes.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

NUMREC: Number of Records to Be Processed

NUMREC specifies the number of input records to be processed. If this parameter is omitted, all input records contained on the input data set are processed.

Use of NUMREC is recommended for the initial ADACMP execution if a large number of records are contained on the input data set. This avoids unneeded processing of all records when a field definition error or invalid input data causes a large number of rejected records. NUMREC is also useful for creating small files for test purposes.

PASSWORD: Password for INFILE

The PASSWORD parameter must specify the correct password if the file is to be decompressed directly from a password-protected Adabas file.

SORTSEQ: Processing Sequence for INFILE File

SORTSEQ determines the sequence in which the file is processed. If this parameter is omitted, the records are processed in physical sequence. SORTSEQ can be specified only when INFILE is also specified.

If a descriptor is specified, the file is processed in the logical sequence of the descriptor values. *Do not* use a hyperdescriptor, a phonetic descriptor, a multiple-value descriptor field, or a descriptor contained in a periodic group.

If the descriptor name refers to a field defined with the null suppression (NU) option, you must specify ",NU" after the descriptor name. In this case, records of the descriptor that contain null values are not decompressed. If NU is not specified in this case (the default), ADACMP rejects NU descriptors.



Note: Even when the descriptor field is not null-suppressed, the record is *not* represented in the inverted list if the descriptor field or a field following it has never been initialized (held a value). Therefore, the record will be dropped when the utility is executed.

If ISN is specified, the file is processed in ascending ISN sequence. For the Adabas checkpoint or security file, only SORTSEQ=ISN is allowed.

TRUNCATE: Truncate Excess Alphanumeric Characters

The TRUNCATE parameter enables truncation of compressed alphanumeric data during decompression. When TRUNCATE is specified and ADACMP DECOMPRESS operation finds

an alphanumeric field containing more characters than the FDT description allows for the field, the extra characters are truncated. If TRUNCATE is not specified, alphanumeric records with extra characters are written to the DDFEHL data set. Non-alphanumeric fields cannot be truncated.

TZ: Time Zone

The TZ parameter can be used to specify the local time zone. As records are decompressed and read from the file, the date-time data is converted from UTC time (Coordinated Universal Time, also known as Greenwich Mean Time) to the corresponding time for the specified time zone. Date-time field data is always stored in UTC time in an Adabas file. Valid time zone names are listed in the TZINFO member of the Adabas source library and are specified in single quotes. The following is an example of a valid TZ specification:

```
TZ='America/New_York'
```



Note: Time zone names are case-sensitive.

This example sets the time zone to Eastern US. When decompressing input, local time is assumed to be Eastern US. When retrieving the stored UTC time data, it is converted to Eastern US time.

Adabas uses the time zone data taken from the **tz database**, which is also called the *zoneinfo* or *Olson* database. The specific list of time zone names that Adabas supports in any given release can be found in the TZINFO member of the Adabas source library (ADA*vrs*.SRCE). For more information about the TZINFO member of the time zone library, read *Supported Time Zones*, in the *Adabas DBA Tasks Manual*.



Note: Review important information about the **daylight savings time (DST)** parameter. If a time zone uses daylight savings time, you must be sure to retrieve the daylight savings indicator with your date-time data or there will be no way to distinguish date-time values in the hour before the time is switched back to standard time.

UACODE: Encoding Protocol for Output Alphanumeric Fields

UACODE defines the encoding of the sequential output of alphanumeric fields. This parameter allows you to override the user encoding for alphanumeric fields passed in the header of the compressed sequential input.

UARC: Architecture for Output Uncompressed User Data

The UARC parameter specifies the architecture of the sequential output of the uncompressed user data. This parameter allows you to override the user encoding passed in the header of the compressed sequential input.

The 'userdata-architecture-key' is an integer which is the sum of the following numbers:

byte order	b=0	high-order byte first
	b=1	low-order byte first
encoding family	e=0	ASCII encoding family
	e=2	EBCDIC encoding family (default)
floating-point format	f=0	IBM370 floating-point format
	f=4	VAX floating-point format
	f=8	IEEE floating-point format

The default is ARC = b + e + f = 2; that is, high-order byte first; EBCDIC encoding family; and IBM370 floating-point format (b=0; e=2; f=0).

User data from an Intel386 PC provides the example: b=1; e=0; f=8; or ARC=9.

UTYPE: User Type

The user type to be in effect when unloading the file specified by INFILE. Allowed values are

EXF	no access/update allowed for other users of the file.
EXU	access only is allowed for other users of the file. EXU is the default.

UWCODE: Encoding Protocol for Output Wide-Character Fields

UWCODE defines the encoding of the sequential output of wide-character fields. This parameter allows you to override the user encoding for wide-character fields passed in the header of the compressed sequential input.

Decompressing Multiclient Files

ADACMP decompresses Adabas data to a sequential user file. The DECOMPRESS function can decompress records selectively if the INFILE parameter specifies a multiclient file and a valid ETID value is specified.

The DECOMPRESS function skips the owner ID, if present. The output of a DECOMPRESS operation on a multiclient file contains neither owner ID nor any ETID information.

If the INFILE parameter specifies a multiclient file for the DECOMPRESS function, you can use the ETID parameter to limit decompression to records for a specific user only. ADACMP then reads and decompresses records only for the specified user. If the ETID parameter is not specified when decompressing a multiclient file, all records in the file are decompressed.

Example:

Only records owned by USER1 from file 20 are decompressed to a sequential output file:

```
ADACMP  DECOMPRESS  INFILE=20,ETID=USER1
```

ADACMP DECOMPRESS Examples

Example 1:

The DECOMPRESS function is to be executed. The input data set to be used is the output of a previous execution of the ADAULD utility:

```
ADACMP DECOMPRESS
```

Example 2:

Adabas file 23 is to be decompressed. The ISN of each record is to be included in the decompressed output:

```
ADACMP DECOMPRESS INFILE=23,ISN
```

39 Field Definition Statements

- FNDEF: Field and Group Definition 231
- FNDEF: Periodic Group Definition 250
- COLDE: Collation Descriptor Definition 253
- HYPDE: Hyperdescriptor Definition 256
- PHONDE: Phonetic Descriptor 259
- SUBDE: Subdescriptor Definition 261
- SUBFN: Subfield Definition 263
- SUPDE: Superdescriptor Definition 264
- SUPFN: Superfield Definition 273

The field definitions provided as input to ADACMP are used to:

- provide the length and format of each field contained in the input record. This enables ADACMP to determine the correct field length and format during editing and compression.
- create the Field Definition Table (FDT) for the file. This table is used by Adabas during the execution of Adabas commands to determine the logical structure and characteristics of any given field (or group) in the file.

The following syntax must be followed when entering field definitions. A minimum of one and a maximum of 3214 definitions may be specified.

Statement Type	Syntax
Field and Group	**FNDEF = '***level*, *name* **[** , *length*, *format* **] [, MU [(***occurrences* **)]] [** , *option* **] ... '**
Periodic Group	**FNDEF = '***level*, *name* **[** , **PE [(** *occurrences* **)]]'**
Collation descriptor	**COLDE = '***number*, *name* **[** , **UQ [, XI]] =** *parent-field* **'**
Hyperdescriptor	**HYPDE = '***number*, *name*, *length*, *format* **[{** , *option* **} ...] = {** *parent-field* **} , ...'**
Phonetic descriptor	**PHONDE = '** *name* **(***field***)'**
Subdescriptor	**SUBDE = '***name* **[, UQ [, XI]] =** *parent-field* **(***begin*, *end***)'**
Subfield	**SUBFN = '** *name* **=** *parent-field* **(***begin*, *end***)'**

Statement Type	Syntax
Superdescriptor	**SUPDE = '***name* **[, UQ [, XI]] = {***parent-field* **(***begin*, *end* **) } , ...'**
Superfield	**SUPFN = '***name* **=** *parent-field* **(***begin* , *end* **)[,** *parent-field* **(** *begin* , *end* **)]...'**

User comments may be entered to the right of each definition. At least one blank must be present between a definition and any user comments.

Each of these definition types is described in this chapter.

FNDEF: Field and Group Definition

The FNDEF parameter can be used to specify an Adabas field or group definition. The syntax used in constructing field and group definition entries is:

FNDEF = '*level*, *name* **[** , *length*, *format* **] [, MU [(***occurrences* **)]] [** , *option* **] ... '**

Level number and name are required. Any number of spaces may be inserted between definition entries.

Each FNDEF parameter is described in this section.

- level
- name
- length
- format
- occurrences
- field options

The MU parameter is documented in ***field options***

level

The level number is a one- or two-digit number in the range 01-07 (the leading zero is optional) used in conjunction with field grouping. Fields assigned a level number of 02 or greater are considered to be a part of the immediately preceding group which has been assigned a lower level number.

The definition of a group enables reference to a series of fields (may also be only 1 field) by using the group name. This provides a convenient and efficient method of referencing a series of consecutive fields.

Level numbers 01-06 may be used to define a group. A group may consist of other groups. When assigning the level numbers for nested groups, no level numbers may be skipped.

In the following example, fields A1 and A2 are in group GA. Field B1 and group GC (consisting of fields C1 and C2) are in group GB:

FNDEF='01,GA'	group	
FNDEF='02,A1,...'		elementary or multiple-value field
FNDEF='02,A2,...'		elementary or multiple-value field
FNDEF='01,GB'	group	
FNDEF='02,B1,...'		elementary or multiple-value field
FNDEF='02,GC'	group (nested)	
FNDEF='03,C1,...'		elementary or multiple-value field
FNDEF='03,C2,...'		elementary or multiple-value field

name

The name to be assigned to the field (or group).

Names must be unique within a file. Names are case-sensitive and must be two characters long: the first character must be alphabetic; the second character can be either alphabetic or numeric. No special characters are permitted.



Note: Lowercase fields will not display correctly (they will be converted to uppercase) if you use the ADARUN parameter settings MSGCONSL=UPPER, MSGDRUCK=UPPER, or MSGPRINT=UPPER.

The values E0-E9 are reserved as edit masks and may not be used.

Valid Names	Invalid Names
AA	A (not two characters)
s3	E3 (edit mask)
S3	F* (special character)
wm	6M (first character not alphabetic)
wM	
Wm	

length

The length of the field (expressed in bytes). The length value is used to

- indicate to ADACMP the length of the field as it appears in each input record; and
- define the standard (default) length to be used by Adabas during command processing.

The standard length specified is entered in the FDT and is used when the field is read/updated unless the user specifies a length override.

The maximum field lengths that may be specified depend on the "format" value:

Format	Maximum Length
Alphanumeric (A)	253 bytes
Binary (B)	126 bytes
Fixed Point (F)	8 bytes (always exactly 2, 4, or 8 bytes)
Floating Point (G)	8 bytes (always exactly 4 or 8 bytes)
Packed Decimal (P)	15 bytes
Unpacked Decimal (U)	29 bytes
Wide-character (W)	253 bytes*

** Depending on the FWCODE attribute value, the maximum byte length of the W field may be less than 253. For example, if the default value of FWCODE is used (that is, Unicode), the maximum length is 252 (2 bytes per character).*

Standard length may not be specified with a group name.

Standard length does not limit the size of any given field value unless the FI option is used - see ***FI: Fixed Storage*** . A read or update command may override the standard field length, up to the maximum length permitted for that format.

If standard length is zero for a field, the field is assumed to be a variable-length field. Variable-length fields have no standard (default) length. A length override for fixed-point (F) fields can specify a length of two or four bytes only; for floating-point (G) fields, the override can specify four or eight bytes only.

If a variable-length field is referenced without a length override during an Adabas command, the value in the field will be returned preceded by a one-byte binary length field (including the length byte itself). This length value must be specified when the field is updated, and also in the input records that are to be processed by ADACMP. If the field is defined with the long alpha (LA) option, the value is preceded by a two-byte binary length field (including the two length bytes).

format

The standard format of the field (expressed as a one-character code):

A	Alphanumeric (left-justified)
B	Binary (right-justified, unsigned/positive)
F	Fixed point (right-justified, signed, two's complement notation)
G	Floating point (normalized form, signed)
P	Packed decimal (right-justified, signed)
U	Unpacked decimal (right-justified, signed)
W	Wide character (left-justified)

The standard format is used to

- indicate to ADACMP the format of the field as it appears in each input record; and
- define the standard (default) format to be used by Adabas during command processing. The standard format specified is entered in the FDT and is used when the field is read/updated unless the user specifies a format override.

Standard format must be specified for a field. It may not be specified with a group name. When the group is read (written), the fields within the group are always returned (must be provided) according to the standard format of each individual field. The format specified determines the type of compression to be performed on the field.

A fixed-point field is either two, four, or eight bytes long. A positive value is in normal form, and a negative value in two's complement form.

A field defined with floating-point format may be either four bytes (single precision) or eight bytes (double precision) long. Conversion of a value of a field defined as floating point to another format is supported.

If a binary field is to be defined as a descriptor, and the field may contain both positive and negative numbers, "F" format should be used instead of "B" format because "B" format assumes that all values are unsigned (positive).

Like an alphanumeric field, a wide-character field may be a standard length in bytes defined in the FDT, or variable length. Any non-variable format override for a wide-character field must be

compatible with the user encoding; for example, a user encoding in Unicode requires an *even* length. Format conversion from numbers (U, P, B, F, G) to wide-character format is not allowed.

occurrences

The number of occurrences of MU fields that will occur in a record if the MUPEX option is specified. This is an optional parameter.

field options

Options are specified by the two-character codes. More than one code may be specified (as applicable for the field). They may be specified in any order, separated by a comma.

The available options for field and group definitions are listed in the following table. For more information about a specific option, click on its name in the table.



Note: The PE option is another Adabas field option. However, it is available only if you are specifying a periodic group. For more information, read ***FNDEF: Periodic Group Definition***, elsewhere in this section.

Field Option	Description
CR	The system field value should only be set when the record is inserted into the Adabas file (it will not be modified when an update operation occurs). This option can only be specified for a field that is defined with the SY field option (system fields) and cannot be specified for a field with the MU field option (multiple value fields).
DE	The field is to be a descriptor (key).
DT	A date-time edit mask is specified for the binary, fixed point, packed decimal, or unpacked decimal field.
FI	The field is to have a fixed storage length; values are stored without an internal length byte, are not compressed, and cannot be longer than the defined field length.
LA	This A or W-format variable-length field may contain a value up to 16,381 bytes long.
LB	An alphanumeric field may contain up to 2,147,483,643 (about 2 GB) of data.
MU	The field may contain up to about 65,534 values in a single record.
NB	Trailing blanks should not be removed (compressed) from the LA or LB fields.
NC	Adabas includes two data definition options, **NC** and **NN**, to provide SQL-compatible null representation for Software GmbH's mainframe Adabas SQL Gateway(ACE) and other Structured Query Language (SQL) database query languages. For more information about these options, read ***Representing SQL Null Value***, elsewhere in this section. With the **NC** option, the field may contain a null value that satisfies the SQL interpretation of a field having no value; that is, the field's value is not defined (not counted). With the **NN** option, the field defined with the NC option must always have a value defined; it cannot contain an SQL null (not null).
NN	

Field Option	Description
NU	Null values occurring in the field are to be suppressed.
NV	This A or W-format field is to be processed in the record buffer without being converted.
SY	The field is a *system field*. This field option identifies the type of system field.
TZ	The date-time field value is presented in the user's local time and stored in UTC time, allowing for differences in time zones.
UQ	The field is to be a unique descriptor; that is, for each record in the file, the descriptor must have a different value.
XI	For this field, the index (occurrence) number is excluded from the UQ option set for a PE.

CR: Insert-Only System Field

Use the CR option to indicate that the system field value should only be set when a record is inserted and not when it is updated. The CR field option can only be specified for fields defined with the **SY field option**, but *cannot* be specified for an **MU field**.

For complete information about system fields, read *System Fields*, in *Adabas Concepts Manual*.

DE: Descriptor Field Option

DE indicates that the field is to be a descriptor (key). Entries will be made in the Associator inverted list for the field, enabling the field to be used in a search expression, as a sort key in a FIND command, to control logical sequential reading, or as the basis for file coupling.

The descriptor option should be used judiciously, particularly if the file is large and the field that is being considered as a descriptor is updated frequently.

Although the definition of a descriptor field is independent of the record structure, note that if a descriptor field is not ordered first in a record and logically falls past the end of the physical record, the inverted list entry for that record is not generated for performance reasons. To generate the inverted list entry in this case, it is necessary to unload short, decompress, and reload the file; or use an application program to reorder the field first for each record of the file.

DT: Date-Time Edit Mask Field Option

DT assigns a date-time edit mask to a binary, fixed point, packed decimal, or unpacked decimal field. This option cannot be specified for fields of other formats.

The syntax of the DT option is:

```
DT=E(edit-mask-name)
```

Valid values for edit-mask-name substitutions are described in the following table. It also shows the required minimum field lengths for the different formats of fields that can specify the DT option;

the length of the field must be large enough to store the date-time values. Detailed discussions of each edit mask is provided in *Date-Time Edit Mask Reference*, in the *Adabas DBA Tasks Manual*.



Note: In the table, "YYYY" represents the 4-digit year (1-9999), "MM" represents the 2-digit month (1-12), "DD" represents the 2-digit day of the month (1-31), "HH" represents the 2-digit hour (0-23), "II" represents the 2-digit minute within the hour (0-59), "SS" represents the 2-digit second within the minute (0-59), and "XXXXXX" represents the 6-digit microsecond within the second.

edit-mask-name	Description	Minimum Field Length for Field Format			
		B	F	P	U
DATE	The date field is in the format `Z'YYYYMMDD'`.	4	4	5	8
TIME	The time field is in the format `Z'HHIISS'`.	3	4	4	6
DATETIME	The date and time field is in the format `Z'YYYYMMDDHHIISS'`	6	8	8	14
TIMESTAMP	The date and time field is in the format `Z'YYYYMMDDHHIISSXXXXXX'`, with microsecond precision	--	--	11	20
NATTIME	The time field is in Natural T format (tenths of seconds since year zero)	6	8	7	13
NATDATE	The date field is in Natural D format (days since year zero)	3	4	4	7
UNIXTIME	The time field is in UNIX time_t type format (seconds since January 1, 1970)	4	4	6	10
XTIMESTAMP	The date and time field is in UNIX timestamp format, with microsecond precision, since January 1, 1970 (`UNIXTIME * 10**6 + microseconds`).	8	8	10	18

The following table contains some examples.

Example	The field contains...
`1,SD,8,U,DT=E(DATE)`	Numeric data in the form `Z'YYYYMMDD'`.
`1,TI,6,U,DT=E(TIME)`	Numeric time in the form `Z'HHIISSD'`
`1,DT,14,U,DT=E(DATETIME)`	A value composed of DATE and TIME
`1,TS,20,U,DT=E(TIMESTAMP)`	A value composed of DATETIME plus microseconds
`1,TT,7,P,DT=E(NATTIME)`	Natural T-format data (tenths of seconds since the year zero)
`1,DD,4,P,DT=E(NATDATE)`	Natural D-format data (days since the year zero)
`1,UU,4,F,DT=E(UNIXTIME)`	UNIX time_t-type data (seconds since January 1, 1970)
`1,XS,8,F,DT=E(XTIMESTAMP)`	UNIX time data (microseconds since January 1, 1970)
`1,DZ,14,U,TZ,DT=E(DATETIME)`	Data and time data stored internally in UTC format; timezone of the user session (TZ) determines user local time.

FI: Fixed Storage Field Option

FI indicates that the field is to have a fixed storage length. Values in the field are stored without an internal length byte, are not compressed, and cannot be longer than the defined field length.

The FI option is recommended for fields with a length of one or two bytes that have a low probability of containing a null value (personnel number, gender, etc.) and for fields containing values that cannot be compressed.

The FI option is not recommended for multiple-value fields, or for fields within a periodic group. Any null values for such fields are not suppressed (or compressed), which can waste disk storage space and increase processing time.

The FI option *cannot* be specified for

- U-format fields;
- NC, NN, or NU option fields;
- variable-length fields defined with a length of zero (0) in the FNDEF statement;
- a descriptor within a periodic (PE) group.

A field defined with the FI option *cannot* be updated with a value that exceeds the standard length of the field.

Example of FI usage:

	Definition	User Data	Internal Representation
Without FI Option	FNDEF='01,AA,3,P'	33104C 00003C	0433104F (4 bytes) 023F (2 bytes)
With FI Option	FNDEF='01,AA,3,P,FI'	33104C 00003C	33104F (3 bytes) 00003F (3 bytes)

LA: Long Alpha Field Option

The LA (long alphanumeric) option can be specified for variable-length alphanumeric and wide format fields; i.e., A or W format fields having a length of zero in the field definition (FNDEF). With the LA option, such a field can contain a value up to 16,381 bytes long.

An alpha or wide field with the LA option is compressed in the same way as an alpha or wide field without the option. The maximum length that a field with LA option can actually have is restricted by the block size where the compressed record is stored.

When a field with LA option is updated or read, its value is either specified or returned in the record buffer, preceded by a two-byte length value that is inclusive (field length, plus two).

A field with LA option:

- can also have the NU, NC/NN, NV, or MU option;

- can be a member of a PE group;
- cannot have the FI option;
- cannot be a descriptor field;
- cannot be a parent of a hyperdescriptor or phonetic descriptor; and
- cannot be specified in the search buffer, or response code 61 (ADARSP061) occurs.

For more information, read *Specifying Field Lengths of LA (Long Alpha) Fields in Format Buffers* and *Specifying Field Lengths of LA (Long Alpha) Fields in Record Buffers* in the *Adabas Command Reference Guide* .

Example of LA usage:

	Definition	User Data	Internal Representation
Without LA Option	FNDEF='01,BA,0,A'	X'06',C'HELLO' --	X'06C8C5D3D3D6' (1-byte length) --
With LA Option	FNDEF='01,BA,0,A,LA'	X'0007',C'HELLO' X'07D2',C' ... (2000 data bytes) ...'	X'06C8C5D3D3D6' (1-byte length) X'87D2 ... (2000 data bytes) ... '

LB: Large Object Field Option

The large object (LB) option can be specified for some fields to identify them as *large object* fields. LB fields can contain up to 2,147,483,643 bytes (about 2 GB) of data.

The format of an LB field must be "A" (alphanumeric) and its default field length must currently be defined as zero.

LB fields *cannot* be:

- Descriptors or parents of a special (phonetic, sub-, super-, or hyper-) descriptor.
- Defined with the FI or LA options.

  To assist you in determining whether to define a field as an LA or an LB field, read *Comparing LA and LB Fields*, in the *Adabas Concepts Manual*.
- Specified in a search buffer or in format selection criteria in a format buffer.

LB fields may be:

- Defined with any of the following options: MU, NB, NC, NN, NU, or NV
- Part of a simple group or a PE group.

The presence of the NB (no blank compression) field option in the LB field definition indicates whether on not Adabas removes trailing blanks in LB fields containing characters.

LB fields containing both binary and character data are supported. An LB field defined with both the NV and NB options can store binary large object data, as Adabas will not modify binary LB fields in any way. The identical LB binary byte string that was stored is what is retrieved when the LB field is read. In addition, because LB fields containing binary values are defined with the NV and the NB options, Adabas will not convert LB field binary values according to some character code page nor will it cut off trailing blanks in LB fields containing binary values.



Note: LB fields containing binary values are not defined using format B, because format B can imply byte swapping in some environments with different byte orders. Byte swapping does not apply to binary LB fields.

The following table provides some valid example of FDT definitions for LB fields:

FDT Specification	Description
`1,L1,0,A,LB,NU`	Field L1 is a null-suppressed, character, large object field
`1,L2,0,A,LB,NV,NB,NU,MU`	Field L2 is a null-suppressed, multiple-value, binary, large object field.

Commands dealing with LB fields must always be directed to the *base file* of a *LOB file group*. User commands against *LOB files* are rejected.

For information on getting started using LB fields, read *Large Object (LB) Files and Fields*, in *Adabas DBA Tasks Manual*.

MU: Multiple-Value Field Option

The multiple-value field option, indicating that the field may contain more than one value in a single record. If the **MUPEX** parameter is specified and the **MUPECOUNT** parameter is set to "2", the field may contain up to 65,534 values in a single record. If these parameters are not set, the field may contain up to 191 values in a single record. At least one value (even if null) must be present in each record input to ADACMP.

The values are stored according to the other options specified for the field. The first value is preceded by a count field that indicates the number of values currently present for the field. The number of values that are stored is equal to the number of values provided in the ADACMP input record, plus any values added during later updating of the field, less any values suppressed (this applies only if the field is defined with the NU option).

If the number of values contained in each record input to ADACMP is constant, the number can be specified in the MU definition statement in the form MU(*n*), where *n* is the number of values present in each input record. In the following example, three values of the multiple-value field AA are present in each input record:

```
FNDEF='01,AA,5,A,MU(3)' ↩
```

Specifying a value of zero MU(0) indicates that no values are present for the multiple-value field in the input record.

If the number of values is not constant for all input records, a one- or two-byte binary count field (depending on the **MUPECOUNT** parameter setting) must precede the first value in each input record to indicate the number of values present in that record (see also the section ***Input Data Requirements***).

If the FDT is provided (see the **FDT parameter** description), the field count must be contained as a one- or two-byte binary value in each input record (depending on the **MUPECOUNT** parameter setting).

If the input records were created using the DECOMPRESS function, all required count fields are already contained in the input record. In this case, the count must not be specified in the field definition statements.

All values provided during input or updating will be compressed (unless the **FI option** has also been specified). Care should be taken when using the FI and MU options together since a large amount of disk storage may be wasted if a large number of compressible values are present.

If the NU option is specified with the MU option, null values are both logically and physically suppressed. The positional relationship of all values (including null values) is usually maintained in MU occurrences, unless the occurrences are defined with the NU option. If a large number of null values are present in an MU field group, the NU option can reduce the disk storage requirements for the field but should not be used if the relative positions of the values must be maintained.

The NC (or NC/NN) option *cannot* be specified for an MU field.

For information on how to identify MU and PE occurrences greater than 191 in the compressed record, read ***Identifying MU and PE Occurrences Greater Than 191 in Compressed Records***, elsewhere in this section.

Example of MU usage with NU:

```
FNDEF='01,AA,5,A,MU,NU'
```

The original content where "L" is the length of the "value" is:

- after file loading:

3	L value A	L value B	L value C
count	AA1	AA2	AA3

- after update of value B to null value:

2	L value A	L value C
count	AA1	AA2

Example of MU usage without NU:

```
FNDEF='01,AA,5,A,MU'
```

The original content where "L" is the length of the "value" is

- after file loading:

3	L value A	L value B	L value C
count	AA1	AA2	AA3

- after update of value B to null value:

3	L value A	L value B	L value C
count	AA1	AA2	AA3

NB: Blank Compression Field Option

The NB option can be used with **LA** and **LB** fields to control blank compression. When specified, the NB option indicates that Adabas should *not* remove trailing blanks for the field; when not specified, Adabas removes trailing blanks when storing an alphanumeric or wide-character field value. If you specify the NB option for a field, you must also specify the NU or NC option for the field; NB processing requires the use of NC or NU as well.

Note: Fields specified without the NB option can lead to differences in the stored and retrieved lengths of the fields. The retrieved length of a non-NB field is likely to be smaller than the length specified for the field when it is stored due to blank compression. This may matter if the value is not really a character string, but rather a binary value that happens to end with the character codes for a blank. Therefore, if you want the stored and retrieved lengths of a field to be the same, use the NB option.

NU: Null Value Suppression Field Option

NU suppresses null values occurring in the field.

Normal compression (NU or FI not specified) represents a null value with two bytes (the first for the value length, and the second for the value itself, in this case a null). Null value suppression represents an empty field with a one-byte empty field indicator. The null value itself is not stored.

A series of consecutive fields containing null values and specifying the NU option is represented by a one-byte empty field (binary 11nnnnnn) indicator, where *nnnnnn* is the number of the fields' successive bytes containing null values, up to a total of 63. For this reason, fields defined with the NU option should be grouped together whenever possible.

If the NU option is specified for a descriptor, any null values for the descriptor are not stored in the inverted list. Therefore, a find command in which this descriptor is used and for which a null value is used as the search value will always result in no records selected, even though there may be records in Data Storage that contain a null value for the descriptor. If a descriptor defined with the NU option is used to control a logical sequence in a read logical sequence (L3/L6) command, those records that contain a null value for the descriptor will not be read.

Descriptors to be used as a basis for file coupling and for which a large number of null values exist should be specified with the NU option to reduce the total size of the coupling lists.

The NU option cannot be specified for fields defined with the combined NC/NN options or with the FI option.

Example of NU usage:

	Definition	User Data	Internal Representation
Normal Compression	FNDEF='01,AA,2,B'	0000	0200 (2 bytes)
With FI Option	FNDEF='01,AA,2,B,FI'	0000	0000 (2 bytes)
With NU Option	FNDEF='01,AA,2,B,NU'	0000	C1 (1 byte)*

** C1 indicates 1 empty field.*

NV: No Conversion Field Option

The "do not convert" option for alphanumeric (A) or wide-character (W) format fields specifies that the field is to be processed in the record buffer without being converted.

Fields with the NV option are not converted to or from the user: the field has the characteristics of the file encoding; that is, the default blank

- for A fields, is always the EBCDIC blank (X'40'); and
- for W fields, is always the blank in the file encoding for W format.

The NV option is used for fields containing data that cannot be converted meaningfully or should not be converted because the application expects the data exactly as it is stored.

The field length for NV fields is byte-swapped if the user architecture is byte-swapped.

For NV fields, "A" format cannot be converted to "W" format and vice versa.

SY: System Field

Use the SY field option to identify a field as a system field and to specify the type of information stored in the system field. A system field is a field in an Adabas file whose value is automatically set by the Adabas nucleus when records are inserted or updated on the file. Optionally, you can specify that some system field values only be set when records are inserted using the **CR field option**. A system field *cannot* be a **PE group field**.

System fields containing the following types of information can currently be defined in an Adabas file:

- Job name: The job name of the user inserting or updating a record.
- ETID: The eight-byte user ID of the user inserting or updating a record. This is the user ID set in the Additions 1 field of an OP (open) command for the user session.
- Session ID: The 28-byte user ID of the user inserting or updating a record.
- Session user: The last eight bytes of the 28-byte session ID or the user inserting or updating a record.
- Security system user ID: The eight-byte security system ID of the user inserting or updating a record.
- Time: The date or date and time at which a record is inserted or updated. The format of the stored date and time is defined by the **DT (date-time edit mask) field option**.

Valid values are:

Value	The system field is ...	Field Format	Field Length (bytes)
JOBNAME	A job name system field.	Alphanumeric	8
OPUSER	An ETID system field.	Alphanumeric	8
SESSIONID	A session ID system field.	Alphanumeric	28
SESSIONUSER	A session user system field.	Alphanumeric	8
SECUID	A security system user ID system field.	Alphanumeric	8
TIME	A date or date and time system field. The format of the stored date and time is defined by the **DT (date-time edit mask) field option**. A DT field option is required in a system field definition if SY is set to TIME.	varies, based on the edit mask	varies, based on the edit mask

For complete information about system fields and the rules surrounding them, read *System Fields*, in *Adabas Concepts Manual*.

TZ: Time Zone Field Option

The TZ field option identifies a date-time field that should be presented in the user's local time and stored in UTC time, allowing for differences in time zones. There is no specific syntax for the TZ field option as there are no parameters; simply specifying TZ in the field definition of a date-time field provides time zone support.

When TZ is specified, date-time values are converted and displayed in the user's local time, but are stored in *coordinated universal (UTC) time*. This allows users in different time zones to view the data in their individual local times, but still share the same data. Storing values in standardized UTC time makes them easily comparable.

Adabas uses the time zone data taken from the **tz database**, which is also called the *zoneinfo* or *Olson* database. The specific list of time zone names that Adabas supports in any given release can be found in the TZINFO member of the Adabas source library (ADA*vrs*.SRCE). For more information about the TZINFO member of the time zone library, read *Supported Time Zones*, in the *Adabas DBA Tasks Manual*.

The TZ option can be specified in field definitions that use the following **date-time edit masks**:

- DATETIME
- TIMESTAMP
- NATTIME
- UNIXTIME
- XTIMESTAMP

You cannot use the TZ option in field definitions that use the DATE, TIME, or NATDATE date-time edit masks because the timezone offsets depend on the presence of both date and time values in the data.

Note that UNIXTIME and XTIMESTAMP fields are by definition based on the UTC; standard conversion routines will perform time zone handling outside of Adabas. In other words, the TZ option has no effect when reading or writing fields with the UNIXTIME or XTIMESTAMP edit mask.

However, when the DATETIME, NATTIME, and TIMESTAMP edit masks are set, the TZ option will convert the times to local time; otherwise they will be converted and stored as UTC times.

For example, if a date-time field is stored in UTC format is February 14, 2009, 16:00 hours, user A in time zone America/New_York will see the field displayed as February 14, 2009, 11:00 hours or 10:00 (UTC time minus 5 or 6 hours, depending on the differences in daylight savings time). Al-

ternately, user G in time zone Europe/Berlin will see the field as February 14, 2009, 17:00 hours (UTC time plus 1).

For information on the conversions between date-time fields defined with the TZ option, read *Conversions Between Date-Time Representations for Fields with the TZ option*, in the *Adabas DBA Tasks Manual*.

UQ: Unique Descriptor Field Option

UQ indicates that the field is to be a unique descriptor. A unique descriptor must contain a different value for each record in the file. In FNDEF statements, the UQ option can only be specified if the DE option is also specified. The UQ option can also be used in SUBDE, SUPDE, and HYPDE statements.

The UQ option *must* be specified if the field is to be used as an ADAM descriptor (see the **ADAMER utility**).

ADACMP does not check for unique values; this is done by the ADALOD utility, or by the ADAINV utility when executing the INVERT function. If a non-unique value is detected during file loading, ADALOD terminates with an error message.

Because ADAINV and ADALOD must execute separately for each file in an expanded file chain, they cannot check for uniqueness across the chain. However, Adabas does check the value of unique descriptors across an expanded file chain. If the value being added (N1/N2) or updated (A1) is not unique across all files within the chain, response code 198 (ADARSP198) is returned.

XI: Exclude Instance Number Field Option

By default, the occurrence number of fields within periodic groups (PE) defined as unique descriptors (UQ) is included as part of the descriptor value. This means that the same field value can occur in different periodic group occurrences in different records.

The XI option is used to exclude the occurrence number from the descriptor value for the purpose of determining the value's uniqueness. If the XI option is set, any field value can occur at most once over all occurrences of the PE field in all records.

Representing SQL Null Value

Adabas includes two data definition options, NC and NN, to provide SQL-compatible null representation for Software GmbH's mainframe Adabas SQL Gateway (ACE) and other Structured Query Language (SQL) database query languages.

The NC and NN options *cannot* be applied to fields defined

- with Adabas null suppression (NU)
- with fixed-point data type (FI)

- with multiple-values (MU)
- within a periodic group (PE)
- as group fields

In addition, the NN option can only be specified for a field that specifies the NC option.

A parent field for sub-/superfields or sub-/superdescriptors can specify the NC option. However, parent fields for a single superfield or descriptor cannot use a mix of NU and NC fields. If any parent field is NC, no other parent field can be an NU field, and vice versa.

Examples:

A correct ADACMP COMPRESS FNDEF statement for defining the field AA and assigning the NC and NN option:

```
ADACMP   FNDEF='01,AA,4,A,NN,NC,DE'
```

Incorrect uses of the NC/NN option that would result in an ADACMP utility ERROR-127:

Incorrect Example	Reason
ADACMP FNDEF='01,AA,4,A,NC,NU'	NU and NC options are not compatible
ADACMP FNDEF='01,AB,4,A,NC,FI'	NC and FI options are not compatible
ADACMP FNDEF='01,PG,PE' ADACMP FNDEF='02,P1,4,A,NC'	NC option within a PE group is not allowed

This section covers the following topics:

- NC: SQL Null Value Option
- Null Indicator Value
- NN: SQL Not Null Option

NC: SQL Null Value Option

Without the NC (not counted) option, a null value is either zero or blank depending on the field's format.

With the NC option, zeros or blanks specified in the record buffer are interpreted according to the ***null indicator value***: either as true zeros or blanks (that is, as *significant* nulls) or as undefined values (that is, as true SQL or *insignificant* nulls).

If the field defined with the NC option has no value specified in the record buffer, the field value is always treated as an SQL null.

When interpreted as a true SQL null, the null value satisfies the SQL interpretation of a field having no value. This means that no field value has been entered; that is, the field's value is not defined.

The null indicator value is thus responsible for the internal Adabas representation of the null. For more information, read the next section in this guide, ***Null Indicator Value*** , and read *Search Buffers* in the *Adabas Command Reference Guide*.

The following rules apply when compressing or decompressing records containing NC fields:

1. If the FORMAT parameter is specified, ADACMP behaves in the same way the nucleus does for update-type commands. See the *Adabas Command Reference Guide* documentation.
2. If the FORMAT parameter is *not* specified:
   - For *compression*:

     Only the value of the NC field is placed in the input record; the two null value indicator bytes must be omitted. The value is compressed as if the null value indicator bytes were set to zero. It is not possible to assign a null value to an NC field using this method.

     Example:

Field Definition Table (FDT) definition	`FNDEF='01,AA,4,A,NC'`
Input record contents:	`MIKE`

   - For *decompression*:

     If the value of an NC field is *not significant*, the record is written to DDFEHL (or FEHL) with response code 55 (ADARSP055).

     If the value of an NC field is *significant*, the value is decompressed as usual. There are no null indicator bytes.

     Example:

Field Definition Table (FDT) definition	`FNDEF='01,AA,4,A,NC'`
Output record contents	`MIKE`

Null Indicator Value

The *null indicator* value is always two bytes long and has fixed-point format, regardless of the data format. It is specified in the record buffer when a field value is added or changed; it is returned in the record buffer when the field value is read.

For an update (Ax) or add (Nx) command, the null indicator value must be set in the record buffer position that corresponds to the field's designation in the format buffer. The setting must be one of the following:

Hex Value	Indicates that . . .
FFFF	the field's value is set to "undefined", an insignificant null; the differences between no value, binary zeros, or blanks for the field in the record buffer are ignored; all are interpreted equally as "no value".
0000	no value, binary zeros, or blanks for the field in the record buffer are interpreted as significant null values.

For a read (L*x*) or find with read (S*x* with format buffer entry) command, your program must examine the null indicator value (if any) returned in the record buffer position corresponding to the field's position in the format buffer. The null indicator value is one of the following values, indicating the meaning of the actual value that the selected field contains:

Hex Value	Indicates that . . .
FFFF	a zero or blank in the field is not significant.
0000	a zero or blank in the field is a significant value; that is, a true zero or blank.
xxxx	the field is truncated. The null indicator value contains the length (xxxx) of the entire value as stored in the database record if the length is less than 32768.
0001	the field is significant and the value is truncated, and the length of the value does not fit into the S element because it is greater than 32 767.

Example:

The field definition of a null represented in a two-byte Adabas binary field AA defined with the NC option is:

01,AA,2,B,NC

For a . . .	Null Indicator Value (Record Buffer)	Data	Adabas Internal Representation
non-zero value	0 (binary value is significant)	0005	0205
blank	0 (binary null is significant)	0000 (zero)	0200
null	FFFF (binary null is not significant)	(not relevant)	C1

NN: SQL Not Null Option

The NN (*not null* or *null value not allowed*) option may only be specified when the NC option is also specified for a data field. The NN option indicates that an NC field must always have a value (including zero or blank) defined; it cannot contain "no value".

The NN option ensures that the field will not be left undefined when a record is added or updated; a significant value must always be set in the field. Otherwise, Adabas returns a response code 52 (ADARSP052).

The following example shows how an insignificant null would be handled in a two-byte Adabas alphanumeric field AA when defined with and without the NN option:

Example

An insignificant null handled in a two-byte Adabas alphanumeric field AA when defined with and without the NN option is as following:

Option	Field Definition	Null Indicator Value	Adabas Internal Representation
With NN	01,AA,2,A,NC,NN	FFFF (insignificant null)	none; response code 52 (ADARSP052) occurs
Without NN	01,AA,2,A,NC	FFFF (insignificant null)	C1

FNDEF: Periodic Group Definition

The syntax used in constructing periodic group definition entries is:

```
FNDEF = 'level, name [ , PE [(occurrences )] ]'
```

Level number and name are required. Any number of spaces may be inserted between definition entries.

Each FNDEF parameter in these definitions is described in this section.

- level
- name
- PE: Periodic Group

- occurrences

level

The level number is a one- or two-digit number in the range 01-07 (the leading zero is optional) used in conjunction with field grouping. Fields assigned a level number of 02 or greater are considered to be a part of the immediately preceding group which has been assigned a lower level number.

The definition of a group enables reference to a series of fields (may also be only 1 field) by using the group name. This provides a convenient and efficient method of referencing a series of consecutive fields.

Level numbers 01-06 may be used to define a group. A group may consist of other groups. When assigning the level numbers for nested groups, no level numbers may be skipped.

In the following example, fields A1 and A2 are in group GA. Field B1 and group GC (consisting of fields C1 and C2) are in group GB:

FNDEF='01,GA'	group	
FNDEF='02,A1,...'		elementary or multiple-value field
FNDEF='02,A2,...'		elementary or multiple-value field
FNDEF='01,GB'	group	
FNDEF='02,B1,...'		elementary or multiple-value field
FNDEF='02,GC'	group (nested)	
FNDEF='03,C1,...'		elementary or multiple-value field
FNDEF='03,C2,...'		elementary or multiple-value field

name

The name to be assigned to the field (or group).

Names must be unique within a file. Names are case-sensitive and must be two characters long: the first character must be alphabetic; the second character can be either alphabetic or numeric. No special characters are permitted.



Note: Lowercase fields will not display correctly (they will be converted to uppercase) if you use the ADARUN parameter settings MSGCONSL=UPPER, MSGDRUCK=UPPER, or MSGPRINT=UPPER.

The values E0-E9 are reserved as edit masks and may not be used.

Valid Names	Invalid Names
AA	A (not two characters)
B4	E3 (edit mask)
S3	F* (special character)
wm	6M (first character not alphabetic)
wM	
Wm	

PE: Periodic Group

The periodic group field option, indicating that the group field is to be followed by a periodic group definition that may occur multiple times in a given record. If the **MUPEX** parameter is specified and the **MUPECOUNT** parameter is set to "2", the periodic group may occur up to 65,534 times in a single record. If these parameters are not set, the periodic group may occur up to 191 times in a single record. At least one occurrence (even if it contains all null values) must be present in each ADACMP input record.

A periodic group:

- May comprise one or more fields. A maximum of 254 elementary fields may be specified. Descriptors and/or multiple value fields and other groups may be specified, but a periodic group may not contain another periodic group.
- If the **MUPEX** parameter is specified and the **MUPECOUNT** parameter is set to "2", a periodic group may occur from 0 to 65,534 times within a given record, although at least one occurrence (even if it contains all null values) must be present in each ADACMP input record. If these parameters are not set, the periodic group may occur from 0 to 191 times within a given record.
- Must be defined at the 01 level. All fields in the periodic group must immediately follow and must be defined at level 02 or higher (in increments of 1 to a maximum of 7). The next 01 level definition indicates the end of the current periodic group.
- May only be specified with a group name. Length and format parameters may not be specified with the group name.

For information on how to identify MU and PE occurrences greater than 191 in the compressed record, read *Identifying MU and PE Occurrences Greater Than 191 in Compressed Records*, elsewhere in this section.

The following are two examples of period group definitions:

Periodic Group GA:

```
FNDEF='01,GA,PE'
FNDEF='02,A1,6,A,NU'
FNDEF='02,A2,2,B,NU'
FNDEF='02,A3,4,P,NU'
```

In this example, periodic group GA consists of fields A1, A2, and A3. The number of occurrences of the periodic group in a record is defined as a one- or two-byte binary value before each occurrence group in every record (depending on the setting of the **MUPECOUNT** parameter).

Periodic Group GB:

```
FNDEF='01,GB,PE(3)'
FNDEF='02,B1,4,A,DE,NU'
FNDEF='02,B2,5,A,MU(2),NU'
FNDEF='02,B3'
FNDEF='03,B4,20,A,NU'
FNDEF='03,B5,7,U,NU'
```

In this example, periodic group GB consists of fields B1, B2, and group B3 (which includes fields B4 and B5). Three (3) occurrences of the periodic group can occur in a record.

occurrences

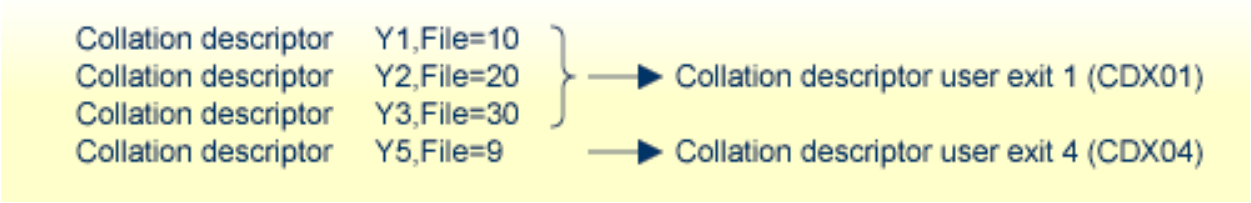
The number of occurrences of PE fields that will occur in a record if the MUPEX option is specified. This is an optional parameter.

COLDE: Collation Descriptor Definition

The collation descriptor option enables descriptor values to be sorted (collated) based on a user-supplied algorithm.

The values are based on algorithms coded in special collation descriptor user exits (CDX01 through CDX08). Each collation descriptor must be assigned to a user exit, and a single user exit may handle multiple collation descriptors.

Example:



The Collation Exit functions are called on the following events:

INITIALIZE function

- nucleus session start
- utility initialization when collation exits have been defined (ADARUN parameters)

ENCODE function

- update/insert/delete of the parent's value (Nucleus)
- Search specifying the collation descriptor with the search value (Nucleus)
- compression of a record (ADACMP)

DECODE function

Read Index (L9) by Collation DE, only if the exit supports the DECODE function (Nucleus)

Input parameters supplied to the user exit are described in *Collation Descriptor Exits 01 - 08* in *Adabas User, Hyperdescriptor, Collation Descriptor, and SMF Exits Manual*. They include the:

- address and length of input string
- address and size of output area
- address of fullword for the returned output string length.

The user exit sets the length of the returned output string.

Read *CDXnn : Collation Descriptor User Exit* in *Adabas Operations Manual* for more information.

Notes:

1. A collation descriptor can be defined for an alphanumeric (A) or wide character (W) parent field. The format, length, and options (except UQ and XI) are taken from the parent field defined in the COLDE parameter. The unique descriptor (UQ) and exclude index (XI) options are separately defined for the collation descriptor itself.
2. A search using a collation descriptor value is performed in the same manner as for standard descriptors.

3. The user is responsible for creating correct collation descriptor values. There is no standard way to check the values of a collation descriptor for completeness against the Data Storage. The maintenance utility ADACHK only checks the structure of an index, not the contents. The user must set the rules for each value definition and check the value for correctness.
4. If a file contains more than one collation descriptor, the assigned exits are called in the alphabetical order of the collation descriptor names.

Collation Descriptor Syntax

A collation descriptor is defined using the following syntax:



where:

number	is the user exit number to be assigned to the collation descriptor. The Adabas nucleus uses this number to determine the collation descriptor user exit to be called.
name	is the name to be used for the collation descriptor. The naming conventions for collation descriptors are identical to those for Adabas field names.
UQ	indicates that the unique descriptor option is to be assigned to the collation descriptor.
XI	indicates that the uniqueness of the collation descriptor is to be determined with the index (occurrence) number excluded.
parent-field	is the name of an elementary A or W field. A collation descriptor can have one parent field. The field name and address is passed to the user exit. A parent field *cannot* be a long alphanumeric LA or large object (LOB) field.

MU, NU, and PE options are taken from the parent field and are implicitly set in the collation descriptor.

If a parent field with the NU option is specified, no entries are made in the collation descriptor's inverted list for those records containing a null value for the field. This is true regardless of the presence or absence of values for other collation descriptor elements.

If a parent field is not initialized and logically falls past the end of the physical record, the inverted list entry for that record is not generated, for performance reasons. To generate the inverted list entry in this case, it is necessary to unload short, decompress, and reload the file; or use an application program to initialize the field for each record of the file.

Collation Descriptor Definition Example:

Field definition:

```
FNDEF='01,LN,20,A,DE,NU'    Last-Name
```

Collation descriptor definition:

```
COLDE='1,Y2=LN'
```

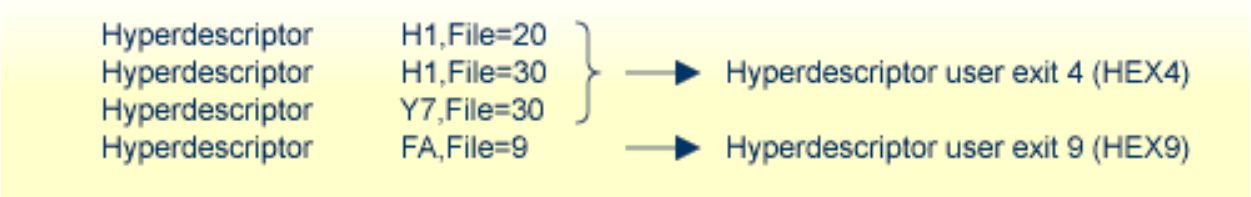
- Collation descriptor user exit 1 (CDX01) is assigned to this collation descriptor, and the name is Y2.
- The collation descriptor length (20 characters), format (alphanumeric), and null suppression (NU) are taken from the parent field LN.
- The values for the collation descriptor are to be derived from the parent field LN.

HYPDE: Hyperdescriptor Definition

The hyperdescriptor option enables descriptor values to be generated, based on a user-supplied algorithm.

The values are based on algorithms coded in special hyperdescriptor user exits (HEX01 through HEX31). Each hyperdescriptor must be assigned to a user exit, and a single user exit may handle multiple hyperdescriptors.

Example:



The exit is called whenever a hyperdescriptor value is to be generated by the Adabas nucleus or by the ADACMP utility.

Input parameters supplied to the user exit are:

- hyperdescriptor name
- file number
- addresses of fields taken from the Data Storage record, together with field name and PE index (if applicable). These addresses point to the compressed values of the fields. The names of these fields must be defined using the HYPDE parameter of ADACMP or ADAINV.

The user exit must return the descriptor value(s) (DVT) in compressed format. No value, or one or more values may be returned depending on the options (PE, MU) assigned to the hyperdescriptor.

The original ISN assigned to the input value(s) may be changed.

For complete information about hyperdescriptor user exits, read *Hyperdescriptor Exits 01 - 31* in *Adabas User, Hyperdescriptor, Collation Descriptor, and SMF Exits Manual*.

Notes:

1. The format, the length, and the options of a hyperdescriptor are user-defined. They are not taken from the parent fields defined in the HYPDE parameter.
2. A search using a hyperdescriptor value is performed in the same manner as for standard descriptors.
3. The user is responsible for creating correct hyperdescriptor values. There is no standard way to check the values of a hyperdescriptor for completeness against the Data Storage. The maintenance utility ADACHK only checks the structure of an index, not the contents. The user must set the rules for each value definition and check the value for correctness.
4. If a hyperdescriptor is defined as packed or unpacked format, Adabas checks the returned values for validity. The sign half-byte for packed values can contain A, C, E, F (positive) or B, D (negative). Adabas converts the sign to F or D.
5. If a file contains more than one hyperdescriptor, the assigned exits are called in the alphabetical order of the hyperdescriptor names.

Hyperdescriptor Syntax

A hyperdescriptor is defined using the following syntax:

```
HYPDE = 'number, name, length, format [ { , option} ... ] = { parent-field } , ...'
```

where

number	is the user exit number to be assigned to the hyperdescriptor. The Adabas nucleus uses this number to determine the hyperdescriptor user exit to be called.
name	is the name to be used for the hyperdescriptor. The naming conventions for hyperdescriptors are identical to those for Adabas field names.
length	is the default length of the hyperdescriptor.
format	is the format of the hyperdescriptor:

Format	**Maximum Length**
Alphanumeric (A)	253 bytes
Binary (B)	126 bytes
Fixed Point (F)	4 bytes (always 4 bytes)
Floating Point (G)	8 bytes (always 4 or 8 bytes)
Packed Decimal (P)	15 bytes
Unpacked Decimal (U)	29 bytes

Note: Wide-character (W) format is not valid for a hyperdescriptor.

option	is an option to be assigned to the hyperdescriptor. The following options may be used together with a hyperdescriptor: ■ MU (multiple-value field) ■ NU (null-value suppression) ■ PE (field of a periodic group) ■ UQ (unique descriptor) The parent field of a hyperdescriptor *cannot* be a long alphanumeric (LA) field.
parent-field	is the name of an elementary field. A hyperdescriptor can have 1-20 parent fields. The field names and addresses are passed to the user exit. A parent field *cannot* be a long alphanumeric LA or large object (LOB) field. **Note:** A hyperdescriptor parent-field may not have W (wide-character) format.

A parent field with both the NU option specified and a null value doesn't provide the hyperdescriptor exit with an input parameter element for the null parent field. Input parameters for other parent fields may still be provided to the exit. (See Null Value Option for details).

If a parent field is not initialized and logically falls past the end of the physical record, it is treated like a field with the NU option containing a null value (i.e., no value).

Hyperdescriptor Definition Example:

Field definitions:

```
FNDEF='01,LN,20,A,DE,NU'   Last-Name
FNDEF='01,FN,20,A,MU,NU'   First-Name
FNDEF='01,ID,4,B,NU'       Identification
FNDEF='01,AG,3,U'          Age
FNDEF='01,AD,PE'           Address
FNDEF='02,CI,20,A,NU'      City
FNDEF='02,ST,20,A,NU'      Street
FNDEF='01,FA,PE'           Relatives
FNDEF='02,NR,20,A,NU'      R-Last-Name
FNDEF='02,FR,20,A,MU,NU'   R-First-Name
```

Hyperdescriptor definition:

```
HYPDE='2,HN,60,A,MU,NU=LN,FN,FR'
```

- Hyperdescriptor user exit 2 is assigned to this hyperdescriptor, and the name is HN.
- The hyperdescriptor length is 60, the format is alphanumeric, and is a multiple-value (MU) field with null suppression (NU).
- The values for the hyperdescriptor are to be derived from fields LN, FN and FR.

The ADACMP HYPDE= statement may be continued on another line, as shown in the following example. To do so, first specify a minus (-) after a whole argument and before the closing apostrophe on the first line. Then enter the remaining positional arguments, beginning after the statement name (ADACMP) enclosed in apostrophes on the following line:

```
ADACMP HYPDE='1,HY,20,A=AA,BB,CC,-'
ADACMP                'DD,EE,FF'
```

PHONDE: Phonetic Descriptor

The use of a phonetic descriptor in a FIND command results in the return of all the records that contain similar phonetic values. The phonetic value of a descriptor is based on the first 20 bytes of the field value. Only alphabetic values are considered; numeric values, special characters, and blanks are ignored. Lower- and uppercase alphanumeric characters are internally identical.

A phonetic descriptor is defined using the following syntax:

PHONDE = '*name* **(***field***)'**

where

name	is the name to be used for the phonetic descriptor. The naming conventions for phonetic descriptors are identical to those for Adabas field names.
field	is the name of the field to be used for the phonetic descriptor.

The field *must* be

- an elementary or a multiple value field; and
- defined with alphanumeric format.

The field can be a descriptor.

The field *cannot* be

- a subdescriptor, superdescriptor, or hyperdescriptor;
- contained within a periodic group;
- used as the source field for more than one phonetic descriptor.
- format W (wide-character)

The parent field of a phonetic descriptor *cannot* be a long alphanumeric (LA) or large object (LOB) field.

If the field is defined with the NU option, no entries are made in the phonetic descriptor's inverted list for those records that contain a null value (within the byte positions specified) for the field. The format is the same as for the field.

If the field is not initialized and logically falls past the end of the physical record, the inverted list entry for that record is not generated for performance reasons. To generate the inverted list entry in this case, it is necessary to unload short, decompress, and reload the file; or use an application program to initialize the field for each record of the file.

Phonetic Descriptor Definition Example:

Field definition:

```
FNDEF='01,AA,20,A,DE,NU'
```

Phonetic definition:

```
PHONDE='PA(AA)'
```

SUBDE: Subdescriptor Definition

A subdescriptor is a descriptor created from a portion of an elementary field. The elementary field may or may not be a descriptor itself. A subdescriptor can also be used as a subfield; that is, it can be specified in the format buffer to control the record's output format.

A subdescriptor definition is entered using the following syntax:

SUBDE = '*name* **[, UQ [, XI]] =** *parent-field* **(***begin*, *end***)'**

where

name	is the subdescriptor name. The naming conventions for a subdescriptor are identical to those for Adabas field names.
UQ	indicates that the subdescriptor is to be defined as unique (see the **definition of option UQ**).
XI	indicates that the uniqueness of the subdescriptor is to be determined with the index (occurrence) number excluded (see the **definition of option XI**).
parent-field	is the name of the field from which the subdescriptor is to be derived. A parent field *cannot* be a large object (LOB) field.
*begin**	is the relative byte position within the parent field where the subdescriptor definition is to begin.
*end**	is the relative byte position within the parent field where the subdescriptor definition is to end.

** Counting is from left to right beginning with 1 for alphanumeric or wide-character fields, and from right to left beginning with 1 for numeric or binary fields. If the parent field is defined with P format, the sign of the resulting subdescriptor value is taken from the 4 low-order bits of the low-order byte (that is, byte 1).*

A parent field of a subdescriptor can be

- an elementary field;
- a multiple-value field (but *not* a particular occurrence of a multiple-value field);
- contained within a periodic group (but *not* a particular occurrence of a periodic group);
- a long alphanumeric (LA) field (for parents with Alpha (A) or Wide (W) format the begin and end values must be in the 1 to 253 range).

The parent field of a subdescriptor *cannot* be

- a sub/super field, subdescriptor, superdescriptor, or phonetic descriptor;

- format G (floating point);
- a large object (LB) field.

If the parent field is defined with the NU option, no entries are made in the subdescriptor's inverted list for those records that contain a null value (within the byte positions specified) for the field. The format is the same as for the parent field.

If a parent field is not initialized and logically falls past the end of the physical record, the inverted list entry for that record is not generated for performance reasons. To generate the inverted list entry in this case, it is necessary to unload short, decompress, and reload the file; or use an application program to initialize the field for each record of the file.

Subdescriptor Definition Example 1:

Parent-field definition:

```
FNDEF='01,AR,10,A,NU'
```

Subdescriptor definition:

```
SUBDE='SB=AR(1,5)'
```

The values for subdescriptor SB are derived from the first five bytes (counting from left to right) of all the values for the parent field AR. All values are shown in character format.

AR Values	SB Values
DAVENPORT	DAVEN
FORD	FORD
WILSON	WILSO

Subdescriptor Definition Example 2:

Parent-field definition:

```
FNDEF='02,PF,6,P'
```

Subdescriptor definition:

```
SUBDE='PS=PF(4,6)'
```

The values for subdescriptor PS are derived from bytes 4 to 6 (counting from right to left) of all the values for the parent field PF. All values are shown in hexadecimal.

PF Values	PS Values
00243182655F	02431F
00000000186F	0F (see note)
78426281448D	0784262D

Note: If the NU option had been specified for parent field PF, no value would have been created for PS for this value.

Subdescriptor Definition Example 3:

Source-field definition:

```
FNDEF='02,PF,6,P'
```

Subdescriptor definition:

```
SUBDE='PT=PF(1,3)'
```

The values for PT are derived from bytes 1 to 3 (counting from right to left) of all the values for PF. All values are shown in hexadecimal.

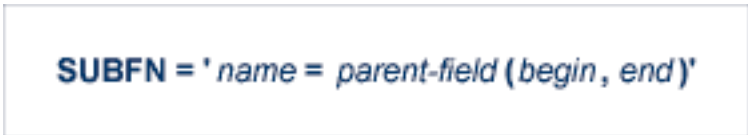
PF Values	PT Values
00243182655F	82655F
00000000186F	186F
78426281448D	81448D

SUBFN: Subfield Definition

A subfield:

- is a portion of an elementary field that can be read using an Adabas read command;
- cannot be updated;
- can be changed to a subdescriptor using ADAINV INVERT SUBDE=... .

A subfield definition is entered using the following syntax:



where

name	is the subfield name. The naming conventions for a subfield are identical to those for Adabas field names.
parent-field	is the name of the field from which the subfield is to be derived. A parent field *cannot* be a large object (LOB) field.
*begin**	is the relative byte position within the parent field where the subfield definition is to begin.
*end**	is the relative byte position within the parent field where the subfield definition is to end.

** Counting is from left to right beginning with 1 for alphanumeric or wide-character fields, and from right to left beginning with 1 for numeric or binary fields. If the parent field is defined with "P" format, the sign of the resulting subfield value is taken from the 4 low-order bits of the low-order byte (that is, byte 1).*

The parent field for a subfield can be:

- a multiple-value field;
- within a periodic group;
- a long alphanumeric (LA) field (for parents with Alpha (A) or Wide (W) format the begin and end values must be in the 1 to 253 range).

The parent field for a subfield *cannot*:

- have format "G" (floating point);
- be a large object (LB) field.

Subfield Definition Example:

```
SUBFN='X1=AA(1,2)'
```

SUPDE: Superdescriptor Definition

A superdescriptor is a descriptor created from several fields, portions of fields, or a combination thereof.

Each source field (or portion of a field) used to define a superdescriptor is called a *parent*. From 2 to 20 parent fields or field portions may be used to define a superdescriptor. The total size must be less than or equal to 253.

Note: Mainframe Adabas databases *do not* allow fields in floating point format (format G) to be used as superdescriptor parent fields; open systems Adabas databases *do* allow fields in floating point format to be used as superdescriptor parent fields.

A superdescriptor may be defined as a unique descriptor.

A superdescriptor can be used as a superfield; that is, it can be specified in the format buffer to determine the record's output format.

This section covers the following topics:

- SUPDE Syntax
- Superdescriptor Interfaces with Adabas Commands
- Format Conversions of Superdescriptors
- SUPDE Examples

SUPDE Syntax

A superdescriptor definition has the following syntax:

```
SUPDE = 'name [, UQ [, XI ] ] = {parent-field (begin, end ) } , ...'
```

where

name	is the superdescriptor name. The naming conventions for superdescriptors are identical to those for Adabas names.
UQ	indicates that the superdescriptor is to be defined as unique (see the **definition option UQ**).
XI	indicates that the uniqueness of the superdescriptor is to be determined with the index (occurrence) number excluded (see the **definition option XI**).
parent-field	is the name of a parent field from which a superdescriptor element is to be derived; up to 20 parent fields can be specified. A parent field *cannot* be a large object (LOB) field. **Note:** Mainframe Adabas databases *do not* allow fields in floating point format (format G) to be used as superdescriptor parent fields; open systems Adabas databases *do* allow fields in floating point format to be used as superdescriptor parent fields.
*begin**	is the relative byte position within the field where the superdescriptor element begins.
*end**	is the relative byte position within the field where the superdescriptor element is to end.

** Counting is from left to right beginning with 1 for fields defined with alphanumeric or wide-character format, and from right to left beginning with 1 for fields defined with numeric or binary format. For any*

parent field except those defined as "FI", any begin and end values within the range permitted for the parent field's data type are valid.

A parent field of a superdescriptor can be:

- an elementary field;
- a maximum of one MU field (but not a specific MU field value);
- within a periodic group (but not a specific occurrence);
- a long alphanumeric (LA) field (for parents with Alpha (A) or Wide (W) format the begin and end values must be in the 1 to 253 range).

A parent field of a superdescriptor *cannot* be

- a super-, sub-, or phonetic descriptor;
- format G (floating point);
- an NC option field if another parent field is an NU option field;
- a large object (LB) field.

If a parent field with the NC or NU option is specified, no entries are made in the superdescriptor's inverted list for those records containing a null value for the field. In other words, no value is created if the parent value is empty and the NC/NU option has been specified. This is true regardless of the presence or absence of values for other superdescriptor elements.

If a parent field is not initialized and logically falls past the end of the physical record, the inverted list entry for that record is not generated for performance reasons. To generate the inverted list entry in this case, it is necessary to unload short, decompress, and reload the file; or use an application program to initialize the field for each record of the file.

The total length of any superdescriptor value may not exceed 253 bytes (alphanumeric) or 126 bytes (binary).

The superdescriptor format is B (binary) if no element of the superdescriptor is derived from an A (alphanumeric) or W (wide-character) parent field; if any element of the superdescriptor is derived from an A or W parent field, the format of the superdescriptor reflects the last occurring A or W element; for example, if the last occurring A or W element is W, the format of the superdescriptor is W.

All binary format superdescriptor values are treated as unsigned numbers.

The ADACMP SUPDE= statement may be continued on another line by specifying a minus (-) after an argument just before the closing apostrophe on the first line. Then enter the remaining positional arguments enclosed in apostrophes on the following line beginning after the statement name (ADACMP). For example:

```
ADACMP SUPDE='SI=AA(10,20),BB(20,21),-'
ADACMP       'CC(12,13),DD(14,15)'
```

Superdescriptor Interfaces with Adabas Commands

The following commands can interface with superdescriptors.

Adabas Command	Superdescriptor values can be...
N1 or A1	implicitly built with given fields at insert and update, when parent fields force the creation of superdescriptors
S*x* or L3	specified in the value buffer of search expressions and logical reads.
L9	returned in the record buffer.

Format Conversions of Superdescriptors

Superdescriptors have a final superdescriptor format which is calculated as follows:

Superdescriptor Format	Used if:
A (alphabetic)	At least one parent field has format A.
B (binary)	All other superdescriptor fields.

This section covers the following topics:

- Format Conversions During Updates
- Format Conversions In Value Buffers
- Format Conversions For Output (L9 Command)



Note: Conversions are only performed if a parent field has not been set with the NV option.

Format Conversions During Updates

Superdescriptors should be built so that they have the same collating sequence in all environments. However, problems exist for some combinations, as described in this section:

- Alphanumeric (Format A) Values in IBM (EBCDIC) and UNIX (ASCII) Environments
- Numeric Values (Format U) in IBM (EBCDIC) and UNIX (ASCII) Environments
- Binary Values (Formats B, F, G) with Big-endian and Little-endian Storage Formats

- Different Packed Value Signs (Format P)

Alphanumeric (Format A) Values in IBM (EBCDIC) and UNIX (ASCII) Environments

All alphabetic field values will be converted from EBCDIC to ASCII if an insert or update call comes from an IBM mainframe environment to a UNIX database. Consequently, the superdescriptor parent values are automatically converted to ASCII. In this case, an application might fail if it expects a specific sort sequence (for example using uppercase and lowercase characters). In EBCDIC formats, lowercase characters come prior to uppercase characters; in ASCII formats, this sequence is reversed (uppercase characters come prior to lowercase characters).

One of two methods can be used to resolve this problem:

- Use the NV option on parent fields with EBCDIC-ASCII conflicts. This will disable the EBCDIC-ASCII conversion.
- Use a hyperdescriptor instead of a superdescriptor.

Numeric Values (Format U) in IBM (EBCDIC) and UNIX (ASCII) Environments

All numeric field values will be converted from EBCDIC to ASCII if an insert or update call comes from an IBM mainframe environment to a UNIX database. Consequently, the superdescriptor parent values are automatically converted to ASCII, even if the final superdescriptor requests formats of A (alphabetic) or B (binary).

Binary Values (Formats B, F, G) with Big-endian and Little-endian Storage Formats

Some platforms store binary byte sequences in big-endian format; others store them in little-endian format. For example, IBM and HP-UX processors use big-endian format (the byte significance runs from right to left), while Intel processors use little-endian sequence (the byte significance runs from left to right).

Adabas performs conversions on superdescriptors containing binary values in swapped architectures (little-endian binary values with significance running from left to right) to get them into a standard sort sequence before storing them in the index.

- For alphabetic superdescriptors containing at least one binary field with parent lengths greater than one, the binary parent values will be swapped.
- For binary superdescriptors, the order of the parent entries will be swapped and the non-binary parent values will be swapped.

Suppose superdescriptor S1 is defined as: S1=AA(1,2),BB(1,3). Because AA is an alphabetic field, S1 is an alphabetic superdescriptor.

- The big-endian format (non-swapped) of S1 values would be stored in this sequence:

  AA1, AA2, BB3, BB2, BB1

- The little-endian format (swapped) of S1 values would be in this sequence: AA1, AA2, BB1, BB2, BB3.

Different Packed Value Signs (Format P)

Sign information of packed values is represented differently on different platforms. Adabas on open systems converts positive values (A, C, or F) to C and negative values (B or D) to D. Adabas for mainframes uses F to represent positive values. Consequently, collating sequence problems arise if packed values are used in superdescriptors because the packed value signs lose their meaning; they become normal bit patterns. When this happens, positive packed values can be sorted as negative packed values.

In addition, when combined in a superdescriptor in Adabas for mainframes, negative packed values are sorted before positive packed values, while on Adabas for open systems, positive packed values are sorted before negative packed values.

To resolve these problems, we recommend that you use a hyperdescriptor instead of a super-descriptor.

Format Conversions In Value Buffers

When superdescriptors are specified in a value buffer, they are converted so they can be matched to an associated index entry.

Format Conversions For Output (L9 Command)

Superdescriptor values retrieved by L9 commands must be converted before they are returned in the record buffer. Alphabetic fields are converted from ASCII to EBCDIC, if required. In addition, binary parts of the superdescriptor are swapped if necessary. The packed signs of packed value parts of the superdescriptor are not converted.

SUPDE Examples

- Superdescriptor Definition Example 1
- Superdescriptor Definition Example 2
- Superdescriptor Definition Example 3
- Superdescriptor Definition Example 4

- Superdescriptor Definition Example 5

Superdescriptor Definition Example 1

Field definitions:

```
FNDEF='01,LN,20,A,DE,NU'   Last-Name
FNDEF='01,FN,20,A,MU,NU'   First-Name
FNDEF='01,ID,4,B,NU'       Identification
FNDEF='01,AG,3,U'          Age
FNDEF='01,AD,PE'           Address
FNDEF='02,CI,20,A,NU'      City
FNDEF='02,ST,20,A,NU'      Street
FNDEF='01,FA,PE'           Relatives
FNDEF='02,NR,20,A,NU'      R-Last-Name
FNDEF='02,FR,20,A,MU,NU'   R-First-Name
```

Superdescriptor definition:

```
SUPDE='SD=LN(1,4),ID(3,4),AG(2,3)'
```

Superdescriptor SD is to be created. The values for the superdescriptor are to be derived from bytes 1 to 4 of field LN (counting from left to right), bytes 3 to 4 of field ID (counting from right to left), and bytes 2 to 3 of field AG (counting from right to left). All values are shown in hexadecimal.

LN	ID	AG	SD
C6D3C5D4C9D5C7	00862143	F0F4F3	C6D3C5D40086F0F4
D4D6D9D9C9E2	02461866	F0F3F8	D4D6D9D90246F0F3
D7C1D9D2C5D9	00000000	F0F3F6	No value is stored (because of ID)
404040404040	00432144	F0F0F0	No value is stored (because of LN)
C1C1C1C1C1C1	00000144	F1F1F1	C1C1C1C10000F1F1
C1C1C1C1C1C1	00860000	F0F0F0	C1C1C1C10086F0F0

The format for SD is alphanumeric since at least one element is derived from a parent field defined with alphanumeric format.

Superdescriptor Definition Example 2

Field definitions:

```
FNDEF='01,LN,20,A,DE,NU'   Last-Name
FNDEF='01,FN,20,A,MU,NU'   First-Name
FNDEF='01,ID,4,B,NU'       Identification
FNDEF='01,AG,3,U'          Age
FNDEF='01,AD,PE'           Address
FNDEF='02,CI,20,A,NU'      City
FNDEF='02,ST,20,A,NU'      Street
FNDEF='01,FA,PE'           Relatives
FNDEF='02,NR,20,A,NU'      R-Last-Name
FNDEF='02,FR,20,A,MU,NU'   R-First-Name
```

Superdescriptor definition:

```
SUPDE='SY=LN(1,4),FN(1,1)'
```

Superdescriptor SY is to be created from fields LN and FN (which is a multiple-value field). All values are shown in character format.

LN	FN	SY
FLEMING	DAVID	FLEMD
MORRIS	RONALD RON	MORRR MORRR
WILSON	JOHN SONNY	WILSJ WILSS

The format of SY is alphanumeric since at least one element is derived from a parent field defined with alphanumeric format.

Superdescriptor Definition Example 3

Field definitions:

```
FNDEF='01,PN,6,U,NU'
FNDEF='01,NA,20,A,DE,NU'
FNDEF='01,DP,1,B,FI '
```

Superdescriptor definition:

```
SUPDE='SZ=PN(3,6),DP(1,1)'
```

Superdescriptor SZ is to be created. The values for the superdescriptor are to be derived from bytes 3 to 6 of field PN (counting from right to left), and byte 1 of field DP. All values are shown in hexadecimal.

PN	DP	SZ
F0F2F4F6F7F2	04	F0F2F4F604
F8F4F0F3F9F8	00	F8F4F0F300
F0F0F0F0F1F1	06	F0F0F0F006
F0F0F0F0F0F1	00	F0F0F0F000
F0F0F0F0F0F0	00	no value is stored (because of PN)
F0F0F0F0F0F0	01	no value is stored (because of PN)

The format of SZ is binary since no element is derived from a parent field defined with alphanumeric format. A null value is not stored for the last two values shown because the superdescriptor option is NU (from the PN field) and the PN field value contains unpacked zeros (X'F0'), the null value.

Superdescriptor Definition Example 4

Field definitions:

```
FNDEF='01,PF,4,P,NU'
FNDEF='01,PN,2,P,NU'
```

Superdescriptor definition:

```
SUPDE='SP=PF(3,4),PN(1,2)'
```

Superdescriptor SP is to be created. The values for the superdescriptor are to be derived from bytes 3 to 4 of field PF (counting from right to left), and bytes 1 to 2 of field PN (counting from right to left). All values are shown in hexadecimal.

PF	PN	SP
0002463F	003F	0002003F
0000045F	043F	0000043F
0032464F	000F	No value is stored (because of PN)
0038000F	044F	0038044F

The format of SP is binary since no element is derived from a parent field defined with alphanumeric format.

Superdescriptor Definition Example 5

Field definitions:

```
FNDEF='01,AD,PE'
FNDEF='02,CI,4,A,NU'
FNDEF='02,ST,5,A,NU'
```

Superdescriptor definition:

```
SUPDE='XY=CI(1,4),ST(1,5)'
```

Superdescriptor XY is to be created from fields CI and ST. All values are shown in character format.

CI	ST	XY
(1st occ.) BALT	(1st occ.) MAIN	BALTMAIN
(2nd occ.) CHI	(2nd occ.) SPRUCE	CHI SPRUC
(3rd occ.) WASH	(3rd occ.) 11TH	WASH11TH
(4th occ.) DENV	(4th occ.) bbbbb	No value stored (because of ST)

The format of XY is alphanumeric since at least 1 element is derived from a parent field which is defined with alphanumeric format.

SUPFN: Superfield Definition

A superfield is a field composed of several fields, portions of fields, or combinations thereof, which may be read using an Adabas read command. A superfield *cannot*

- be updated;
- comprise fields defined with the NC option if another parent field has the NU option;
- be used as a descriptor.

A superfield *can* be changed to a superdescriptor using the ADAINV utility function INVERT SUPDE=....

A superfield is defined using the following syntax:

SUPFN = '*name* **=** *parent-field* **(***begin* **,** *end* **)[,** *parent-field* **(** *begin* **,** *end* **)]...'**

where

name	superfield name. The naming conventions for superfields are identical to those for Adabas names.
parent-field	name of the field from which a superfield element is to be derived. A parent field *cannot* be a large object (LOB) field.
*begin**	relative byte position within the field where the superfield element is to begin.
*end**	relative byte position within the field where the superfield element is to end.

** Counting is from left to right beginning with 1 for fields defined with alphanumeric or wide-character format, and from right to left beginning with 1 for fields defined with numeric or binary format.*

A parent field of a superfield can be:

- a multiple-value field;
- contained within a periodic group;
- a long alphanumeric (LA) field (for parents with Alpha (A) or Wide (W) format the begin and end values must be in the 1 to 253 range).

A parent field of a superfield *cannot*:

- have format "G" (floating point);
- be a large object (LB) field.

The total length of any superfield value may not exceed 253 bytes (alphanumeric) or 126 bytes (binary).

The superfield format is B (binary) if no element of the superfield is derived from an A (alphanumeric) or W (wide-character) parent field; if any element of the superfield is derived from an A or W parent field, the format of the superfield reflects the last occurring A or W element; for example, if the last occurring A or W element is W, the format of the superfield is W.

Superfield Definition Example:

```
SUPFN='X2=AA(1,2),AB(1,4),AC(1,1)'
```

40 JCL Requirements and Examples

- User Exits with ADACMP 276
- Requirements to Run ADACMP JCL 277

This section describes the job control information required to run ADACMP on the z/OS operating system and shows examples of job streams.

Note: When the recovery log is active, sequential data sets used by the utilities whose runs are logged on the RLOG must be kept and made available for any recovery operation; for example, the DDEBAND input to an ADALOD LOAD operation.

User Exits with ADACMP

- Compression with User Exit
- Collation with User Exit

Compression with User Exit

User exit 6 can be used to perform user processing on a record before it is processed by the ADACMP COMPRESS utility. It can also be used to control the sequence and contents of the de-compressed records that are output from the ADACMP DECOMPRESS utility; when used in this way, the user exit controls which decompressed records ADACMP writes to the DDAUSBA data set. For more information about user exit 6, read *User Exit 6 (User Processing Before Data Compression)* in *Adabas Utilities Manual*.

If user exit 6 is to be used during ADACMP execution, the specified user exit routine must be loadable at execution time; that is, it must be assembled and linked into the Adabas load library (or any library concatenated with it).

The ADACMP COMPRESS utility job must specify:

```
ADARUN UEX6 = exit-name
```

where:

exit-name	is the name of a user routine that gets control at the user exit; the name can be up to 8 characters long.

For more information, read *UEXn : User Exit* in *Adabas Operations Manual*.

Collation with User Exit

If a collation user exit is to be used during ADACMP execution, the ADARUN CDXnn parameter must be specified for the utility run.

Used in conjunction with the universal encoding support (UES), the format of the collation descriptor user exit parameter is:

```
ADARUN CDXnn= exit-name
```

where:

nn	is the number of the collation descriptor exit, a two-digit decimal integer in the range 01-08 inclusive.
exit-name	is the name of the user routine that gets control at the collation descriptor exit; the name can be up to 8 characters long.

Only one program may be specified for each collation descriptor exit. Up to 8 collation descriptor exits may be specified (in any order). For more information, read *CDXnn : Collation Descriptor User Exit* in *Adabas Operations Manual*.

Requirements to Run ADACMP JCL

Data Set	DD Name	Storage	More Information
User input data (COMPRESS function)	DDEBAND	tape/ disk	
Compressed data (DECOMPRESS function)	DDEBAND	tape/ disk	Not used if the parameter INFILE is specified
Compressed data for a data base with files containing large object (LB) fields (COMPRESS function)	DDAUSB1	tape/disk	This additional data set receives the compressed large object records to be loaded into the *LOB file* as the compressed records in the first output data set (DDAUSBA) are loaded into the *base file*
Compressed data (COMPRESS function)	DDAUSBA	tape/ disk	
Decompressed data (DECOMPRESS function)	DDAUSBA	tape/ disk	
Rejected data	DDFEHL	tape/ disk	
ADACMP report	DDDRUCK	printer	
ADARUN messages	DDPRINT	printer	

Data Set	DD Name	Storage	More Information
ADARUN parameters	DDCARD	reader	
Time zone file	TZINFO	disk	Required with the TZ parameter.
ADACMP parameters and data definitions	DDKARTE	reader	

JCL Examples

In the JOBS data set, refer to ADACMP and ADACMPS for the COMPRESS examples and ADACMPD for the DECOMPRESS example.

ADACMP COMPRESS

This example can be found in the ADACMP member of the JOBS data set.

```
//ADACMP    JOB
//*
//*   ADACMP COMPRESS
//*      COMPRESS A FILE
//*
//CMP       EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD      <=== ADABAS LOAD
//TZINFO    DD   DISP=SHR,DSN=ADABAS.Vvrs.TZOO
//*
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDEBAND   DD   DISP=OLD,DSN=EXAMPLE.DByyyyy.INPUT,UNIT=TAPE, <===
//          VOL=SER=TAPE01                                     <===
//DDAUSBA   DD   DISP=(NEW,KEEP),DSN=EXAMPLE.DByyyyy.COMP01,UNIT=DISK, <==
//          VOL=SER=DISK01,SPACE=(TRK,(200,10),RLSE)
//DDFEHL    DD   DISP=(NEW,KEEP),DSN=EXAMPLE.DByyyyy.FEHL,UNIT=DISK, <===
//          VOL=SER=DISK01,SPACE=(TRK,1)
//DDCARD    DD   *
ADARUN PROG=ADACMP,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADACMP COMPRESS FILE=1
ADACMP FNDEF='01,AA,008,B,DE'
ADACMP FNDEF='01,BA,020,A,NU,DE'
ADACMP FNDEF='01,BB,015,A,NU,DE'
ADACMP FNDEF='01,BC,001,A,FI'
ADACMP FNDEF='01,CA,001,A,NU,DE'
ADACMP FNDEF='01,CB,002,U,NU,DE'
ADACMP FNDEF='01,CC,010,A,NU,DE'
ADACMP FNDEF='01,CD,002,U,NU,DE'
ADACMP FNDEF='01,DA,005,U,NU'
ADACMP FNDEF='01,DB,020,A,NU,DE'
ADACMP FNDEF='01,DC,015,A,NU,DE'
```

```
ADACMP FNDEF='01,DD,002,A,NU,DE'
ADACMP FNDEF='01,DE,005,U,NU,DE'
ADACMP FNDEF='01,DF,008,A,NU,DE'
ADACMP FNDEF='01,FA,020,A,NU,DE'
ADACMP FNDEF='01,FB,006,U,NU,DE'
ADACMP FNDEF='01,FC,006,U,NU'
ADACMP FNDEF='01,GA,002,U,NU'
ADACMP FNDEF='01,HA,002,U,NU'
ADACMP FNDEF='01,IA,002,U,NU'
ADACMP FNDEF='01,KA,002,U,NU'
ADACMP FNDEF='01,LA,030,A,NU,DE'
ADACMP SUBDE='SB=DE(3,5)'
ADACMP SUPDE='SP=CA(1,1),CB(1,2),CD(1,2)'
ADACMP PHONDE='PA(BA)'
/*
```

The following example can be found in member ADACMPS of the JOBS data set. This example shows the use of the ADACMP spanned record parameters and extended MU/PE limit parameters.

```
//ADACMP    JOB
//*
//*   ADACMP COMPRESS
//*      COMPRESS A FILE
//*         USING MUPEX AND SPAN OPTIONS WITH 2-BYTE MU/PE COUNTS
//*
//CMP       EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD      <=== ADABAS LOAD
//TZINFO    DD   DISP=SHR,DSN=ADABAS.Vvrs.TZ00
//*
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDEBAND   DD   DISP=OLD,DSN=EXAMPLE.DByyyyy.INPUT,UNIT=TAPE, <===
//          VOL=SER=TAPE01                                     <===
//DDAUSBA   DD   DISP=(NEW,KEEP),DSN=EXAMPLE.DByyyyy.COMP01,UNIT=DISK, <
//          VOL=SER=DISK01,SPACE=(TRK,(200,10),RLSE)
//DDFEHL    DD   DISP=(NEW,KEEP),DSN=EXAMPLE.DByyyyy.FEHL,UNIT=DISK, <==
//          VOL=SER=DISK01,SPACE=(TRK,1)
//DDCARD    DD   *
ADARUN PROG=ADACMP,MODE=MULTI,SVC=xxx,DEVICE=3390,DBID=YYYYY
/*
//DDKARTE   DD   *
ADACMP COMPRESS FILE=1
ADACMP FNDEF='01,AA,008,B,DE'
ADACMP FNDEF='01,BA,020,A,NU,DE'
ADACMP FNDEF='01,BB,015,A,NU,DE'
ADACMP FNDEF='01,BC,001,A,FI'
ADACMP FNDEF='01,CA,001,A,NU,DE'
ADACMP FNDEF='01,CB,002,U,NU,DE'
ADACMP FNDEF='01,CC,010,A,NU,DE'
ADACMP FNDEF='01,CD,002,U,NU,DE'
```

```
ADACMP FNDEF='01,DA,005,U,NU'
ADACMP FNDEF='01,DB,020,A,NU,DE'
ADACMP FNDEF='01,DC,015,A,NU,DE'
ADACMP FNDEF='01,DD,002,A,NU,DE'
ADACMP FNDEF='01,DE,005,U,NU,DE'
ADACMP FNDEF='01,DF,008,A,NU,DE'
ADACMP FNDEF='01,FA,020,A,NU,DE'
ADACMP FNDEF='01,FB,006,U,NU,DE'
ADACMP FNDEF='01,FC,006,U,NU'
ADACMP FNDEF='01,GA,002,U,NU'
ADACMP FNDEF='01,HA,002,U,NU'
ADACMP FNDEF='01,IA,002,U,NU'
ADACMP FNDEF='01,KA,002,U,NU'
ADACMP FNDEF='01,LA,030,A,NU,DE'
ADACMP FNDEF='01,MA,010,A,MU,NU,DE'
ADACMP FNDEF='01,PB,PE'
ADACMP FNDEF='02,P1,008,A,NU'
ADACMP FNDEF='02,P2,002,A,NU'
ADACMP FNDEF='02,P3,020,A,NU'
ADACMP SUBDE='SB=DE(3,5)'
ADACMP SUPDE='SP=CA(1,1),CB(1,2),CD(1,2)'
ADACMP PHONDE='PA(BA)'
ADACMP MUPEX                        <== EXTENDED MU/PE FILE
ADACMP MUPECOUNT=2                  <== 2-BYTE MU/PE COUNTS IN INPUT
ADACMP SPAN                         <== SPANNED RECORD FILE
ADACMP DATADEVICE=3390              <== DATA STORAGE DEVICE TYPE
/*
//   ↩
```

ADACMP DECOMPRESS

The following example can be found in member ADACMPD of the JOBS data set.

```
//ADACMP    JOB
//*
//*   ADACMP COMPRESS
//*      DECOMPRESS A FILE
//*
//DECMP     EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD      <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <===DATA
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <===WORK
//TZINFO    DD   DISP=SHR,DSN=ADABAS.Vvrs.TZOO
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDEBAND   DD   DISP=OLD,DSN=EXAMPLE.DByyyyy.COMPO1,UNIT=TAPE,
//          VOL=SER=TAPE01
//DDAUSBA   DD   DISP=(NEW,KEEP),DSN=EXAMPLE.DByyyyy.DECOMPO1,UNIT=DISK,
```

```
//          VOL=SER=DISK01,SPACE=(TRK,(200,10),RLSE)
//DDFEHL    DD   DISP=(NEW,KEEP),DSN=EXAMPLE.DByyyyy.FEHL,UNIT=DISK,
//          VOL=SER=DISK01,SPACE=(TRK,1)
//DDCARD    DD   *
ADARUN PROG=ADACMP,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADACMP DECOMPRESS INFILE=1
/*
```

Using ADACMP with UES Parameters, Wide Field Formats, or Collation Descriptions

The following compression example can be found in member ADACMPU of the JOBS data set. It can be used as a basis for compression jobs that make use of UES parameters, wide-character field formats, or collation descriptors.

```
//ADACMPU   JOB
//*
//*   ADACMP COMPRESS
//*      COMPRESS A FILE
//*         USING UES FEATURES
//*
//CMP       EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD  <=== ADABAS LOAD
//          DD   DISP=SHR,DSN=SAG.ICSvrs.LOnn     <=== Code pages
//*
//TZINFO    DD   DISP=SHR,DSN=ADABAS.ADAvrs.TZOO  <=== TIMEZONE DATA
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDEBAND   DD   DISP=OLD,DSN=EXAMPLE.DBYYYYY.INPUT,UNIT=TAPE, <===
//          VOL=SER=TAPE01                                     <===
//DDAUSBA   DD   DSN=EXAMPLE.DBYYYYY.COMP01,UNIT=DISK,         <===
//          VOL=SER=DISK01,SPACE=(TRK,(200,10),RLSE),DISP=(NEW,KEEP)
//DDFEHL    DD   DISP=(NEW,KEEP),DSN=EXAMPLE.DBYYYYY.FEHL,     <===
//          VOL=SER=DISK01,SPACE=(TRK,1),UNIT=DISK
//DDCARD    DD   *
ADARUN PROG=ADACMP,MODE=MULTI,SVC=XXX,DEVICE=3390,DBID=YYYYY
/*
//DDKARTE   DD   *
ADACMP COMPRESS FILE=1
ADACMP FACODE=273
ADACMP FNDEF='01,AA,008,B,DE'
ADACMP FNDEF='01,BA,020,A,NU,DE'
ADACMP FNDEF='01,BB,015,A,NU,DE'
ADACMP FNDEF='01,BC,001,A,FI'
ADACMP FNDEF='01,CA,001,A,NU,DE'
ADACMP FNDEF='01,CB,002,U,NU,DE'
ADACMP FNDEF='01,CC,010,A,NU,DE'
ADACMP FNDEF='01,CD,002,U,NU,DE'
ADACMP FNDEF='01,DA,005,U,NU'
```

```
ADACMP FNDEF='01,DB,020,A,NU,DE'
ADACMP FNDEF='01,DC,015,A,NU,DE'
ADACMP FNDEF='01,DD,002,A,NU,DE'
ADACMP FNDEF='01,DE,005,U,NU,DE'
ADACMP FNDEF='01,DF,008,A,NU,DE'
ADACMP FNDEF='01,FA,020,A,NU,DE'
ADACMP FNDEF='01,FB,006,U,NU,DE'
ADACMP FNDEF='01,FC,006,U,NU'
ADACMP FNDEF='01,GA,002,U,NU'
ADACMP FNDEF='01,HA,002,U,NU'
ADACMP FNDEF='01,IA,002,U,NU'
ADACMP FNDEF='01,KA,002,U,NU'
ADACMP FNDEF='01,LA,030,A,NU,DE'
ADACMP SUBDE='SB=DE(3,5)'
ADACMP SUPDE='SP=CA(1,1),CB(1,2),CD(1,2)'
ADACMP PHONDE='PA(BA)'
/*
//
```

41 COMPRESS Function Output

- Compressed Data Records 284
- Rejected Data Records 284
- ADACMP Report 290

This chapter describes the ADACMP COMPRESS function output.

Compressed Data Records

The data records that ADACMP has processed, edited, and compressed are written out together with the file definition information to a sequential data set with the variable blocked record format. This data set may be used as input to the ADALOD utility. The output of several ADACMP executions may also be used as input to ADALOD.

If the output data set contains no records (no records provided on the input data set or all records rejected), the output may still be used as input to the ADALOD utility. In this case, you must ensure that the amount of Associator space allocated to the file is sufficient since an accurate estimate cannot be made by the ADALOD utility without a representative sample of input record values (see the **ADALOD utility** for additional information).

For information on how to identify MU and PE occurrences greater than 191 in the compressed record, read ***Identifying MU and PE Occurrences Greater Than 191 in Compressed Records***, elsewhere in this section.

Rejected Data Records

Any records rejected during ADACMP compression are written to the DDFEHL error data set. The records are output in variable blocked format and may be segmented into multiple physical records. Each logical rejected record will be preceded by an initial ADAF rejected record header. If the logical record and the ADAF header do not fit in the DDFEHL physical record length, ADAN rejected record headers will precede the remaining physical rejected record segments that comprise the logical rejected record.

The functions of these two different headers are as follows:

- ADAF headers indicate the error condition and pertinent information.
- ADAN headers are smaller and are used for rejected record continuation and ADAH/ADAC header error reporting.

DSECTs for the ADAF and ADAN headers can be found in members ADAF and ADAN of the distributed Adabas SRCE data set.

Traditionally, the DDFEHL error data set produced for ADACMP errors has truncated rejected records that exceeded the FEHL physical record length. In Version 8, the rejected records are segmented instead of truncated. Because of this change, the DDFEHL LRECL setting must be at least 500 bytes.

If HEADER=YES is specified, an error may occur while segmented uncompressed records are being assembled into a logical record. If the ADAH header is in error, the ADAH record is written and subsequent ADAC records are not written until the next ADAH record is processed. If an ADAC header is in error, the preceding ADAH header will be written without its payload data. The ADAC record in error will be written in its entirety. Subsequent ADAC records are not written until the next ADAH record is processed.

The following response codes may occur:

X'E7'(231 - ADARSP231)	Input record too short (COMPRESS)
X'E8'(232 - ADARSP232)	Output record length error (COMPRESS)
X'E9'(233 - ADARSP233)	An invalid ADAH spanned record header has been encountered. The following subcodes provide more detail: ■ 1 - incorrect ADAH eye-catcher ■ 2 - incorrect ADAH header length ■ 3 - unexpected continuation indicator ■ 4 - reserved area not set to binary zeros ■ 5 - invalid segment length ■ 6 - total payload data length exceeds MAXLOGRECLEN setting For complete information about spanned records and the ADAH header, read *Spanned Records* in *Adabas Concepts Manual*. The DSECT for the ADAH header can be found in member ADAH of the distributed Adabas SRCE data set.
X'EA'(234 - ADARSP234)	An invalid ADAC spanned record header has been encountered. The following subcodes provide more detail: ■ 1 - incorrect ADAC eye-catcher ■ 2 - incorrect ADAC header length ■ 3 - unexpected continuation indicator ■ 4 - reserved area not set to binary zeros ■ 5 - invalid segment length ■ 6 - unexpected continuation record sequence number ■ 7 - invalid segment offset ■ 8 - accumulated payload data length exceeds specified total length in ADAH ■ 9 - accumulated payload data length exceeds MAXLOGRECLEN setting For complete information about spanned records and the ADAC header, read *Spanned Records* in *Adabas Concepts Manual*. The DSECT for the ADAC header can be found in member ADAC of the distributed Adabas SRCE data set.

Only the first incorrect field within a record is detected and referenced. If there are other errors, they are not detected until subsequent runs are made.

Example of Rejected Data Records

The following table depicts the FEHL output for four rejected records during ADACMP compression. Rejected records 1 and 3 have only one FEHL record (ADAF); rejected record 2 is segmented into two FEHL records (ADAF and ADAN); rejected record 4 is segmented into three FEHL records (one ADAF record and two ADAN records):



Note: DSECTs for the ADAF and ADAN headers can be found in members ADAF and ADAN of the distributed Adabas SRCE data set.

Rejected Record	FEHL Records	FEHL Fields		Description
		Field	Value	
1	ADAF	ADAFEYE	ADAF	ADAF header eye-catcher
		ADAFLEN	72	ADAF header length
		ADAFTYPE	R	Type. Valid values are: H: ADAH header P: ADAH header and payload R: Logical record
		ADAFIND	E	Continuation indicator. Valid values are: C: Continuation record to follow E: End of logical record (last segment)
		Reserved	0	Reserved
		ADAFSLEN	22000	Segment length (length of payload data following the header)
		ADAFTOTL	22000	Total length. **Note:** The sum of the values of all ADANSLEN fields and ADAFSLEN should equal the value of ADAFTOTL (for record 1 in this example, 0 + 22000 = 22000)
		ADAFISN	1	ISN of record
		ADAFLNUM	1	Logical record number
		ADAFPNUM	1	Physical record number
		ADAFEOFF	5000	Error offset in logical record
		ADAFPEX	0	PE index
		ADAFFN	ZA	Field name
		ADAFRSP	41	Response code
		ADAFSUB	2	Subcode

Rejected Record	FEHL Records	FEHL Fields		Description
		Field	Value	
		Reserved	0	Reserved
		ADAFDATA	'Record 1 -- Payload Data'	Rejected input data
2	ADAF	ADAFEYE	ADAF	ADAF header eye-catcher
		ADAFLEN	72	ADAF header length
		ADAFTYPE	R	Type. Valid values are: H: ADAH header P: ADAH header and payload R: Logical record
		ADAFIND	C	Continuation indicator. Valid values are: C: Continuation record (ADAN) to follow E: End of logical record (last segment)
		Reserved	0	Reserved
		ADAFSLEN	27962	Segment length (length of payload data following the header)
		ADAFTOTL	50000	Total length. **Note:** The sum of the values of all ADANSLEN fields and ADAFSLEN should equal the value of ADAFTOTL (for record 2 in this example, 22038 + 27962 = 50000)
		ADAFISN	2	ISN of record
		ADAFLNUM	2	Logical record number
		ADAFPNUM	3	Physical record number
		ADAFEOFF	35000	Error offset in logical record
		ADAFPEX	0	PE index
		ADAFFN	ZA	Field name
		ADAFRSP	41	Response code
		ADAFSUB	2	Subcode
		Reserved	0	Reserved
		ADAFDATA	'Record 2 -- Payload data part 1'	Rejected input data
	ADAN	ADANEYE	ADAN	ADAN header eye-catcher
		ADANLEN	24	ADAN header length
		ADANTYPE	R	Type. Valid values are: C: ADAC header D: ADAC header and payload

Rejected Record	FEHL Records	FEHL Fields		Description
		Field	Value	
				P: ADAH record segment R: Logical record segment
		ADANIND	E	Continuation indicator. Valid values are: C: Continuation record (ADAN) to follow E: End of logical record (last segment)
		Reserved	0	Reserved
		ADANSLEN	22038	Segment length (length of payload data following the header)
		ADANOFF	27962	Error offset in logical record.
		ADANDATA	'Record 2 -- Payload data part 2'	Continued rejected input data
3	ADAF	ADAFEYE	ADAF	ADAF header eye-catcher
		ADAFLEN	72	ADAF header length
		ADAFTYPE	P	Type. Valid values are: H: ADAH header P: ADAH header and payload R: Logical record
		ADAFIND	E	Continuation indicator. Valid values are: C: Continuation record (ADAN) to follow E: End of logical record (last segment)
		Reserved	0	Reserved
		ADAFSLEN	20000	Segment length (length of payload data following the header)
		ADAFTOTL	20000	Total length. **Note:** The sum of the values of all ADANSLEN fields and ADAFSLEN should equal the value of ADAFTOTL (for record 3 in this example, 0 + 20000 = 20000)
		ADAFISN	3	ISN of record
		ADAFLNUM	3	Logical record number
		ADAFPNUM	4	Physical record number
		ADAFEOFF	0	Error offset in logical record
		ADAFPEX	0	PE index
		ADAFFN		Field name
		ADAFRSP	233	Response Code
		ADAFSUB	1	Subcode

Rejected Record	FEHL Records	FEHL Fields		Description
		Field	Value	
		Reserved	0	Reserved
		ADAFDATA	ADAH and payload data	Rejected input data
4	ADAF	ADAFEYE	ADAF	ADAF header eye-catcher
		ADAFLEN	72	ADAF header length
		ADAFTYPE	H	Type. Valid values are: H: ADAH header P: ADAH header and payload R: Logical record
		ADAFIND	C	Continuation indicator. Valid values are: C: Continuation record (ADAN) to follow E: End of logical record (last segment)
		Reserved	0	Reserved
		ADAFSLEN	32	Segment length (length of payload data following the header)
		ADAFTOTL	10064	Total length. **Note:** The sum of the values of all ADANSLEN fields and ADAFSLEN should equal the value of ADAFTOTL (for record 4 in this example, 32 + 32 + 10000 = 10064)
		ADAFISN	4	ISN of record
		ADAFLNUM	4	Logical record number
		ADAFPNUM	8	Physical record number
		ADAFEOFF	0	Error offset in logical record
		ADAFPEX	0	PE index
		ADAFFN		Field name
		ADAFRSP	234	Response Code
		ADAFSUB	3	Subcode
		Reserved	0	Reserved
		ADAFDATA	ADAH header	Rejected input data
	ADAN	ADANEYE	ADAN	ADAN header eye-catcher
		ADANLEN	24	ADAN header length
		ADANTYPE	C	Type. Valid values are: C: ADAC header D: ADAC header and payload P: ADAH record segment R: Logical record segment

Rejected Record	FEHL Records	FEHL Fields		Description
		Field	Value	
		ADANIND	C	Continuation indicator. Valid values are: C: Continuation record (ADAN) to follow E: End of logical record (last segment)
		Reserved	0	Reserved
		ADANSLEN	32	Segment length (length of payload data following the header)
		ADANOFF	32	Error offset in logical record.
		ADANDATA	ADAC header	Continued rejected input data
	ADAN	ADANEYE	ADAN	ADAN header eye-catcher
		ADANLEN	24	ADAN header length
		ADANTYPE	D	Type. Valid values are: C: ADAC header D: ADAC header and payload P: ADAH record segment R: Logical record segment
		ADANIND	E	Continuation indicator. Valid values are: C: Continuation record (ADAN) to follow E: End of logical record (last segment)
		Reserved	0	Reserved
		ADANSLEN	10000	Segment length (length of payload data following the header)
		ADANOFF	64	Error offset in logical record.
		ADANDATA	ADAC and payload data	Continued rejected input data

ADACMP Report

ADACMP calculates the approximate amount of space (in both blocks and cylinders) required for Data Storage for the compressed records. This information is printed as a matrix which contains the required space for the different device types requested by the DEVICE parameter for various Data Storage padding factors between 5 and 30 percent.

The following is an example of ADACMP report output:

```
PARAMETERS:

ADACMP COMPRESS NUMREC=1000
ADACMP FNDEF='01,AA,8,B,DE'
ADACMP FNDEF='01,BA,6,A,NU'
ADACMP FNDEF='01,BB,8,P,NU'
ADACMP FNDEF='01,AD,1,A,FI'
ADACMP SUBDE='CA=BA(1,3)'

COMPRESS PROCESSING STATISTICS:

NUMBER OF RECORDS READ                1,000
NUMBER OF INCORRECT RECORDS               0
NUMBER OF COMPRESSED RECORDS          1,000

RAW DATA                             24,000 BYTES
COMPRESSED DATA                      16,656 BYTES
COMPRESSION RATE                       31.9 %
LARGEST COMPRESSED RECORD                 20 BYTES


DATASTORAGE SPACE REQUIREMENTS:

I  DEVICE  I  PADDING I  BLOCKSIZE I          NUMBER OF     I
I          I  FACTOR  I    BYTES   I      BLOCKS      CYLS  I
I----------I----------I------------I-------------------------I
I   3380   I          I    4,820   I                         I
I          I     5%   I    4,578   I           4         1   I
I          I    10%   I    4,337   I           4         1   I
I          I    15%   I    4,096   I           5         1   I
I          I    20%   I    3,856   I           5         1   I
I          I    25%   I    3,615   I           5         1   I
I          I    30%   I    3,373   I           5         1   I
I          I          I            I                         I
I----------I----------I------------I-------------------------I

TEMP SPACE ESTIMATION:

I  DEVICE  I  BLOCKSIZE I          NUMBER OF     I
I          I    BYTES   I      BLOCKS      CYLS  I
I----------I------------I-------------------------I
I   3380   I    7,476   I           5          1 I
I----------I------------I-------------------------I

THE LARGEST DESCRIPTOR IS AA, IT WILL OCCUPY 1 TEMP BLOCKS


SORT SPACE ESTIMATION:
```

```
I DEVICE I BLOCKSIZE I   LWP              I      NR OF      I
I        I  (BYTES)  I  (BYTES)           I  BLOCKS    CYLS I
-------------------------------------------------------------
I  3380  I     7476  I   139264 (MINIMUM) I       2       1 I
I        I           I  1048576 (DEFAULT) I       2       1 I
I        I           I   139264 (OPTIMUM) I       2       1 I
I--------I-----------I--------------------I-----------------I
```

The compression rate is computed based on the real amount of data used as input to the compression routine. Fields skipped by a format element "nX" (used to fill a field with blanks) are not counted.

If SPAN was specified in an ADACMP COMPRESS run, statistics about the spanned Data Storage records are also printed:

```
Spanned Record Statistics:
--------------------------

Number of Non-Spanned records              2
Number of Spanned records                  8
Min Number of Segments             2
Max Number of Segments             5
Avg Number of Segments             4

Max MU Count                       0
Max PE Count                     300         ↩
```

If large object (LB) fields are compressed, statistics about the LB fields (listed as "LOBs" in the report) are printed, as shown below. In addition, if the LOBDEVICE parameter is specified, ADACMP calculates and displays a report of the space requirements for each specified device type. Since the LOB fields are split into segments with the maximum possible size for the first device specified in LOBDEVICE, you should not use the output for loading with any other device specified in LOBDEVICE. Here is an example of the output produced for LOB files when the LOBDEVICE parameter is specified:

```
Large Objects Statistics:
-------------------------

Number of bytes for largest LOB             567,041
Total number of LOBs                          2,572
Total number of outsourced LOBs                  60
Number of LOB file records                      571


DATASTORAGE Space requirements for Base File
--------------------------------------------

I  Device  I  Padding I  Blocksize I            Number of         I
I          I  Factor  I   (bytes)  I         Blocks         Cyls   I
```

```
I----------I----------I------------I--------------------------------I
I   8391   I          I   10,796   I                                I
I          I     5%   I   10,255   I             31            1   I
I          I    10%   I    9,715   I             33            1   I
I          I    15%   I    9,175   I             35            1   I
I          I    20%   I    8,637   I             37            1   I
I          I    25%   I    8,097   I             39            1   I
I          I    30%   I    7,556   I             41            1   I
I          I          I            I                                I
I----------I----------I------------I--------------------------------I


DATASTORAGE Space requirements for LOB File
-------------------------------------------


I  Device  I   Blocksize I             Number of        I
I          I    (bytes)  I         Blocks          Cyls I
I----------I-------------I------------------------------I
I   8391   I    10,796   I            571             8 I
I   3390   I     5,064   I          1,185             8 I
I   8393   I    27,644   I            241             9 I
I----------I-------------I------------------------------I

Note: LOB fields were split into segments with the maximum possible size for
      the first LOBDEVICE. Do not use the output for loading with the
      other LOBDEVICEs.


TEMP Space Estimation
---------------------


I  Device  I   Blocksize I             Number of        I
I          I    (bytes)  I         Blocks          Cyls I
I----------I-------------I------------------------------I
I   8391   I    18,452   I             22             1 I
I----------I-------------I------------------------------I
```

42 DECOMPRESS Function Output

- Rejected Data Records 296

The ADACMP DECOMPRESS function decompresses each record and then stores the record in a sequential data set. The records are output in variable-length, blocked format. Each decompressed record is output either with or without the ISN option according to the format shown below:

```
length xx [ISN] data
```

where

length	is a two-byte binary length of the data, + 8 (or +4 if the ISN parameter is not specified).
xx	is a two-byte field containing binary zeros.
ISN	is a four-byte binary ISN of the record.
data	is a decompressed data record.

The fields of the data record are provided in the order in which they appeared in the FDT when the file was unloaded. The standard length and format are in effect for each field.

If a field value exceeds the standard length, the value will be truncated to the standard length if the field is alphanumeric and the TRUNCATE parameter was specified; otherwise, ADACMP writes the record to the DDFEHL error data set (read the next chapter, ***Rejected Data Records***).

Any count bytes for multiple-value fields or periodic groups contained in the record are included in the decompressed data output. ADACMP generates a count of 1 if the MU field or PE group is empty. This makes it possible to use the output of the DECOMPRESS operation as the input to a subsequent COMPRESS operation.

Rejected Data Records

ADACMP rejects a record whenever a compressed field's size is greater than the default length held in the FDT, unless the TRUNCATE parameter is specified.

Any records rejected during ADACMP decompression are written to the DDFEHL error data set. The records are output in variable blocked format and may be segmented into multiple physical records. Each logical rejected record will be preceded by an initial ADAF rejected record header. If the logical record and the ADAF header do not fit in the DDFEHL physical record length, ADAN rejected record headers will precede the remaining physical rejected record segments that comprise the logical rejected record.

The functions of these two different headers are as follows:

- ADAF headers indicate the error condition and pertinent information.
- ADAN headers are smaller and are used for rejected record continuation and ADAH/ADAC header error reporting.

DSECTs for the ADAF and ADAN headers can be found in members ADAF and ADAN of the distributed Adabas SRCE data set.

Traditionally, the DDFEHL error data set produced for ADACMP errors has truncated rejected records that exceeded the FEHL physical record length. In Version 8, the rejected records are segmented instead of truncated. Because of this change, the DDFEHL LRECL setting must be at least 500 bytes.

The following response codes may occur:

X'E7'(231 - ADARSP231)	Input record too short (DECOMPRESS)
X'E8'(232 - ADARSP232)	Output record length error (DECOMPRESS). If the output records exceed the LRECL size, use HEADER=YES to segment the output records.



Notes:

1. Only the first incorrect field within a record is detected and referenced in DDFEHL. Other errors within the record are not detected or recorded.

Example of Rejected Data Records

The following table depicts the FEHL output for two rejected records during ADACMP decompression. Rejected record 1 has only one FEHL record (ADAF); rejected record 2 is segmented into two FEHL records (ADAF and ADAN):



Note: DSECTs for the ADAF and ADAN headers can be found in members ADAF and ADAN of the distributed Adabas SRCE data set.

Rejected Record	DDFEHL Records	DDFEHL Fields		Description
		Field	Value	
1	ADAF	ADAFEYE	ADAF	ADAF header eye-catcher
		ADAFLEN	72	ADAF header length
		ADAFTYPE	R	Type. Valid values are: H: ADAH header P: ADAH header and payload R: Logical record
		ADAFIND	E	Continuation indicator. Valid values are: C: Continuation record to follow E: End of logical record (last segment)
		Reserved	0	Reserved
		ADAFSLEN	22000	Segment length
		ADAFTOTL	22000	Total length

Rejected Record	DDFEHL Records	DDFEHL Fields		Description
		Field	Value	
		ADAFISN	1	ISN of record
		ADAFLNUM	1	Logical record number
		ADAFPNUM	1	Physical record number
		ADAFEOFF	5000	Error offset in logical record
		ADAFPEX	0	PE index
		ADAFFN	ZA	Field name
		ADAFRSP	41	Response code
		ADAFSUB	2	Subcode
		Reserved	0	Reserved
		ADAFDATA	'Record 1 -- Payload Data'	Rejected input data
2	ADAF	ADAFEYE	ADAF	ADAF header eye-catcher
		ADAFLEN	72	ADAF header length
		ADAFTYPE	R	Type. Valid values are: H: ADAH header P: ADAH header and payload R: Logical record
		ADAFIND	C	Continuation indicator. Valid values are: C: Continuation record (ADAN) to follow E: End of logical record (last segment)
		Reserved	0	Reserved
		ADAFSLEN	27962	Segment length
		ADAFTOTL	50000	Total length
		ADAFISN	2	ISN of record
		ADAFLNUM	2	Logical record number
		ADAFPNUM	3	Physical record number
		ADAFEOFF	35000	Error offset in logical record
		ADAFPEX	0	PE index
		ADAFFN	ZA	Field name
		ADAFRSP	41	Response code
		ADAFSUB	2	Subcode
		Reserved	0	Reserved
		ADAFDATA	'Record 2 -- Payload data part 1'	Rejected input data
	ADAN	ADANEYE	ADAN	ADAN header eye-catcher
		ADANLEN	24	ADAN header length

Rejected Record	DDFEHL Records	DDFEHL Fields		Description
		Field	Value	
		ADANTYPE	R	Type. Valid values are: C: ADAC header D: ADAC header and payload P: ADAH record segment R: Logical record segment
		ADANIND	E	Continuation indicator. Valid values are: C: Continuation record (ADAN) to follow E: End of logical record (last segment)
		Reserved	0	Reserved
		ADANSLEN	22038	Segment length. **Note:** The sum of the values of ADANSLEN and ADAFSLEN should equal the value of ADAFTOTL (in this example, 22038 +27962=50000)
		ADANOFF	27962	Error offset in logical record.
		ADANDATA	'Record 2 -- Payload data part 2'	Continued rejected input data

V ADACNV Utility: Database Conversion

This chapter covers the following topics:

Functional Overview
CONVERT: Convert Database to Higher Version
REVERT: Revert Database to Lower Version
JCL Requirements and Examples

43 Functional Overview

- Database Status 304
- Procedure 305

The ADACNV utility can convert (CONVERT) an Adabas mainframe database from version 6.1 or above to a higher version or revert (REVERT) an Adabas database from a higher version back to a lower version.

Caution: Before you convert a database, you must terminate all active nucleus or utility jobs normally.

In general, you will need to run ADACNV on your database whenever the Adabas version or release number (*v*.*r*) has changed, but not for modification (SM) levels. For example, ADACNV should be run to convert your database from 8.3 to 8.4 and from 8.5 to 8.6. However, there is usually no need to run ADACNV to convert your database between 8.6 releases.

To ensure database integrity, ADACNV writes changed blocks first to intermediate storage; that is, to the sequential data set DDFILEA. After all changed blocks have been written out to DDFILEA, a *point-of-no-return* is reached and the changed blocks are written to the database. If ADACNV terminates abnormally after the *point-of-no-return*, the RESTART parameter can be used to begin the ADACNV run by reading the contents of DDFILEA and writing them out to the database.

The TEST parameter is provided to check the feasibility of a conversion or reversion without writing any changes to the database. It is therefore not necessary to terminate all activity on the database before running ADACNV when you use the TEST parameter.

Database Status

Internally, the utility converts or reverts one version at a time until the target version is attained. It is therefore important to ensure that all requirements for conversion or reversion between the current and target database levels have been met before you execute ADACNV without the TEST parameter.

Before a conversion or reversion begins, ADACNV checks the status of the database:

- The DIB must be empty; that is, no Adabas nucleus or utility may be active or have been terminated abnormally. If RESTART is specified, the DIB must contain the entry of ADACNV, which includes a time stamp.
- The Work data set must not have a pending autorestart.

If this check is successful, ADACNV locks the database and creates a DIB entry.

For reversions, ADACNV checks whether any features are used that do not exist in the target version and returns a message if any are found.

Note: There may be files in a database that are not loaded but that have a Field Definition Table (FDT) stored in the FDT blocks. If ADACNV encounters such FDTs while converting a database to Version 8, they are deleted as part of its cleanup processing.

Procedure

The procedure for converting or reverting an Adabas database is as follows:

1. If the nucleus is active, use ADAEND to stop it.
2. Use ADARES PLCOPY/CLCOPY to copy all protection and command logs.

   For your installation, this may be done automatically with user exit 2 or user exit 12.

   Wait until the logs have been copied.
3. Optionally, back up the database (full or delta).
4. Execute the ADACNV utility.
5. Start the nucleus of the version to which you have converted or reverted.



Important: To ensure database integrity, DDFILEA must be defined permanently and be deleted only after ADACNV has completed successfully. The DDFILEA data set must not be defined as a temporary data set that is automatically deleted at the end of the job.

44 CONVERT: Convert Database to Higher Version

- Optional Parameters 308
- Conversion Considerations 309
- Example 310

The CONVERT function starts from the Adabas version of the last nucleus session.

```
ADACNV  CONVERT  [IGNPPT]
                 [NOUSERABEND]
                 [PLOGDEV = multiple-PLOG-device-type ]
                 [RESTART]
                 [TEST]
                 [TOVERS = target-version ]
```

Optional Parameters

IGNPPT: Ignore Parallel Participant Table PLOG Entries

When converting from a version of Adabas that uses the parallel participant table (PPT) structure to a higher version of Adabas, an error is printed and conversion fails if the system detects one or more protection logs (PLOGs) from the current version that have not been copied/merged.

Specify IGNPPT to continue processing in spite of the uncopied/unmerged PLOGs.

Note: If DDPLOG*x* statements have been specified in the JCL in addition to the DDPLOG*x* data sets in the PPT, the specified data sets must be empty or the error will still be received. IGNPPT only pertains to the PPT processing. If PLOG data sets are supplied in the JCL, they must be empty.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PLOGDEV: Multiple PLOG Device Type

PLOGDEV specifies the physical device type on which the multiple protection log data sets to be converted are contained. If PLOGDEV is not specified, the device type specified by the ADARUN DEVICE parameter is used.

RESTART: Rerun after Point of No Return

If ADACNV terminates abnormally after the *point-of-no-return*, that is, after all changed blocks have been written to DDFILEA, the RESTART parameter instructs ADACNV to begin its run by reading the contents of DDFILEA and continue by writing them to the database.

TEST: Test Conversion

The TEST parameter tests the feasibility of the conversion operation without actually writing any changes to the database.

TOVERS: Target Version

The two-character version of Adabas database (version and revision level) to achieve at the end of the ADACNV run. If the TOVERS parameter is

- specified, it must be a version higher than the source version.
- not specified, ADACNV uses its own version as the target version.

The version format is *vr* indicating the version and release level; for example, "85" or "86".



Note: There may be files in a database that are not loaded but that have a Field Definition Table (FDT) stored in the FDT blocks. If ADACNV encounters such FDTs while converting a database to Version 8, they are deleted as part of its cleanup processing.

Conversion Considerations

In general, you will need to run ADACNV on your database whenever the Adabas version or release number (*v*.*r*) has changed, but not for modification (SM) levels. For example, ADACNV should be run to convert your database from 8.4 to 8.5 and from 8.5 to 8.6. However, there is usually no need to run ADACNV to convert your database between 8.6 releases.

The following is an overview of special conversion considerations for ADACNV.

All Versions

The data protection area on the Work data set and the multiple PLOG data sets (if supplied) are cleared to binary zeros.

From Version 6.1 to 6.2

Any Adabas Delta Save Facility DLOG area header is set to the correct version.

From Version 6.2 to 7.1

Any Adabas Delta Save Facility DLOG area header is set to the correct version.

Any Version to Version 8

There may be files in a database that are not loaded but that have a Field Definition Table (FDT) stored in the FDT blocks. If ADACNV encounters such FDTs while converting a database to Version 8, they are deleted as part of its cleanup processing.

Example

The following example indicates that the database should be converted to the version of Adabas of which the ADACNV utility is part.

```
ADACNV CONVERT
```

45 REVERT: Revert Database to Lower Version

- Essential Parameter and Subparameter 312
- Optional Parameter 312
- Reversion Considerations 313
- Example 315

The REVERT function starts from the Adabas version of the last nucleus session.

```
ADACNV  REVERT  [TOVERS = target-version ]
                [IGNPPT]
                [NOUSERABEND]
                [PLOGDEV = multiple-PLOG-device-type ]
                [RESTART]
                [TEST]
```

Essential Parameter and Subparameter

TOVERS: Target Version

The version of Adabas database (version and revision level) to achieve at the end of the ADACNV run. The TOVERS parameter value must be a version lower than the source version.

The version format is *vr* indicating the version and release level; for example, "85" or "86".

Optional Parameter

IGNPPT: Ignore Parallel Participant Table PLOG Entries

When reverting from a version of Adabas that uses the parallel participant table (PPT) structure to a lower version of Adabas, an error is printed and conversion fails if the system detects one or more protection logs (PLOGs) from the current version that have not been copied/merged.

If IGNPPT is specified, the utility will continue processing in spite of the uncopied/unmerged PLOGs.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "`utility` TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PLOGDEV: Multiple PLOG Device Type

PLOGDEV specifies the physical device type on which the multiple protection log data sets to be reverted is contained. If PLOGDEV is not specified, the device type specified by the ADARUN DEVICE parameter is used.

RESTART: Rerun after Point of No Return

If ADACNV terminates abnormally after the *point-of-no-return*, that is, after all changed blocks have been written to DDFILEA, the RESTART parameter instructs ADACNV to begin its run by reading the contents of DDFILEA and continue by writing them to the database.

TEST: Test Conversion

The TEST parameter tests the feasibility of the reversion operation without actually writing any changes to the database.

Reversion Considerations

The following is a list of special reversion considerations for ADACNV.

For information about reverting back to versions prior to Adabas 8.2, refer to the *Adabas 8.2 Release Notes*. To access a copy of these, read *Documentation and Other Online Information*, in the *Adabas Release Notes*.

All Versions

Reversion is not possible if any Adabas feature is used in the current version that is not supported in the target version. This statement applies to all Adabas features that affect the structure of the database.

From Version 8.4 to a version prior to 8.3.2

ADACNV will not allow you to REVERT the database to a version prior to Adabas 8.3 if the nucleus has been run with INDEXUPDATE=ADVANCED, and files have actually been updated according to the advanced index structure rules.

If an Adabas nucleus has been run with INDEXUPDATE=ADVANCED, once files have actually been updated according to the advanced index structure rules, Adabas can no longer be directly downgraded to an earlier version. Prior to a downgrade, the indexes of those files must be rebuilt using the ***ADALOD*** or ***ADAORD*** utilities.

If the database is to be converted back to a version below 8.3, the ADACNV utility checks for files whose indexes have been updated according to the advanced index structure rules. If there is any such file, ADACNV declines to convert the database and terminates with ERROR-147:

```
File (fnr) has the advanced index structure.
```

The *ADAREP* utility can be used to list all files that have the advanced index structure. In the detail section of the file report, ADAREP prints a line saying

```
'Index structure ... Advanced'
```

if the file has been updated according to the advanced index structure rules. Otherwise, the line says

```
'Index structure ... Original.'
```

Utilities always build the index structure according to the original rules. The ADALOD and ADAORD utilities can be used to change indexes with the advanced structure back to the original structure. Any of the following functions can be used:

- ADAULD UNLOAD followed by ADALOD LOAD
- ADAORD REORASSO
- ADAORD REORDB
- ADAORD REORFASSO
- ADAORD REORFILE
- ADAORD RESTRUCTUREDB (REDB) followed by ADAORD STORE
- ADAORD RESTRUCTUREF (REF) followed by ADAORD STORE

Caution: Once the nucleus is run with INDEXUPDATE=ADVANCED, do not downgrade Adabas to version 8.3 SP1 if there are files in the database that have the advanced index structure. The Adabas 8.3 SP1 nucleus and utilities would recognize such indexes as inconsistent. Rather, before going back to Adabas 8.3 SP1, rebuild the indexes of those files using ADALOD or ADAORD, as described above.

From Version 8.3 to 8.2

ADACNV will not allow you to REVERT the database to Adabas 8.2 if any of the following apply:

- The database contains a Security file in which a password is or was defined that applies to more than 191 files or relates to more security-by-value criteria than fit into a single data storage record. (This condition persists even if that password is deleted.)
- The database contains a file in which a system field of type SECUID is defined.

Example

The following example indicates that the database should be converted back (reverted) to a version 8.5 Adabas database.

```
ADACNV REVERT TOVERS=85
```

46 JCL Requirements and Examples

- Requirements to Run ADACNV JCL 318

This section describes the job control information required to run ADACNV on the z/OS operating system and shows an example job stream.

Requirements to Run ADACNV JCL

Data Set	DD Name	Storage	More Information
Associator	DDASSORn	disk	
Data Storage	DDDATARn	disk	
Work	DDWORKR1 DDWORKR4	disk	
Multiple protection logs	DDPLOGRn	disk	
Intermediate storage	DDFILEA	tape/ disk	
ADARUN parameters	DDCARD	reader	*Operations*
ADACNV parameters	DDKARTE	reader	
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADACNV messages	DDDRUCK	printer	*Messages and Codes*

ADACNV JCL Example

Refer to ADACNV in the JOBS data set for this example.

```
//ADACNV    JOB
//*
//*    ADACNV:
//*       EXAMPLE HOW TO USE ADACNV TO CONVERT DATABASE
//*       TO A DIFFERENT VERSION
//*
//CNV      EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD      <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <=== WORK
//DDPLOGR1  DD   DSN=EXAMPLE.DByyyyy.PLOGR1,DISP=SHR <=== PLOG 1
//DDPLOGR2  DD   DSN=EXAMPLE.DByyyyy.PLOGR2,DISP=SHR <=== PLOG 2
//DDFILEA   DD   DSN=EXAMPLE.DByyyyy.FILEA,         <=== INTERMEDIATE FILE
//               UNIT=SYSDA,SPACE=(TRK,(150,150),RLSE),
//               DISP=(NEW,CATLG)
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
```

```
ADARUN PROG=ADACNV,SVC=xxx,DE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADACNV CONVERT TOVERS=vr
/*
```

VI ADADBS Utility: Database Services

This chapter covers the following topics:

Functional Overview
ADD: Add Data Set
ADDCLOG: Dynamically Add CLOG Data Sets
ADDPLOG: Dynamically Add PLOG Data Sets
ALLOCATE: Allocate File Extent
CHANGE: Change Standard Length or Format of a Field
CVOLSER: Print Adabas Extents on Given Volume
DEALLOCATE: Deallocate File Extent
DECREASE: Decrease Last Associator or Data Storage Data Set Size
DELCLOG: Dynamically Delete CLOG Data Sets
DELCP: Delete Checkpoint Records
DELDE: Logically Deleting a Descriptor
DELETE: Delete File
DELFN: Logically Delete Fields
DELPLOG: Dynamically Delete PLOG Data Sets
DEVENTLOG: Display Adabas Event Log
DSREUSE: Reuse Data Storage Blocks
ENCODEF: Change File Encoding
EXPFILE: Insert or Remove Files in an Expanded File Chain
INCREASE: Increase Last Associator or Data Storage Data Set Size
ISNREUSE: Reuse ISNs
MODFCB: Modify File Parameters
MUPEX: Set Maximum Count for MU and PE Fields
NEWFIELD: Add New Field
ONLADD: Allocating New ASSO or DATA Space Dynamically

ONLINCREASE: Allocating Appended ASSO or DATA Space Dynamically
ONLINVERT: Start Online Invert Process
ONLREORFASSO: Start Online Reorder Associator for Files
ONLREORFDATA: Start Online Reorder Data for Files
ONLREORFILE: Start Online Reorder Associator and Data for Files
OPERCOM: Issue Adabas Operator Commands
PRIORITY: Change User Priority
REACTLOG: Reactivate Command Logging
RECORDSPANNING: Enable or Disable Record Spanning
RECOVER: Recover Space
REFRESH: Set File to Empty Status
REFRESHSTATS: Reset Statistical Values
RELEASE: Release Descriptor
RENAME: Rename a File or Database
RENUMBER: Change File Number
REPLICATION: Activating or Deactivating Replication
REPTOR: Activate, Deactivate, Open, or Close Event Replicator Resources
RESETDIB: Reset Entries in Active Utility List
RESETPPT: Reset PPT Blocks
SPANCOUNT: Count Spanned Records
TRANSACTIONS: Suspend and Resume Update Transaction Processing
UNCOUPLE: Uncouple Files
UNDELDE: Undeleting a Logically Deleted Descriptor
UNDELFN: Logically Undelete Fields
JCL/JCS Requirements and Examples

47 Functional Overview

- Syntax Checking with the TEST Parameter 324

Note: All ADADBS functions can also be performed using Adabas Online System (AOS). When using the Adabas Recovery Aid, using AOS is preferable for file change operations because it writes checkpoints that are necessary for recovery operation.

Any number of functions may be performed during a single execution of ADADBS.

Syntax Checking with the TEST Parameter

The ADADBS functions now include a syntax-checking-only mode. When the TEST parameter is specified, the ADADBS function syntax is checked, but not performed.

The ADADBS utility can perform multiple functions. As a result, ADADBS reads the parameters up to the next specified ADADBS function, and then executes the function/parameters just read. Then, ADADBS reads the function and parameters up to the following function, and so on. Therefore, to ensure that no functions are executed, the TEST parameter must be specified either before or within the first function/parameter group, as the following example shows:

```
ADADBS TEST
ADADBS DELETE FILE=1
ADADBS DELETE FILE=2
```

48 ADD: Add Data Set

- Associator or Data Storage Data Set 326
- Essential Parameter and Subparameter 327
- Optional Parameters 327
- Examples 327

The ADD function adds a new data set to the Associator or Data Storage.

```
ADADBS  ADD  { ASSOSIZE = size [ ASSODEV = device-type ] |
               DATASIZE = size [ DATADEV = device-type ] }
             [NOUSERABEND]
             [TEST]
```

This chapter describes the syntax, processing, and parameters of the ADADBS ADD function.

Associator or Data Storage Data Set

For the Associator or for Data Storage, the data set to be added may be on the same device type as that currently being used or on a different one. A maximum of 99 physical extents is now set for Associator and Data Storage data sets. However, your actual real maximum could be less because the extent descriptions of all Associator, Data Storage, and Data Storage Space Table (DSST) extents must fit into the general control blocks (GCBs). For example, on a standard 3390 device type, there could be more than 75 Associator, Data Storage, and DSST extents each (or there could be more of one extent type if there are less for another).

Note: The Associator and Data Storage data set sizes must be added separately. It is *not* possible to add both with a single operation.

After an ADD operation is completed for an Associator or Data Storage data set, the ADD function automatically ends the current nucleus session. A message informs you when the nucleus has been stopped. Assuming the added data set has already been formatted and the JCL has been updated for all nucleus startup and utility procedures, the nucleus can then be restarted. Once it is restarted, the additional free space added by the ADD operation will be available.

Procedure

To add an additional data set to the Associator or Data Storage:

1. Allocate the data set with the operating system, then format the additional space using the ADAFRM utility.
2. Add necessary JCL to all Adabas nucleus and Adabas utility execution procedures.
3. Execute the ADD function.
4. Restart the nucleus.

Essential Parameter and Subparameter

ASSODEV | DATADEV: Device Type

The device type to be used for the new data set. These parameters are required only if a different device type from the device type specified by the ADARUN DEVICE parameter is to be used.

ASSOSIZE | DATASIZE: Size of Data Set to be Added

The number of cylinders to be contained in the new data set.

Optional Parameters

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See **Syntax Checking with the TEST Parameter** for more information about using the TEST parameter in ADADBS functions.

Examples

A new data set containing 800 cylinders on Adabas device type 8391 is to be added to Data Storage.

```
ADADBS ADD DATASIZE=800,DATADEV=8391
```

A new data set containing 100 cylinders is to be added to the Associator on the Associator's existing device type.

```
ADADBS ADD ASSOSIZE=100
```

49 ADDCLOG: Dynamically Add CLOG Data Sets

- Essential Parameters 331
- Optional Parameters 331
- Examples 332

The ADDCLOG function allows you to dynamically add a new command log (CLOG) data set without terminating your current nucleus session. Using this utility function, you can specify up to eight CLOG data sets. This will reduce the chances of a wait condition in the nucleus, when the nucleus waits for an available CLOG. You might find this particularly useful during busier times of the month or year.

To add a CLOG data set dynamically, the nucleus must know about its JCL at startup time. We recommend that you set up your Adabas nucleus startup jobs to include definition statements for the maximum number of CLOG data sets as you plan to use, but limit the actual usage of the CLOGs using the ADARUN `NCLOG` parameter. For example, you might start a nucleus with eight CLOG definitions in the Adabas startup JCL, but limit the number of CLOGs actually used during nucleus processing to three CLOGs by setting the `NCLOG` parameter to "3". When the nucleus starts up, only three CLOGs will be opened and logged in the PPT, even though eight are defined in the JCL. The additional CLOG data sets can then be dynamically added using this ADADBS ADDCLOG utility or its equivalent function in the Adabas Online System (AOS).



Note: Any CLOG data sets you add dynamically will not be retained once you recycle your Adabas nucleus. To retain these new CLOG data sets when Adabas is stopped and restarted, alter the Adabas startup JCL as well, ensuring that the number of CLOG definition statements in the JCL matches the increased number of CLOG data sets and that the `NCLOG` ADARUN parameter setting includes the new CLOG data sets.

Running the ADADBS ADDCLOG utility function is invalid when Adabas is running with dual CLOGs.

```
ADADBS  ADDCLOG NUMBER = clog-ds-number
                [NOUSERABEND]
                [NUCID = nucid ]
                [CLOGDEV = device-type ]
                [TEST]
```

This chapter describes the syntax, processing, and parameters of the ADADBS ADDCLOG function.

Essential Parameters

NUMBER: CLOG Data Set Number

Use the NUMBER parameter to specify the number of the nonsequential CLOG data set to be added. Valid values are integers ranging from "3" through "8" (inclusive).

Note: Be sure that the Adabas startup JCL allows for this additional CLOG data set by including a definition statement for the data set. If a definition statement is *not* already specified for this CLOG data set in the Adabas startup JCL, you will need to add it now and recycle the nucleus. Ideally, you would already have included definition statements in the JCL for all potential CLOG data sets, even though they are not all in use when the nucleus starts up.

NUCID: Cluster Nucleus ID

This parameter is required only in cluster environments.

Use the NUCID parameter to specify the nucleus ID of the Adabas within the cluster to which the new CLOG data set should be dynamically added.

Optional Parameters

CLOGDEV

Use the optional CLOGDEV parameter to specify the device type to be used for the new CLOG data set. This parameter is required only if a different device type from the device type specified by the ADARUN DEVICE parameter is to be used. The default is to use the device type specified by the ADARUN DEVICE parameter.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "`utility` TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See **Syntax Checking with the TEST Parameter** for more information about using the TEST parameter in ADADBS functions.

Examples

In the following example, CLOG data set 3 is dynamically added using a 3390 device.

```
ADADBS ADDCLOG NUMBER=3,CLOGDEV=3390
```

In the following example, CLOG data set 6 is dynamically added for the Adabas nucleus 65590 in a cluster environment.

```
ADADBS ADDCLOG NUMBER=6,NUCID=65590
```

50 ADDPLOG: Dynamically Add PLOG Data Sets

- Essential Parameters 335
- Optional Parameters 335
- Examples 336

The ADDPLOG function allows you to dynamically add a new protection log (PLOG) data set without terminating your current nucleus session. Using this utility function, you can specify up to eight PLOG data sets. This will reduce the chances of a wait condition in the nucleus, when the nucleus waits for an available PLOG. You might find this particularly useful during busier times of the month or year.

To add a PLOG data set dynamically, the nucleus must know about its JCL at startup time. We recommend that you set up your Adabas nucleus startup jobs to include definition statements for the maximum number of PLOG data sets as you plan to use, but limit the actual usage of the PLOGs using the ADARUN `NPLOG` parameter. For example, you might start a nucleus with eight PLOG definitions in the Adabas startup JCL, but limit the number of PLOGs actually used during nucleus processing to three PLOGs by setting the `NPLOG` parameter to "3". When the nucleus starts up, only three PLOGs will be opened and logged in the PPT, even though eight are defined in the JCL. The additional PLOG data sets can then be dynamically added using this ADADBS ADDPLOG utility or its equivalent function in the Adabas Online System (AOS).

Note: Any PLOG data sets you add dynamically will not be retained once you recycle your Adabas nucleus. To retain these new PLOG data sets when Adabas is stopped and restarted, alter the Adabas startup JCL as well, ensuring that the number of PLOG definition statements in the JCL matches the increased number of PLOG data sets and that the `NPLOG` ADARUN parameter setting includes the new PLOG data sets.

Running the ADADBS ADDPLOG utility function is invalid when Adabas is running with dual PLOGs.

```
ADADBS  ADDPLOG  NUMBER = plog-ds-number
                 [NOUSERABEND]
                 [NUCID = nucid ]
                 [PLOGDEV = device-type ]
                 [TEST]
```

This chapter describes the syntax, processing, and parameters of the ADADBS ADDPLOG function.

Essential Parameters

NUMBER: PLOG Data Set Number

Use the NUMBER parameter to specify the number of the nonsequential PLOG data set to be added. Valid values are integers ranging from "3" through "8" (inclusive).



Note: Be sure that the Adabas startup JCL allows for this additional PLOG data set by including a definition statement for the data set. If a definition statement is *not* already specified for this PLOG data set in the Adabas startup JCL, you will need to add it now and recycle the nucleus. Ideally, you would already have included definition statements in the JCL for all potential PLOG data sets, even though they are not all in use when the nucleus starts up.

NUCID: Cluster Nucleus ID

This parameter is required only in cluster environments.

Use the NUCID parameter to specify the nucleus ID of the Adabas within the cluster to which the new PLOG data set should be dynamically added.

Optional Parameters

PLOGDEV

Use the optional PLOGDEV parameter to specify the device type to be used for the new PLOG data set. This parameter is required only if a different device type from the device type specified by the ADARUN DEVICE parameter is to be used. The default is to use the device type specified by the ADARUN DEVICE parameter.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "`utility` TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See **Syntax Checking with the TEST Parameter** for more information about using the TEST parameter in ADADBS functions.

Examples

In the following example, PLOG data set 3 is dynamically added using a 3390 device.

```
ADADBS ADDPLOG NUMBER=3,CLOGDEV=3390
```

In the following example, PLOG data set 6 is dynamically added for the Adabas nucleus 65590 in a cluster environment.

```
ADADBS ADDPLOG NUMBER=6,NUCID=65590
```

51 ALLOCATE: Allocate File Extent

- Essential Parameters 338
- Optional Parameters 339
- Example 339

The ALLOCATE function may be used to allocate an address converter, Data Storage, normal or upper index extent of a specific size. It can also be used to allocate a secondary address converter when spanned records are included in the data. Only one extent may be allocated per ADADBS execution.

```
ADADBS  ALLOCATE    FILE = file-number
                    { ACSIZE | AC2SIZE | DSSIZE | NISIZE | UISIZE } = size
                    [DEVICE = device-type ]
                    [NOUSERABEND]
                    [PASSWORD = ' password ' ]
                    [STARTRABN = start-rabn ]
                    [TEST]
```

This chapter describes the syntax, processing, and parameters of the ADADBS ALLOCATE function.

Essential Parameters

FILE: File for Which an Extent Is Allocated

FILE specifies the number of the file for which the extent is to be allocated.

ACSIZE | AC2SIZE | DSSIZE | NISIZE | UISIZE: Extent Type and Size

These parameters are used to indicate the type and size of the extent to be allocated. One and only one extent type and size can be specified in a single ADADBS ALLOCATE statement. The specified value can be either cylinders or blocks; a size in blocks must be followed by "B" (for example, 2000B).

The extents that can be allocated are:

- the address converter (ACSIZE)
- the secondary address converter, when spanned records are used (AC2SIZE)
- Data Storage (DSSIZE)
- the normal index (NISIZE)
- the upper index (UISIZE).

Note: If the specified address converter size (ACSIZE or AC2SIZE) would increase the existing address converter beyond the maximum size derived from the ISNSIZE attribute of the file (for ISNSIZE=3: 16,777,215 ISNs; for ISNSIZE=4: 4,294,967,294 ISNs), Adabas will allocate additional address converter blocks only up to that limit.

Optional Parameters

DEVICE: Device Type

The device type to be used for file allocation. If none is specified, Adabas chooses one from the available device types with free space in the database.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PASSWORD: File Password

The password of the file. This parameter is required if the file is password-protected.

STARTRABN: Starting RABN for Extent

The beginning RABN of the extent to be allocated. If this parameter is omitted, ADADBS will assign the starting RABN.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See **Syntax Checking with the TEST Parameter** for more information on using the TEST parameter in ADADBS functions.

Example

An address converter extent of 30 blocks is to be allocated for file 15.

```
ADADBS ALLOCATE FILE=15,ACSIZE=30B ↩
```

52 CHANGE: Change Standard Length or Format of a Field

- Essential Parameters 342
- Optional Parameters 345
- Example 346

The CHANGE function can be used to change the standard length of an Adabas field or the format of a field, including the change of:

- a unique descriptor field to simply a descriptor field;
- the time zone of the field;
- a normal alphanumeric (A) field to a long-alpha (LA) field; or
- the default field format from unpacked (U) to packed (P).

Only one of these changes may be performed per function execution.

No modifications to records in Data Storage are made by this function. The user is, therefore, responsible for preventing references to the field that would cause invalid results because of an inconsistency between the new standard length as defined to Adabas and the actual number of bytes contained in the record.

When changing the length of an Adabas expanded file field, the change must be made to *each individual component file* of the expanded file. Each CHANGE operation on a component file causes a message that confirms the change, and returns condition code 4.



Essential Parameters

FILE: File Containing the Field

The file in which the field whose length is to be changed is contained. An Adabas system file may not be specified.

FIELD: Field to be Changed

The field whose standard length is to be changed. The field cannot be one that was defined with the FI option, or a field with a defined length of zero (variable-length field). Specify the field name between apostrophes (').

The FIELD and **FNDEF** parameters are mutually exclusive; either the FIELD parameter or the FNDEF parameter can be specified in a given ADADBS CHANGE utility request. However, more than one ADADBS CHANGE request can occur in a single run.

FNDEF: Adabas Field Definition to be Changed

FNDEF specifies an Adabas field (data) definition that should be changed. One FNDEF statement is required for each field to be changed. The syntax used in constructing field definition entries is:

```
FNDEF = 'level, name [ , length, format ] [, MU [(occurrences )] ] [ ,option ] ... '
FNDEF = 'level, name [ , PE [(occurrences )] ]'
```

Each definition must adhere to the field definition syntax as described for the ADACMP utility in *FNDEF: Field and Group Definition* and *FNDEF: Periodic Group Definition* in the section entitled *Field Definition Statements*, in the ADACMP documentation elsewhere in this guide.

Note the following Adabas field definitions change considerations:

- The only field format change supported is the change from U format to P format. This is only supported if the field does not have the FI (fixed storage length) option set and is not the parent of a special field or descriptor.
- Any new `length` specified must be compatible with the new field format and field options. Such a change alters the behavior of Adabas commands and utilities (such as ADACMP) where the field length is not specified in a format or search buffer.
- The following field option or option combination changes are allowed:

Field Options of Original Field Definition	Changes Allowed	Notes
CR (system field insertion only option) is set or is not set.	none	The setting of the CR option cannot be altered using ADADBS CHANGE. It must remain unchanged.
No DT (date-time edit mask) is set.	DT can be set. TZ (time zone) can be specified or not specified.	No data checking is performed to ensure that the values in the database are compliant with the DT edit mask specified. The date-time edit mask can only be changed for a field with the TZ option if the file has no records. TZ cannot be specified if the edit mask name is DATE, TIME, or NATDATE. In addition, due to the differing internal representations of date-time fields, TZ cannot be specified if the file already has records loaded.
DT (date-time edit mask) is set.	DT setting can be removed.	Once the DT setting is removed, you can no longer specify a date-time edit mask for the field in the format buffer. The date-time edit mask can only be changed for a field with the TZ option if the file has no records.

Field Options of Original Field Definition	Changes Allowed	Notes
NB (trailing blanks) is not set.	NB can be set.	Once NB is set, Adabas will not remove trailing blanks for the field.
NU (null value) is set.	NC (SQL null value) can be set	Once NC is set, empty values in fields are converted to null values.
NC (SQL null value) and NN (SQL no-null value) are set.	NC and NN can be removed.	Once NC and NN are removed, blank, non-null values are allowed for the field..
NC (SQL null value) and NN (SQL no-null value) are set.	NN can be removed, but NC must remain unchanged.	In insert commands, the field is not mandatory anymore in the format buffer. If not specified, the field will be assigned a null value.
NV (no conversion) is set.	NV can be removed.	Once NV is removed, A or W-format fields are processed in the record buffer after being converted. This changes the behavior of cross-platform calls.
NV (no conversion) is not set.	NV can be set.	Once NV is added, A or W-format fields are processed in the record buffer without being converted. This changes the behavior of cross-platform calls.
SY (system field) is set or is not set.	none	The setting of the SY option cannot be altered using ADADBS CHANGE. It must remain unchanged.
TZ (time zone) is not set, but a DT (date-time edit mask) is specified.	TZ can be set and DT must remain unchanged.	Once TZ is set, date-time values in the database are converted from UTC to local time when specifying a date-time edit mask. In addition, due to the differing internal representations of date-time fields, TZ cannot be specified if the file already has records loaded. **Note:** This is only allowed if the file has no records. TZ may not be set for date-time edit mask names DATE, TIME and NATDATE.
TZ (time zone) is set.	TZ can be removed.	If the file already has records loaded, TZ cannot be removed due to the differing internal representations of date-time fields. Once TZ is removed, date-time values in the database are no longer converted from UTC to local time when specifying a date-time edit mask. **Note:** This is only allowed if the file has no records.

For example, assume that the FDT has the field NT defined as `01,NT,7,P,NU`. The following ADADBS CHANGE statement adds the date-time edit mask NATTIME to the NT field definition:

```
ADADBS CHANGE FILE=201,FNDEF='01,NT,7,P,NU,DT=E(NATTIME)'
```

The **FIELD** and FNDEF parameters are mutually exclusive; either the FIELD parameter or the FNDEF parameter can be specified in a given ADADBS CHANGE utility request.

FORMAT=P: New Field Format

The new standard field format. The only field format change supported is from "U" (unpacked) to "P" (packed). The field cannot be a parent of a subdescriptor, a superdescriptor, or a hyper-descriptor.

One of the subparameters FORMAT, LENGTH, or OPTION must be specified when the FIELD parameter is specified; but only one of the three may be specified.

LENGTH: New Field Length

The new standard length for the field. A length of 0 is not permitted, nor can a field with an existing defined length of zero (such as a variable-length field) be redefined to a standard length.

One of the subparameters FORMAT, LENGTH, or OPTION must be specified when the FIELD parameter is specified; but only one of the three may be specified.

OPTION: New Field Option

The new field option. The following field option changes are supported:

Field Option	Description
LA	Changes a normal alphanumeric (A) field to long-alpha (LA).
NOUQ	Removes the unique descriptor (UQ) option from a descriptor field (DE).

One of the subparameters FORMAT, LENGTH, or OPTION must be specified when the FIELD parameter is specified; but only one of the three may be specified.

Optional Parameters

INPARALLEL: Execute function in parallel

When INPARALLEL=YES (the default) is specified, the function may be executed even if other users are active on the file. Adabas performs the function while all application commands against the file are temporarily quiesced and suspended. The integrity of the file targeted by the ADADBS function is maintained.

INPARALLEL=YES can be specified only for the function CHANGE FIELD with OPTION=NOUQ.

With INPARALLEL=NO, the function requires exclusive file control – that is, no applications can be executing against the file. Adabas refuses to execute the function (with response code 48) if other users are active on the file.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PASSWORD: File Password

The password of the file containing the field to be changed. This parameter is required if the file is password-protected.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See **Syntax Checking with the TEST Parameter** for more information on using the TEST parameter in ADADBS functions.

Example

The standard length of field AB in file 5 is to be changed to 11 bytes.

```
ADADBS CHANGE FILE=5,FIELD='AB',LENGTH=11
```

53 CVOLSER: Print Adabas Extents on Given Volume

- Essential Parameter 348
- Optional Parameters 348
- Example 349

The CVOLSER function is used to print the Adabas file extents contained on a disk volume.

```
ADADBS  CVOLSER  VOLSER = volume-serial-number
                 [NOUSERABEND]
                 [TEST]
```

Essential Parameter

VOLSER: Volume Serial Number

VOLSER is the volume serial number of the disk volume to be used.

Optional Parameters

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See **Syntax Checking with the TEST Parameter** for more information on using the TEST parameter in ADADBS functions.

Example

The Adabas file extents contained on disk volume DISK02 are to be printed.

```
ADADBS CVOLSER VOLSER=DISK02
```

54 DEALLOCATE: Deallocate File Extent

- Essential Parameters 352
- Optional Parameters 353
- Example 353

The DEALLOCATE function may be used to deallocate an address converter, Data Storage, normal index or upper index extent. It can also be used to deallocate a secondary address converter when spanned records are included in the data. Only one extent may be deallocated per ADADBS execution.

```
ADADBS  DEALLOCATE  FILE = file-number
                    { ACSIZE | AC2SIZE | DSSIZE | NISIZE | UISIZE } = size
                    [NOUSERABEND]
                    [PASSWORD = ' password ' ]
                    [STARTRABN = start-rabn ]
                    [TEST]
```

Essential Parameters

ACSIZE | AC2SIZE | DSSIZE | NISIZE | UISIZE: Extent Type and Size

These parameters specify the type and size of extent to be deallocated. One and only one extent type and size may be specified. The size must be in number of RABN blocks followed by "B" (for example, DSSIZE=20B), and cannot exceed the number of unused RABNs at the end of an extent.

The extents that can be deallocated are:

- the address converter (ACSIZE)
- the secondary address converter, when spanned records are used (AC2SIZE)
- Data Storage (DSSIZE)
- the normal index (NISIZE)
- the upper index (UISIZE).

FILE: File for Which an Extent Is Deallocated

FILE specifies the file for which the extent is to be deallocated. Specify a decimal value.

Optional Parameters

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PASSWORD: File Password

The password of the file for which space is to be deallocated. This parameter is required if the file is password-protected. Specify the password between apostrophes (').

STARTRABN: Starting RABN for Extent

The first RABN of the extent in which deallocation is to take place. If this parameter is omitted, the last extent for the file will be deallocated. In the address converter, only the last extent may be deallocated.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See **Syntax Checking with the TEST Parameter** for more information on using the TEST parameter in ADADBS functions.

Example

An address converter extent of 30 blocks is to be deallocated for file 15.

```
ADADBS DEALLOCATE FILE=15,ACSIZE=30B
```

55 DECREASE: Decrease Last Associator or Data Storage Data Set Size

- Essential Parameter 356
- Optional Parameters 356
- Example 357
- Procedure 357

The DECREASE function decreases the size of the last data set currently being used for the Associator or Data Storage. The space to be released must be available in the free space table (FST).

The DECREASE function does *not* deallocate any of the specified physical extent space.

```
ADADBS  DECREASE   { ASSOSIZE | DATASIZE } = sizeB
                   [NOUSERABEND]
                   [TEST]
```

Essential Parameter

ASSOSIZE | DATASIZE: Blocks to Be Decreased

ASSOSIZE/DATASIZE define the number of blocks by which the Associator or Data Storage data set is to be decreased, specified as a decimal value followed by "B". Either ASSOSIZE or DATASIZE can be specified, but not both. If both ASSOSIZE and DATASIZE are to be specified, each must be entered on a separate ADADBS DECREASE statement.

Optional Parameters

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See **Syntax Checking with the TEST Parameter** for more information on using the TEST parameter in ADADBS functions.

Example

The Associator is to be decreased by 100 blocks and Data Storage is to be decreased by 200 blocks.

```
ADADBS DECREASE ASSOSIZE=100B
ADADBS DECREASE DATASIZE=200B
```

Procedure

To deallocate space, perform the following steps:

1 Decrease the database with the DECREASE function;

2 Save the database with ADASAV SAVE;

3 Reformat the data sets with ADAFRM;

4 Restore the database with ADASAV.

56 DELCLOG: Dynamically Deleting CLOG Data Sets

- Essential Parameters 360
- Optional Parameters 361
- Examples 361

The DELCLOG function allows you to dynamically delete a command log (CLOG) data set without terminating your current nucleus session. CLOG data sets can also be dynamically deleted using an equivalent function in the Adabas Online System (AOS).

Note: Any CLOG data sets you delete dynamically may reappear once you recycle your Adabas nucleus. To ensure the CLOG data set is dropped when Adabas is stopped and re-started, alter the Adabas startup JCL as well, ensuring that the `NCLOG` ADARUN parameter setting is reduced to account for the dropped CLOG data sets.

```
ADADBS  DELCLOG  NUMBER = clog-ds-number
                 [NOUSERABEND]
                 [NUCID = nucid ]
                 [TEST]
```

Running the ADADBS DELCLOG utility function is invalid when Adabas is running with dual CLOGs.

This chapter describes the syntax, processing, and parameters of the ADADBS DELCLOG function.

Essential Parameters

NUMBER: CLOG Data Set Number

Use the NUMBER parameter to specify the number of the nonsequential CLOG data set to be deleted. Valid values are integers ranging from "3" through "8" (inclusive).

NUCID: Cluster Nucleus ID

This parameter is required only in cluster environments.

Use the NUCID parameter to specify the nucleus ID of the Adabas within the cluster to which the CLOG data set should be dynamically deleted.

Optional Parameters

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See **Syntax Checking with the TEST Parameter** for more information about using the TEST parameter in ADADBS functions.

Examples

In the following example, CLOG data set 3 is dynamically deleted from its 3390 device.

```
ADADBS DELCLOG NUMBER=3,CLOGDEV=3390
```

In the following example, CLOG data set 6 is dynamically deleted for the Adabas nucleus 65590 in a cluster environment.

```
ADADBS DELCLOG NUMBER=6,NUCID=65590
```

57 DELCP: Delete Checkpoint Records

- Essential Parameter 364
- Optional Parameters 364
- Example 365

The DELCP function deletes checkpoint records.

After running ADADBS DELCP, the remaining records are reassigned ISNs to include those ISNs made available when the checkpoint records were deleted. The lower ISNs are assigned but the chronological order of checkpoints is maintained.

```
ADADBS  DELCP  TODATE = yyyymmdd
               [NOUSERABEND]
               [TEST]
```

Essential Parameter

TODATE: Last Date for Deleted Records

TODATE specifies the latest date for which checkpoint information is deleted. Checkpoint information dated after the date specified by TODATE= is not deleted. TODATE= must be specified; there is no default date. Specify the date as a four-digit decimal value for year followed by two-digit decimal values for month and day, in that order.

Optional Parameters

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See **Syntax Checking with the TEST Parameter** for more information on using the TEST parameter in ADADBS functions.

Example

All checkpoint records up to and including February 1, 1996 are to be deleted.

```
ADADBS DELCP TODATE=19960201
```

58 DELDE: Logically Deleting a Descriptor

- Essential Parameters 368
- Optional Parameters 368
- Example 369

The DELDE function logically deletes a descriptor from a file. A logically deleted descriptor cannot be used as a search descriptor.

In reports retrieved by the ADAREP FDT function, the Options field in the Field Definition Table uses “DL” to indicate that a descriptor is logically deleted. In the output from the ADACHK FDTPRINT function, deleted logical descriptors are listed as “DE*”.

```
ADADBS  DELDE  FILE = file-number
               DESCRIPTOR = descriptor-name
               [NOUSERABEND]
               [PASSWORD = password ]
               [TEST]
```

Essential Parameters

FILE: File Number

FILE specifies the file from which the descriptor is to be logically deleted. Specify a decimal value.

Descriptor: Descriptor Name

DESCRIPTOR identifies the descriptor to be logically deleted. Specify a valid descriptor name.

Optional Parameters

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PASSWORD: File Password

The password of the file from which the descriptor is to be logically deleted. This parameter is required if the file is password-protected. Specify the password between apostrophes (').

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See **Syntax Checking with the TEST Parameter** for more information on using the TEST parameter in ADADBS functions.

Example

The following example logically removes field AA as a descriptor field in file 12 (which requires password XYZPSWD).

```
ADADBS DELDE FILE=12,DESCRIPTOR=AA,PASSWORD=XYZPSWD
```

59 DELETE: Delete File

- Essential Parameter 372
- Optional Parameters 372
- Examples 373

The DELETE function deletes an Adabas file from the database.

```
ADADBS  DELETE   { FILE = fnr  [KEEPFDT] [PASSWORD = 'password' ] }
                 [NOUSERABEND]
                 [TEST]
```

When an Adabas file is deleted from the database, all logical extents assigned to the file are deallocated. The released space may be used for a new file or for a new extent of an existing file.

The file to be deleted may not be coupled. If an Adabas expanded file is specified, the complete expanded file (the anchor and all component files) is deleted.

When the DELETE function completes successfully, if the file was previously locked automatically by Adabas, the automatic lock is reset. Any locks previously set with the operator commands LOCKU or LOCKF remain set.

Essential Parameter

FILE: File to Be Deleted

FILE specifies the number of the Adabas file to be deleted. Checkpoint, security, trigger, and any other files loaded with the ADALOD utility's SYSFILE option may be specified *only* if ADADBS DELETE is the only Adabas user; deleting any these files automatically causes Adabas to terminate when finished. To delete an Adabas expanded file, specify the file number (also the anchor file).

Optional Parameters

KEEPFDT: Retain the Field Definition Table

The KEEPFDT parameter, if specified, instructs ADADBS DELETE to keep the deleted file's field definition table (FDT) for later use by ADACMP. If this parameter is specified, a file with the same number as the one now being deleted can only be later loaded if either the new file's FDT is the same as that of the deleted file, or the load operation specifies the IGNFDT parameter to accept the new file's FDT.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend

after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PASSWORD: File Password

PASSWORD specifies the password of the file to be deleted. This parameter is required if the file is password-protected.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See **Syntax Checking with the TEST Parameter** for more information on using the TEST parameter in ADADBS functions.

Examples

File 6 is to be deleted.

```
ADADBS DELETE FILE=6
```

Password-protected file 10 is to be deleted. The field definition table is to be retained. File number 10 cannot be used again until another ADALOD LOAD command is issued with the IGNFDT option.

```
ADADBS DELETE
FILE=10,KEEPFDT,PASSWORD='FILE10'
```

60 DELFN: Logically Delete Fields

- Essential Parameter 377
- Optional Parameters 377
- Example 378

The DELFN function allows you to logically delete fields from an Adabas database file. Logically deleting a field from a file removes the field from the FDT for the file, but does not delete the data in the database. This means that a logically deleted field will *not*:

- appear in data selected by the ADASEL utility.
- be decompressed in ADACDC utility runs; the output from the run will not contain logically deleted fields.
- be decompressed by ADACMP DECOMPRESS utility runs; the output from the run will not contain logically deleted fields.

But a logically deleted field will:

- appear in data unloaded by the ADAULD utility.
- be present in ADAORD utility output and the ADAORD STORE utility functions will load the data for logically deleted fields.

Report output produced by the ADACHK FDTPRINT and DSCHECK utility functions lists logically defined fields as asterisks (**). Report output produced by the ADAREP utility identifies and reports on logically deleted fields in the database files. In addition, you cannot use the ADACNV utility to revert to a version of the database older than Adabas 8.2 if logically deleted files exist in the database files.

Caution: ADACMP COMPRESS utility runs that specify an FDT (via the FDT parameter) but do not specify a FORMAT parameter and that run against a file with logically deleted fields require that the data include the values for the logically deleted fields. Failure to include these values could lead to incorrectly compressed records.

Descriptor fields, parents of descriptor fields, subfields, superfields, phonetic fields, collating fields, and fields in large object (LOB) files cannot be deleted.

```
ADADBS DELFN    FILE = file-number
                FIELDLIST = ' field-list '
                [PASSWORD = password ]
                [NOUSERABEND]
                [TEST]
```

Essential Parameter

FILE: File Number

FILE specifies the database file number from which the fields should be logically deleted. The file number may not be the number of a large object (LOB) file; fields in LOB files cannot be deleted.

FIELDLIST: List of Fields

FIELDLIST specifies a list of one or more fields. At least one field must be specified. If more than one field will be deleted, separate the field names with commas (,). A maximum of 800 fields may be specified. A field may only be listed once in an ADADBS DELFN run.

Descriptor fields cannot be deleted, so they cannot be included in the list. Likewise, parent fields of subdescriptor, superdescriptor, hyperdescriptor, phonetic descriptor, and collating descriptor fields cannot be deleted.

Optional Parameters

PASSWORD: File Password

PASSWORD specifies the password of the file containing fields to be logically deleted. This parameter is required if the file is password-protected.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See **Syntax Checking with the TEST Parameter** for more information on using the TEST parameter in ADADBS functions.

Example

In the following example, fields AA, AB, and AC are logically deleted from file 12.

```
ADADBS DELFN FILE=12,FIELDLIST='AA,AB,AC'
```

61 DELPLOG: Dynamically Deleting PLOG Data Sets

- Essential Parameters 380
- Optional Parameters 381
- Examples 381

The DELPLOG function allows you to dynamically delete a protection log (PLOG) data set without terminating your current nucleus session. PLOG data sets can also be dynamically deleted using an equivalent function in the Adabas Online System (AOS).

Note: Any PLOG data sets you delete dynamically may reappear once you recycle your Adabas nucleus. To ensure the PLOG data set is dropped when Adabas is stopped and re-started, alter the Adabas startup JCL as well, ensuring that the `NPLOG` ADARUN parameter setting is reduced to account for the dropped PLOG data sets.

```
ADADBS  DELPLOG  NUMBER = plog-ds-number
                 [NOUSERABEND]
                 [NUCID = nucid ]
                 [TEST]
```

Running the ADADBS DELPLOG utility function is invalid when Adabas is running with dual PLOGs.

This chapter describes the syntax, processing, and parameters of the ADADBS DELPLOG function.

Essential Parameters

NUMBER: PLOG Data Set Number

Use the NUMBER parameter to specify the number of the nonsequential PLOG data set to be deleted. Valid values are integers ranging from "3" through "8" (inclusive).

NUCID: Cluster Nucleus ID

This parameter is required only in cluster environments.

Use the NUCID parameter to specify the nucleus ID of the Adabas within the cluster to which the PLOG data set should be dynamically deleted.

Optional Parameters

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See **Syntax Checking with the TEST Parameter** for more information about using the TEST parameter in ADADBS functions.

Examples

In the following example, PLOG data set 3 is dynamically deleted from its 3390 device.

```
ADADBS DELPLOG NUMBER=3,CLOGDEV=3390
```

In the following example, PLOG data set 6 is dynamically deleted for the Adabas nucleus 65590 in a cluster environment.

```
ADADBS DELPLOG NUMBER=6,NUCID=65590
```

62 DEVENTLOG: Display Adabas Event Log

- Optional Parameters 384
- Examples 385

The DEVENTLOG function allows you to display all entries in the Adabas event log, which stores various events such as occurrences of response code 145 (ADARSP145), or of response code 200 (ADARSP200) for unknown or changed security system user IDs (SECUID).

Note: You can also obtain event log information in the application that received the response 145 with an L2 on a virtual system file -4. For more information, read *Special L2 Command Call for Response 145*, in the *Adabas Command Reference Guide*.

The Adabas event log is a wraparound log in memory. The INFOBUFFERSIZE ADARUN parameter identifies the size of the Adabas event log. When the Adabas event log fills up, the oldest entries in the log are overwritten.

```
ADADBS DEVENTLOG [NOUSERABEND]
                 [NUCID = nucid ]
                 [TEST]
```

This chapter describes the syntax, processing, and parameters of the ADADBS DEVENTLOG function.

Optional Parameters

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

NUCID

Use the optional NUCID parameter to specify the ID of a specific Adabas nucleus for which the Adabas event log should be displayed. This parameter is especially useful in cluster environments. If no specific database ID is specified in a cluster environment, the Adabas event logs of all the databases in the cluster are displayed.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified

parameters can be tested. See **Syntax Checking with the TEST Parameter** for more information about using the TEST parameter in ADADBS functions.

Examples

In the following example, the Adabas event log for Adabas nucleus 12 is displayed.

```
ADADBS DEVENTLOG NUCID=12
```

In the following example, the individual Adabas event log is displayed, or in cluster environments, all of the Adabas event logs for every database in the cluster is displayed.

```
ADADBS DEVENTLOG
```

63 DSREUSE: Reuse Data Storage Blocks

- Essential Parameters 388
- Optional Parameters 388
- Example 389

The DSREUSE function controls the reuse of Data Storage blocks.

```
ADADBS  DSREUSE   FILE = file-number
                  MODE = { ON | OFF [,INPARALLEL=YES|NO]}
                  [NOUSERABEND]
                  [PASSWORD = 'password ' ]
                  [RESET]
                  [TEST]
```

This utility function does not need to lock the database file for its use; this function can perform its processing in parallel with active users. This means that you do not need to set the file to read-only status to run this utility function.

Essential Parameters

FILE: File Number

FILE is the number of the file for which the DSREUSE setting is to apply.

Block reuse is originally determined when the file is loaded into the database with the ADALOD FILE function, or when the system file is defined with the ADADEF DEFINE function. In both cases, block reuse defaults to "YES" unless specified otherwise in those functions.

MODE: Reuse Mode

The Data Storage block assignment mode to be in effect. MODE=OFF indicates that Data Storage blocks which become free as a result of record deletion may not be reused, in effect cancelling the ADADBS DSREUSE function. MODE=ON indicates that Data Storage blocks may be reused. The MODE= parameter has no default, and must be specified.

Optional Parameters

INPARALLEL: Execute function in parallel

When INPARALLEL=YES (the default) is specified, the function may be executed even if other users are active on the file. Adabas performs the function while all application commands against the file are temporarily quiesced and suspended. The integrity of the file targeted by the ADADBS function is maintained.

With INPARALLEL=NO, the function requires exclusive file control – that is, no applications can be executing against the file. Adabas refuses to execute the function (with response code 48) if other users are active on the file.

System files require INPARALLEL=NO.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PASSWORD: File Password

PASSWORD specifies the file's security password, and is required if the file is password-protected.

RESET: Reset Space Pointer

The RESET parameter causes searches for new Data Storage space to start at the beginning of the file.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See **Syntax Checking with the TEST Parameter** for more information on using the TEST parameter in ADADBS functions.

Example

Data Storage blocks for file 6 are not to be reused.

```
ADADBS DSREUSE FILE=6,MODE=OFF
```

64 ENCODEF: Change File Encoding

- Essential Parameter 392
- Optional Parameters 393
- Example 393

```
ADADBS  ENCODEF   FILE = file-number
                  FACODE =  alpha-key | UWCODE = wide-key
                  [NOUSERABEND]
                  [TEST]
```

This chapter describes the syntax, processing, and parameters of the ADADBS ENCODEF function.

Essential Parameter

FILE: File Number

FILE is the number of the file for which encoding is to be changed.

FACODE: Encoding for Alphanumeric Fields in File

The FACODE parameter defines the encoding for alphanumeric fields stored in the file. It can be applied to files already loaded. The encoding must be derived from EBCDIC encoding; that is, X'40' is the space character. Double-byte character set (DBCS) type encodings are supported with the exception of DBCS-only. See *Supplied UES Encodings*, in *Adabas DBA Tasks Manual*, for a list of supplied code pages.

FACODE and UWCODE are mutually exclusive parameters; if one is specified, the other should not be. But one of them must be specified.

UWCODE: User Encoding for Wide-Character Fields in File

The UWCODE parameter defines the user encoding for wide-character fields stored in the file. It can be applied to files already loaded. Note that the wide file encoding is not changed.

To change the encoding of wide-character fields, the file must be unloaded, decompressed, compressed, and reloaded. See *Supplied UES Encodings*, in *Adabas DBA Tasks Manual*, for a list of supplied code pages.

FACODE and UWCODE are mutually exclusive parameters; if one is specified, the other should not be. But one of them must be specified.

Optional Parameters

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See *Syntax Checking with the TEST Parameter* for more information about using the TEST parameter in ADADBS functions.

Example

In the following example, ADADBS ENCODEF is used to change the encoding of alphanumeric fields in file 1425 to use code page 285 (CECP: United Kingdom, EBCDIC-compatible with X'40' fill character). In file 401, ADADBS ENCODEF is used to change the encoding of wide fields to use code page 3396 (IBM, CCSID 4396, Japanese host double byte including 1880 user-defined characters). Note that because UWCODE is changing, file 401 must be unloaded, decompressed, compressed, and reloaded.

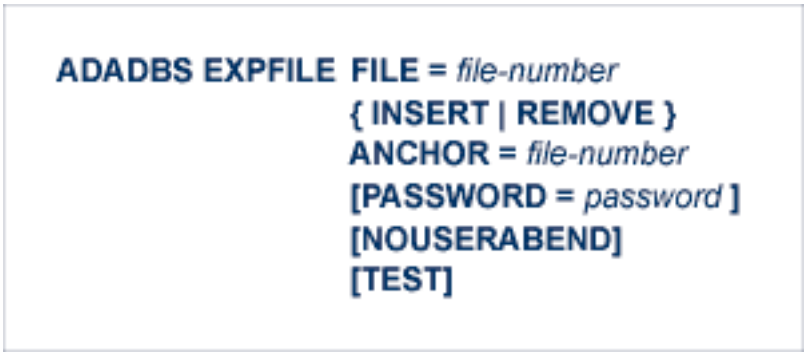
```
ADADBS ENCODEF FILE=1425,FACODE=285

ADADBS ENCODEF FILE=401,UWCODE=3396
```

65 EXPFILE: Insert or Remove Files in an Expanded File Chain

- Essential Parameters 396
- Optional Parameters 397
- Example 397

The EXPFILE function inserts or removes an Adabas file from an expanded file chain.



If an anchor file is removed from the expanded file chain, the next file in the chain becomes the anchor file.

If all files are removed from an expanded file chain, the flags set to indicate that a file is an anchor file are reset. To reestablish a file as an anchor file, assign both the ANCHOR and FILE parameters the file number of the file.

This chapter covers the following topics.

Essential Parameters

FILE: File to be Inserted or Deleted

FILE specifies the number of the Adabas file that should be inserted or removed in the expanded file chain.

INSERT or REMOVE: Action to Perform

Either INSERT or REMOVE must be specified to identify the action that should be performed by the EXPFILE utility function. INSERT will add a file to an expanded file chain; REMOVE will remove a file from an expanded file chain. INSERT and REMOVE are mutually exclusive functions; both of them cannot be specified in the same ADADBS EXPFILE run.

ANCHOR: File Number of the Anchor File

ANCHOR identifies the file number of the anchor file for the expanded file chain.

Optional Parameters

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PASSWORD: File Password

PASSWORD specifies the password of the file to be deleted. This parameter is only required if the file is password protected.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See *Syntax Checking with the TEST Parameter* for more information about using the TEST parameter in ADADBS functions.

Example

In the following example, file 6 will be added into an expanded file chain and will also become the anchor file for the chain:

```
ADADBS EXPFILE FILE=6,INSERT,ANCHOR=6
```

In the following example, file 7 will be inserted into the expanded file chain that is anchored by file 6:

```
ADADBS EXPFILE FILE=7,INSERT,ANCHOR=6
```

66 INCREASE: Increase Last Associator or Data Storage Data Set Size

- Essential Parameter 400
- Optional Parameters 400
- Example 401
- General Procedure 401
- z/OS Procedures 401

The INCREASE function increases the size of the last data set currently being used for the Associator or Data Storage. This function may be executed any number of times for the Associator. The maximum of 99 Data Storage Space Tables (DSSTs) somewhat limits Data Storage increases before all 99 Data Storage extents must be combined into a single extent with either the REORASSO or REORDB function of the ADAORD utility.

Notes:

1. The Associator and Data Storage data set sizes must be increased separately. It is *not* possible to increase both with a single operation.
2. After an INCREASE operation is completed, the INCREASE function automatically ends the current nucleus session. This allows for the necessary Associator or Data Storage formatting with ADAFRM before a new session is started. An informational message occurs to tell you that the nucleus has been stopped.

```
ADADBS  INCREASE  { ASSOSIZE | DATASIZE } = size
                  [ NOUSERABEND ]
                  [TEST]
```

Essential Parameter

ASSOSIZE | DATASIZE: Size to Be Increased

The additional number of blocks or cylinders needed by the Associator or Data Storage data set. To specify blocks, add a "B" after the value; for example, DATASIZE=50B.

Optional Parameters

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

Use the TEST parameter to test the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See **Syntax Checking with the TEST Parameter** for more information about using this parameter.

Example

The Associator is to be increased by 400 cylinders.

```
ADADBS INCREASE ASSOSIZE=400
```

General Procedure

The general procedure for increasing the size of the Associator or Data Storage is as follows:

1. Back up the database using the ADASAV utility. This step is optional but recommended.
2. Execute the ADADBS INCREASE function.
3. Format the additional space being added to the data set with the ADAFRM utility.

z/OS Procedures

The data set may be extended and the new blocks formatted by specifying the DISP=MOD parameter in the JCL. The SPACE parameter for the data set being increased should be set to

```
SPACE=(CYL,(0,n))
```

where *n* is the amount of space (in cylinders) being added. The ADAFRM control statement should also specify the number of cylinders being added.

If sufficient space is available on the volume or volumes currently comprising the data set, follow example 1. If additional volumes are required to satisfy the increase request, follow example 2 (for non DFSMS-managed data sets) or 3 (for DFSMS-managed data sets).

Example 1: INCREASE not requiring additional volumes

400 cylinders are to be added to an Associator data set. The control statement for the INCREASE function would be:

```
ADADBS INCREASE ASSOSIZE=400
```

The following JCL example increases the Associator data set using ADAFRM:

```
//DDASSOR1  DD
DSN=ADA.DB123.ASSOR1,DISP=MOD,SPACE=(CYL,(0,400))
```

and the actual ADAFRM control statement would be

```
ADAFRM ASSOFRM SIZE=400
```

Example 2: INCREASE requiring additional volumes (not DFSMS-managed)

400 cylinders are to be added to a single-volume Associator data set on volume VOL001. For the increase, the data set needs to extend to volume VOL002. The control statement for the INCREASE function would be:

```
ADADBS INCREASE ASSOSIZE=400
```

Include the following step after the INCREASE step but before the FORMAT step to uncatalog the data set:

```
//UNCATLG  EXEC PGM=IEFBR14
//DDASSOR1  DD  DSN=ADA.DB123.ASSOR1,DISP=(SHR,UNCATLG)
```

The following JCL example recatalogs the data set with the additional volume and increases the Associator data set using ADAFRM:

```
//DDASSOR1  DD  DSN=ADA.DB123.ASSOR1,
//   DISP=(MOD,CATLG),VOL=SER=(VOL001,VOL002),SPACE=(CYL,(0,400))...
```

and the actual ADAFRM control statement would be:

```
ADAFRM ASSOFRM SIZE=400
```

To provide the increase in example 1 for multiple volumes, specify the volumes in the JCL:

```
//DDASSOR1  DD  DSN=...
//
DISP=(MOD,CATLG),VOL=SER=(V1,V2,...),SPACE=(CYL,(0,400))...
```

Include the following step after the INCREASE step but before the FORMAT step to ensure a correct catalog entry:

```
//UNCATLG  EXEC PGM=IEFBR14
//DDASSOR1  DD  DSN=...,DISP=(SHR,UNCATLG)
```

Example 3: INCREASE requiring additional volumes (DFSMS-managed)

400 cylinders are to be added to a DFSMS-managed Associator data set. For the increase, the data set needs to extend onto another candidate volume belonging to the DFSMS Storage Group. The control statement for the INCREASE function would be:

```
ADADBS INCREASE ASSOSIZE=400
```

Include the following step after the INCREASE step but before the FORMAT step to add further candidate volumes to the catalog entry for the data set:

```
//ADDVOL EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
ALTER ADA.DB123.ASSOR1 ADDVOLUMES(*)
/*
```

The following JCL example increases the Associator data set using ADAFRM:

```
//DDASSOR1  DD  DSN=ADA.DB123.ASSOR1,DISP=MOD,SPACE=(CYL,(0,400))...
```

and the actual ADAFRM control statement would be:

```
ADAFRM ASSOFRM SIZE=400
```

67 ISNREUSE: Reuse ISNs

- Essential Parameters 406
- Optional Parameters 406
- Examples 407

The ISNREUSE function controls whether ISNs of deleted records may be reassigned to new records.

Note: ISNREUSE can cause excessive fragmentation of the file if the file is very volatile having many deletes and inserts. Adabas uses a performance based algorithm to search for a free ISN when doing an insert. RSP078 may be returned indicating that the file has reached the TOPISN and the file is severely fragmented.

```
ADADBS  ISNREUSE  FILE = file-number
                  MODE = { ON | OFF [,INPARALLEL=YES|NO]}
                  [NOUSERABEND]
                  [PASSWORD = 'password ' ]
                  [RESET]
                  [SETISN=isn]
                  [TEST]
```

This utility function does not need to lock the database file for its use; this function can perform its processing in parallel with active users. This means that you do not need to set the file to read-only status to run this utility function.

Essential Parameters

FILE: File Number

FILE is the number of the file for which the ISNREUSE setting is to be changed. The checkpoint file cannot be specified. Each individual file in an expanded file chain is processed separately and can be reset individually.

MODE: Reuse Mode

MODE causes the ISN reuse mode to be in effect. MODE=OFF causes Adabas not to reuse the ISN of a deleted record for a new record. Each new record will be assigned the next higher unused ISN. MODE=ON indicates that Adabas may reuse the ISN of a deleted record. The MODE parameter has no default; it must be specified.

Optional Parameters

INPARALLEL: Execute function in parallel

When INPARALLEL=YES (the default) is specified, the function may be executed even if other users are active on the file. Adabas performs the function while all application commands against the file are temporarily quiesced and suspended. The integrity of the file targeted by the ADADBS function is maintained.

With INPARALLEL=NO, the function requires exclusive file control – that is, no applications can be executing against the file. Adabas refuses to execute the function (with response code 48) if other users are active on the file.

System files require INPARALLEL=NO.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PASSWORD: File Password

PASSWORD specifies the file's security password, and is required if the file is password-protected.

RESET: Reset ISN Pointer

The RESET parameter causes searches for an unused ISN to start at the beginning of the file.



Note: If the file has reached TOPISN and the available ISNs are in the middle of the file, it may be a better alternative to reorganize the file (unload/reload) for performance reasons.

SETISN: Specific ISN to set the Rotating AC Pointer to

This specifies which ISN will start the search for a reusable ISN. If zero is specified it will be set to one.



Note: SETISN has no effect on AC2. SETISN is only valid when MODE=ON and RESET are specified.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See **Syntax Checking with the TEST Parameter** for more information on using the TEST parameter in ADADBS functions.

Examples

```
ADADBS ISNREUSE FILE=7,MODE=ON
```

ISNs of deleted records in file 7 may be reassigned to new records.

```
ADADBS ISNREUSE FILE=3,MODE=ON,RESET
```

This enables REUSABLE ISNs for file 3. Deleted ISNs may be reassigned to new records. The search for a reusable ISN will start at the beginning of the file.

```
ADADBS ISNREUSE FILE=3,MODE=ON,RESET,SETISN=6000
```

This enables REUSABLE ISNs for file 3. Deleted ISNs may be reassigned to new records. The search for a reusable ISN will start at ISN 6000.

68 MODFCB: Modify File Parameters

- Essential Parameter 410
- Optional Parameters 410
- Example 413

The MODFCB function modifies various parameters for a non-system Adabas file.

```
ADADBS MODFCB  FILE = file-number
               [ASSOPFAC = new-padding-factor ]
               [DATAPFAC = new-padding-factor ]
               [FILEREADONLY = { YES | NO } ]
               [INPARALLEL={ YES|NO }]
               [LOBFILE = file-number ]
               [MAXDS = maximum-secondary-allocation ]
               [MAXNI = maximum-secondary-allocation ]
               [MAXUI = maximum-secondary-allocation ]
               [MAXRECL = maximum-compressed-record-length ]
               [NOUSERABEND]
               [PASSWORD = ' password ' ]
               [PGMREFRESH = { YES | NO } ]
               [RPLUPDATEONLY = {YES | NO} ]
               [SYFMAXUV = nn ]
               [TEST]
```

Essential Parameter

FILE: File Number

FILE is the number of the Adabas file to be modified. An Adabas system file cannot be specified. If large object (LB) fields exist in the file, this is the file number of the *base file*.

Optional Parameters

ASSOPFAC/ DATAPFAC: File Padding Factors

ASSOPFAC/DATAPFAC specify the padding factor (1-90) to be in effect for Associator and Data Storage, respectively. Existing blocks retain their original padding factor (see the ADAORD utility).

The Data Storage padding factor (DATAPFAC) setting is ignored for files containing spanned records. The following warning will appear if DATAPFAC is specified for a file that has data storage spanning enabled:

```
*****DATAPFAC is ignored for files with spanned data storage enabled******
```

A return code of 4 (CC=4) is also returned.

FILEREADONLY: Utility Update Only Flag

The optional FILEREADONLY parameter can be used to indicate whether an Adabas database file should be placed in readonly status (where it can be updated only by Adabas utilities) or

in normal status (where it can be updated in any normal manner). Valid values are "YES" (place the file in readonly status) or "NO" (place the file in normal status). There is no default value.



Note: The ADADBS DSREUSE, ISNREUSE, NEWFIELD, RELEASE DE, and RENAME utility functions do not need to lock the file for their use; these functions can perform their processing in parallel with active users. This means that you do not need to set a file in read-only status (FILEREADONLY=YES) if you will be performing these other ADADBS functions only.

INPARALLEL: Execute function in parallel

When INPARALLEL=YES (the default) is specified, the function may be executed even if other users are active on the file. Adabas performs the function while all application commands against the file are temporarily quiesced and suspended. The integrity of the file targeted by the ADADBS function is maintained.

With INPARALLEL=NO, the function requires exclusive file control – that is, no applications can be executing against the file. Adabas refuses to execute the function (with response code 48) if other users are active on the file.

Setting LOBFILE or FILEREADONLY requires INPARALLEL=NO.

LOBFILE: LOB File Number

LOBFILE specifies the file number of the *LOB file* associated with the *base file*. This parameter is useful in combination with the FILE parameter (which is set to the base file number) to mark the two files as being in sync.

For more information, read *Large Object (LB) Files and Fields*, in *Adabas DBA Tasks Manual*.

MAXDS/ MAXNI/ MAXUI: Maximum Secondary Allocation

The maximum number of blocks per secondary extent allocation for Data Storage (MAXDS), the normal index (MAXNI), and the upper index (MAXUI). The value specified must specify blocks and be followed by a "B" (for example, MAXDS=8000B). If one of the parameters is either not specified or specifies "0B", the maximum secondary extent allocation for that component has no limit.

In all cases, however, Adabas enforces minimum secondary allocations for these parameters:

```
MAXDS=6B
MAXNI=6B
MAXUI=15B
```

If you specify a value lower than these minimum allocations, the Adabas-enforced minimum value is used.

MAXRECL: Maximum Compressed Record Length

The maximum compressed record length permitted for the file. The value specified should not be less than the current maximum record size in the specified file.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PASSWORD: File Password

This parameter is required if the file specified in the FILE parameter is password-protected.

PGMREFRESH: Program-generated File Refresh

The PGMREFRESH parameter determines whether a user program is allowed to perform a file refresh operation by issuing a special E1 command. If the parameter is not specified, the option remains in its current status: either on (YES) or off (NO).

RPLUPDATEONLY: Event Replicator Server Update Only

The `RPLUPDATEONLY` parameter can be used in the ADADBS MODFCB function to indicate whether an Adabas database file may be updated only by the Event Replicator Server as part of Adabas-to-Adabas replication or by other means as well. This parameter is optional. Valid values are "YES" or "NO". A value of "YES" indicates that the file can only be updated via Event Replicator processing; a value of NO indicates that the file can be updated by any normal means, including Event Replicator processing. There is no default; if no value is specified for the `RPLUPDATEONLY` parameter in the ADADBS MODFCB function, the value used previously for the file is used.

SYFMAXUV: Maximum MU System Field Values

The SYFMAXUV parameter can be used to specify the maximum number of values kept for a system field with the MU option during the execution of an update (A1) command (in other words, the maximum number of occurrences allowed for MU system fields during the execution of an update command). The value set for SYFMAXUV applies to all system fields in the file with the MU option. Valid values are integers from 1 through 20. The maximum value for SYFMAXUV is 20.

The internal default, if SYFMAXUV is not specified, is zero (0), which Adabas interprets to mean that there is no setting for this parameter at the file level. In this case, Adabas will assume a default of 1.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See **Syntax Checking with the TEST Parameter** for more information on using the TEST parameter in ADADBS functions.

Example

The following modifications are to be made for file 203: the Associator padding factor is set to 5, the Data Storage padding factor to 5, and the maximum Data Storage secondary extent allocation to 100 blocks.

```
ADADBS MODFCB
FILE=203,ASSOPFAC=5,DATAPFAC=5,MAXDS=100B
```

69 MUPEX: Set Maximum Count for MU and PE Fields

- Syntax 417
- Essential Parameters 417
- Optional Parameters 417
- Example 418

The MUPEX function is used to specify MU field or PE group limits for a file. This function cannot be run on files that use a PE field in a descriptor nor can it be run on system files. In addition, the maximum number of occurrences is also limited by the maximum size of a compressed record in data storage.

To increase the MU/PE count for a file that contains a PE group in a descriptor, any descriptors having to do with the PE group must first be released (using the ADADBS RELEASE function). After they have been released, the MUPECOUNT parameter can be set to "2" using the MUPEX function and the descriptors then reinverted (using the ADADBS INVERT function).

Syntax

```
ADADBS  MUPEX   FILE = file-number
                MUPECOUNT = { 1 | 2 }
                [PASSWORD = 'password' ]
                [NOUSERABEND]
                [TEST]
```

Essential Parameters

FILE: File Number

FILE specifies the file for which the maximum number of occurrences are to be set. Specify the valid file number of the file in the database for which you want MU and PE limits set.

System files cannot be changed.

MUPECOUNT: Size of the count field

MUPECOUNT sets the maximum number of MU and PE occurrences allowed in the file. Valid values are "1" or "2".

A specification of "1" indicates that no more than 191 MU fields and PE groups will occur in the file; a specification of "2" indicates that the file may contain as many as 65534 MU fields and PE groups.

To set the MUPECOUNT parameter to "1", the file must either be empty or have no MU or PE fields. To set the MUPECOUNT to "2", the file cannot contain a PE field that is part of a DE field unless the file is empty.

Optional Parameters

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PASSWORD: File Password

The password of the file for which MU and PE field counts are to be limited. This parameter is required if the file is password-protected. Specify the password between apostrophes ('). In addition, the password must provide update authority for this function to work correctly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See *Syntax Checking with the TEST Parameter* (elsewhere in this section) for more information on using the TEST parameter in ADADBS functions.

Example

In the following example, the maximum number of MU and PE field elements that can be stored in file 17 is 65534:

```
ADADBS MUPEX FILE=17,MUPECOUNT=2
```

70 NEWFIELD: Add New Field

- Essential Parameter 420
- Optional Parameters 421
- Example 422

The NEWFIELD function adds one or more fields to a file. The new field definition is added to the end of the field definition table (FDT).

Note: Although the definition of a descriptor field is independent of the record structure, note that if a descriptor field is not ordered first in a record and logically falls past the end of the physical record, the inverted list entry for that record is not generated for performance reasons. To generate the inverted list entry in this case, it is necessary to unload short, decompress, and reload the file; or use an application program to order the field first for each record of the file.

NEWFIELD cannot be used to specify actual Data Storage data for the new field; the data can be specified later using Adabas add/update or Natural commands.

When adding a field to an Adabas expanded file, the field must be added to *each individual component file* . Each NEWFIELD operation on a component file returns a message that confirms the change and condition code 4.

This utility function does not need to lock the database file for its use; this function can perform its processing in parallel with active users. This means that you do not need to set the file to read-only status to run this utility function.

```
ADADBS  NEWFIELD  FILE = file-number
                  [FNDEF = 'Adabas-field-definition' ]
                  [INPARALLEL={ YES|NO }]
                  [NOUSERABEND]
                  [PASSWORD = 'password' ]
                  [SUBFN = 'name  = parent-field (begin , end )' ]
                  [SUPFN = 'name  = { parent-field (begin, end ) } ...' ]
                  [TEST]
```

Essential Parameter

FILE: File Number

FILE specifies the file in which the field to be added is contained. The file may not be an Adabas system file.

Optional Parameters

FNDEF: Adabas Field Definition

FNDEF specifies an Adabas field (data) definition. One FNDEF statement is required for each field to be added. The syntax used in constructing field definition entries is:

```
FNDEF = 'level, name [ , length, format ] [, MU [(occurrences)] ] [ , option ] ... '
FNDEF = 'level, name [ , PE [(occurrences )] ]'
```

Each definition must adhere to the field definition syntax as described for the ADACMP utility in *FNDEF: Field and Group Definition* and *FNDEF: Periodic Group Definition* in the section entitled *Field Definition Statements*, in the ADACMP documentation elsewhere in this guide.

Note the following restrictions:

- A subdescriptor, superdescriptor, hyperdescriptor, or phonetic descriptor definition cannot be specified.
- Text information or sequence numbers are not permitted.
- If you specify an occurrence number when adding an MU or PE field, it is ignored.

The following rules apply when you set the level number in the first FNDEF statement:

1. A level number 01 is always allowed.
2. A level number of 02 or higher means that this field is to be added to an existing group. If so, the following rules apply:
   - The field can be added if the group is a normal (not periodic) group;
   - If the group is a PE group, the field can be added only if the file control block (FCB) for the file does *not* exist; that is, either the file was deleted with the KEEPFDT option, or the FDT was defined using the Adabas Online System Define FDT function, but the Define File function has not yet been run.

INPARALLEL: Execute function in parallel

When INPARALLEL=YES (the default) is specified, the function may be executed even if other users are active on the file. Adabas performs the function while all application commands against the file are temporarily quiesced and suspended. The integrity of the file targeted by the ADADBS function is maintained.

With INPARALLEL=NO, the function requires exclusive file control – that is, no applications can be executing against the file. Adabas refuses to execute the function (with response code 48) if other users are active on the file.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PASSWORD: File Password

File password. This parameter is required if the file is password-protected.

SUBFN/ SUPFN: Add Subfields or Superfields

These parameters may be used to add subfields and superfields. Each definition must adhere to the definition syntax for sub/superfields as described for the ADACMP utility. Read *COMPRESS: Compress an Adabas File*, elsewhere in this guide for more information.

TEST: Test Syntax

This parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See **Syntax Checking with the TEST Parameter** for more information on using the TEST parameter in ADADBS functions.

Example

Group AB (consisting of fields AC and AX) is to be added to file 24.

```
ADADBS   NEWFIELD    FILE=24
ADADBS               FNDEF='01,AB'
ADADBS               FNDEF='02,AC,3,A,DE,NU'
ADADBS               FNDEF='02,AX,5,P,NU'
```

71 ONLADD: Allocating New ASSO or DATA Space Dynamically

- Essential Parameters 424
- Optional Parameters 425
- Restrictions and Considerations 426
- Example 426

This section describes how you can add data sets to the Associator (ASSO) or Data Storage (DATA) space without restarting your nucleus.

In contrast to the ADADBS ADD utility function, ONLADD does not require that a nucleus restart to make the space that the new data set provides available to the current nucleus session. ADADBS determines the next data set number from the current database configuration (GCB) and opens the new ASSO or DATA data set with this number. Therefore you must be sure that the JCL for the ADADBS ONLADD run is updated accordingly. For example, if you are running ADADBS ONLADD for Data Storage and the highest data set number currently in use for Data Storage is "2", your JCL for the ADADBS ONLADD run must include DD statement "DDDATAR3" that identifies the data set name of the next, new data set.

A maximum of 99 physical extents is allowed for ASSO and DATA data sets. However, your actual real maximum could be less because the extent descriptions of all ASSO, DATA, and Data Storage Space Table (DSST) extents must fit into the general control blocks (GCBs). For example, on a DDASSOR1 with the standard 3390 device type, there can be only 78 Associator, Data Storage, and DSST extents each (or there could be more of one extent type if there is less for another).



For complete steps describing how to add a data set to the available Associator or Data Storage space without shutting down your Adabas database, read *Dynamically Increasing Associator and Data Storage Space*, in the *Adabas DBA Tasks Manual*.

Essential Parameters

One of the following parameters must be specified. Both parameters may *not* be specified in the same ADADBS ONLADD statement.

ASSOSIZE: Size of ASSO Data Set to be Added

Specify the number of cylinders to be contained in the new Associator data set. To specify blocks, append a "B" after the value.

DATASIZE: Size of DATA Data Set to be Added

Specify the number of cylinders to be contained in the new Data Storage data set. To specify blocks, append a "B" after the value.

Optional Parameters

The following parameters are optional.

ASSODEV: ASSO Data Set Device Type

Specify the device type to be used for the new Associator data set. This parameter can only be specified if the ASSOSIZE parameter is also specified.

Specify this parameter if it differs from the ADARUN DEVICE parameter.

DATADEV: DATA Data Set Device Type

Specify the device type to be used for the new Data Storage data set. This parameter can only be specified if the DATASIZE parameter is also specified.

Specify this parameter if it differs from the ADARUN DEVICE parameter.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "`utility` TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.



Note: The validity of values and variables cannot be tested; only the syntax of the specified parameters can be tested. For more information about using the TEST parameter with ADADBS functions, read *Syntax Checking with the TEST Parameter*, elsewhere in this guide.

Restrictions and Considerations

The following restrictions exist for the use of the ADADBS ONLADD function:

1. You cannot run this function while the ADARES REGENERATE database or ADARES REGENERATE FILE with CONTINUE utility functions are in progress.
2. If a nucleus starts and attempts to join an active cluster that is performing an ONLADD operation at the same time, there is a small chance that the start of the new nucleus will fail if it participates only in part of the intracluster communication for the ONLADD. The starting nucleus will get Parm-Error 61 (important fields in the GCB changed while this nucleus was starting). If this happens, restart the nucleus.
3. If an ADARES REGENERATE database needs to be run for an Adabas session in which an ONLADD operation was performed, consider the following:
   - Make sure the added ASSO or DATA data set is available -- that is, allocated and formatted. This can be done either initially or during the regenerate process at the time of the ONLADD operation.
   - ADARES will stop the regenerate process at the SYNS-82 checkpoint that was created by the ONLADD operation. Make sure the added ASSO or DATA data set is available now, if this has not already been done initially.
   - Rerun the original ADADBS ONLADD operation manually to allocate the added ASSO or DATA space to the database.
   - Restart the ADARES REGENERATE utility with the FROMCP and FROMRABN parameters referring to the SYNS-82 checkpoint of the ONLADD.

Example

The following example shows the two jobs that could be used to dynamically add a Data Storage container data set to database 123. JCL statement DDDATAR3 identifies the new Data Storage data set.

The first job creates and formats the new Data Storage data set.

```
//*
//*  Create and format a new DATA container data set
//*  (50,000 cylinders on device type 8391)
//*
//CREATE   EXEC  PGM=ADARUN
//DDDATAR3 DD  DSN=ADA.DB123.DATAR3,DISP=(NEW,CATLG),
//             SPACE=(CYL,50000),...
...
```

```
//DDCARD   DD  *
ADARUN PROG=ADAFRM,DBID=123
//DDKARTE  DD  *
ADAFRM DATAFRM SIZE=50000,NUMBER=3,DEVICE=8391
```

The second job allocates the new Data Storage data set to the database.

```
//*
//*  Dynamically allocate the new DATA data set to the database
//*
//ONLADD   EXEC  PGM=ADARUN
//DDDATAR3 DD  DSN=ADA.DB123.DATAR3,DISP=SHR
...
//DDCARD   DD  *
ADARUN PROG=ADADBS,DBID=123
//DDKARTE  DD  *
ADADBS ONLADD DATASIZE=50000,DATADEV=8391
```

Appropriate JCL statements for the new data set need to be added to the nucleus startup job and to any Adabas utility procedures you have defined.

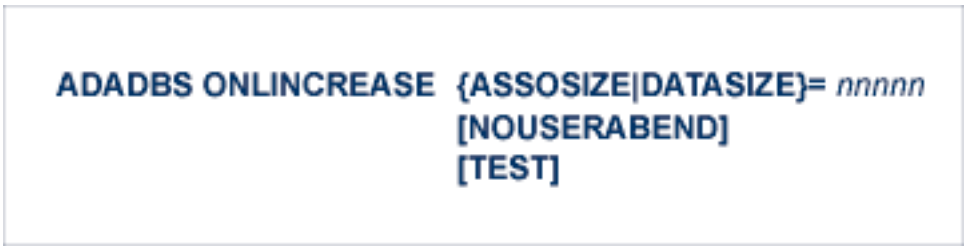
72 ONLINCREASE: Allocating Appended ASSO or DATA Space Dynamically

- Essential Parameters 430
- Optional Parameters 431
- Restrictions and Considerations 431
- Example 432

Use the ADADBS ONLINCREASE function to dynamically increase the size of the last existing ASSO or DATA data set for the database. Running this utility function will trigger Adabas to extend the data set, format the appended space, and allocate the space to the database. This makes new space available for use in the database.

In contrast to the INCREASE function, the ONLINCREASE function does not require a nucleus restart and makes the additional space available in the current nucleus session.

Other than operating system limits, Adabas has no limits on the number of times this function can be executed for the Associator. For Data Storage, Adabas limits the number of executions by the necessity to allocate associated Data Storage Space Table (DSST) space. If the extent information for all Associator, Data Storage, and DSST extents does not fit into the general control blocks (GCBs), multiple DSST extents can be combined using the ADAORD REORDB or REORASSO utility functions, which, however, make the database unavailable for normal application processing.



Essential Parameters

One of the following parameters must be specified. Both parameters may *not* be specified in the same ADADBS ONLINCREASE statement.

ASSOSIZE: Size of ASSO Data Set Space to be Added

Specify the additional number of blocks or cylinders to be added to the last Associator data set. To specify blocks, append a "B" after the value.

If ASSOSIZE is specified in units of blocks (with suffix "B") and the specified block count is not equivalent to an integral number of tracks on the disk, the ONLINCREASE function reduces the actual size of the increase to the next lower track boundary and sets condition code 4 for the ADADBS execution. The ASSO data set size is increased, just for less than originally requested. The ONLINCREASE function also prints a message, in the following format, listing the number of blocks by which the database has been increased: *dddddd* `has been increased by` *nnnn* `BLOCKS`.

DATASIZE: Size of DATA Data Set Space to be Added

Specify the additional number of blocks or cylinders to be added to the last Data Storage data set. To specify blocks, append a "B" after the value.

If DATASIZE is specified in units of blocks (with suffix "B") and the specified block count is not equivalent to an integral number of tracks on the disk, the ONLINCREASE function reduces the actual size of the increase to the next lower track boundary and sets condition code 4 for the ADADBS execution. The Data Storage data set size is increased, just for less than originally requested. The ONLINCREASE function also prints a message, in the following format, listing the number of blocks by which the database has been increased: *dddddd* `has been increased by` *nnnn* `BLOCKS`.

Optional Parameters

The following parameters are optional.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.



Note: The validity of values and variables cannot be tested; only the syntax of the specified parameters can be tested. For more information about using the TEST parameter with ADADBS functions, read *Syntax Checking with the TEST Parameter*, elsewhere in this guide.

Restrictions and Considerations

The following restrictions exist for the use of the ADADBS ONLINCREASE function:

1. You cannot run this function while the ADARES REGENERATE database or ADARES REGENERATE FILE with CONTINUE utility functions are in progress.
2. If a nucleus starts and attempts to join an active cluster that is performing an ONLINCREASE operation at the same time, there is a small chance that the start of the new nucleus will fail if

it participates only in part of the intracluster communication for the ONLINCREASE. The starting nucleus will get Parm-Error 61 (important fields in the GCB changed while this nucleus was starting). If this happens, restart the nucleus.

3. If an ADARES REGENERATE database needs to be run for an Adabas session in which an ONLINCREASE operation was performed, consider the following:
   - ADARES will stop the regenerate process at the SYNS-82 checkpoint that was created by the ONLINCREASE operation. Make sure the increased ASSO or DATA data set is available now, if this has not already been done initially.
   - Rerun the original ADADBS ONLINCREASE operation manually to allocate the added ASSO or DATA space to the database.
   - Restart the ADARES REGENERATE utility with the FROMCP and FROMRABN parameters referring to the SYNS-82 checkpoint of the ONLINCREASE.
4. Users must ensure that there is sufficient RACF authorization available in the nucleus job to allow a dataset to be extended onto a new volume.

   A RACF authorization error during ONLINCREASE could result in a nucleus abend, e.g. system 937 abend with reason code 38.

Example

The following examples show the jobs that could be used to extend the highest-numbered Associator data set (ASSOR3), format the appended space, and allocate it to the database.

If sufficient space is available on the volume or volumes currently comprising the data set, follow example 1. If additional volumes are required to satisfy the online increase request, follow example 2 (for non DFSMS-managed data sets) or 3 (for DFSMS-managed data sets).

Example 1: ONLINCREASE not requiring additional volumes

400 cylinders are to be added to an Associator data set. The control statement for the ONLINCREASE function would be:

```
ADADBS ONLINCREASE ASSOSIZE=400
```

Example 2: ONLINCREASE requiring additional volumes (not DFSMS-managed)

400 cylinders are to be added to the final ASSO data set (ASSOR3). The final ASSO data set is a single-volume Associator data set on volume VOL001. For the online increase, the data set needs to extend to volume VOL002.

Before running ADADBS ONLINCREASE, run the following job steps to uncatalog the data set and then recatalog it with the new volume:

```
//UNCATLG  EXEC PGM=IEFBR14
//DDASSOR3  DD  DSN=ADA.DB123.ASSOR3,DISP=(SHR,UNCATLG)
//*
//RECATLG  EXEC PGM=IEFBR14
//DDASSOR3  DD  DSN=ADA.DB123.ASSOR3,DISP=(SHR,CATLG),
//   UNIT=SYSDA,VOL=SER=(VOL001,VOL002)
```

The control statement for the ONLINCREASE function would be:

```
ADADBS ONLINCREASE ASSOSIZE=400
```

Example 3: ONLINCREASE requiring additional volumes (DFSMS-managed)

400 cylinders are to be added to the final ASSO data set (ASSOR3). For the increase, the data set needs to extend onto another candidate volume belonging to the DFSMS Storage Group.

Before running ADADBS ONLINCREASE, run the following job step to add further candidate volumes to the catalog entry for the data set:

```
//ADDVOL EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
ALTER ADA.DB123.ASSOR3 ADDVOLUMES(*)
/*
```

The control statement for the ONLINCREASE function would be:

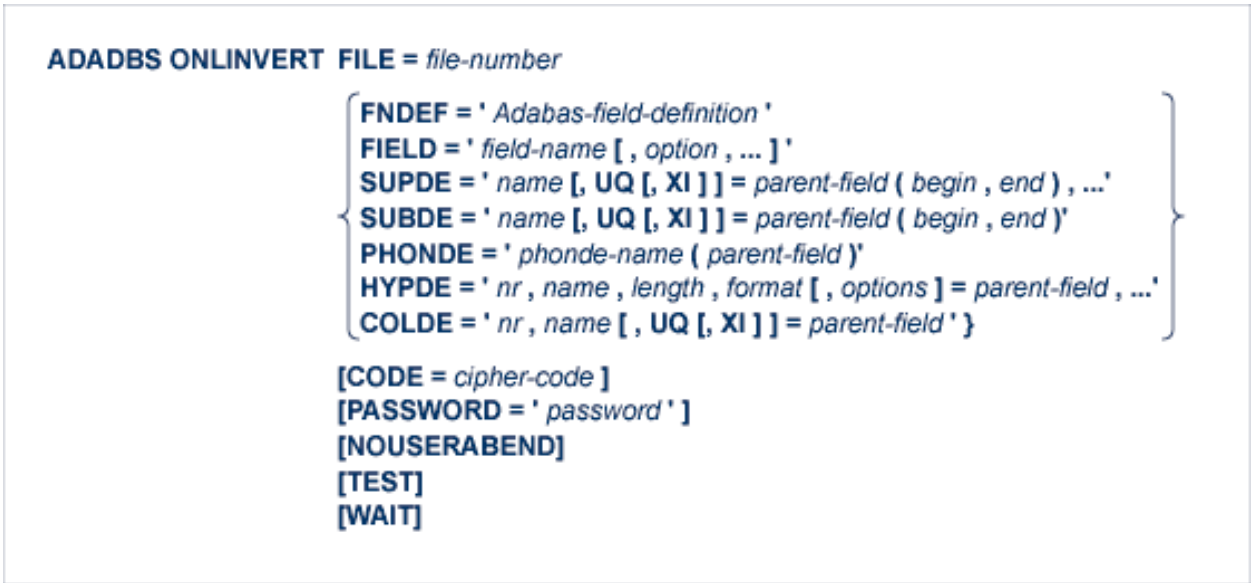
```
ADADBS ONLINCREASE ASSOSIZE=400
```

73 ONLINVERT: Start Online Invert Process

- Essential Parameters 436
- Optional Parameters 438
- Example 438

The ONLINVERT function starts an online invert process.

Note: If the online invert is stopped or suspended, the nucleus releases the partially inverted descriptor when it next starts up. If the ONLINVERT was processing an expanded (chained) file at the time processing was suspended, the nucleus releases the partially inverted descriptor on all component files of the expanded file.



Essential Parameters

FILE: File Number

File is the number of the file for which the new descriptor is to be created. If a component file of an expanded file chain is specified, the descriptor is added to all component files of that chain.

FIELD/ SUBDE/ SUPDE/ PHONDE/ HYPDE/ COLDE: Define Descriptor

Exactly one of these parameters or the FNDEF parameter must be used to define the type of descriptor to be inverted. Only one descriptor per file can be inverted at a time using the online invert function.

Use the FIELD parameter to define a field as descriptor; use the COLDE parameter for a collation descriptor; the HYPDE parameter for a hyperdescriptor; PHONDE for a phonetic descriptor; SUBDE for a subdescriptor; and SUPERDE for a superdescriptor. These fields and the FNDEF parameter are mutually exclusive.

FIELD specifies an existing field to be inverted. The field may be an elementary or multiple-value field and may be contained within a periodic group (unless the field is defined with the FI option).

If the descriptor is to be unique, specify "UQ" following the field name. A field in a periodic group cannot be defined as a unique descriptor. If the uniqueness of the descriptor is to be determined with the index (occurrence number) excluded, specify "XI" as well.

When inverting a sub- or superfield, the respective SUBDE or SUPDE parameter must specify the same parent fields that were specified when the field was created; otherwise, an error occurs. Begin and end values are taken from the original field definitions.

If a parent field with the NU option is specified, no entries are made in the inverted list for those records containing a null value for the field. For super- and hyperdescriptors, this is true regardless of the presence or absence of values for other descriptor elements.

If a parent field is not initialized and logically falls past the end of the physical record, the inverted list entry for that record is not generated for performance reasons. To generate the inverted list entry in this case, it is necessary to unload short, decompress, and reload the file; or use an application program to initialize the field for each record of the file.

See the **ADACMP utility** description for detailed information about the individual descriptor syntax, subparameter values, and coding.

FNDEF: Adabas Field Definition for New Descriptor

FNDEF specifies an Adabas field (data) definition for a new descriptor. One FNDEF statement or a FIELD, SUBDE, SUPDE, PHONDE, HYPDE, or COLDE parameter is required for each field to be inverted. These parameters are mutually exclusive.

The syntax used in constructing field definition entries in FNDEF is:

FNDEF = '*level*, *name* **[** , *length*, *format* **] [, MU [(***occurrences***)]] [** , *option* **] ... '**

FNDEF = '*level*, *name* **[, PE [(***occurrences* **)]]'**

Each definition must adhere to the field definition syntax as described for the ADACMP utility in ***FNDEF: Field and Group Definition*** and ***FNDEF: Periodic Group Definition*** in the section entitled ***Field Definition Statements***, in the ADACMP documentation elsewhere in this guide.

For example, the following ADADBS INVERT statement makes the NT field a descriptor:

```
ADADBS ONLINVERT FILE=201,FNDEF='01,NT,7,P,NU,DT=E(NATTIME)'
```

Optional Parameters

CODE: Cipher Code

If the file specified with the FILE parameter is ciphered, an appropriate cipher code must be supplied using the CODE parameter.

PASSWORD: File Password

If the file specified with the FILE parameter is security-protected, the file's password must be supplied using the PASSWORD parameter.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

WAIT: Wait for End of Process

Specify WAIT if ADADBS is to wait for the end of the online process before proceeding either with the next function or with termination.

If WAIT is not specified, ADADBS proceeds immediately after initiating the online process.

Example

Initiate an online process to make field AA of file 10 a descriptor, without waiting for the end of this process.

```
ADADBS ONLINVERT FILE=10,FIELD=AA
```

74 ONLREORFASSO: Start Online Reorder Associator for Files

- Essential Parameters 440
- Optional Parameters 441
- Example 441

The ONLREORFASSO function starts an online process to reorder the Associator of specified files.

Notes:

1. The online reorder process does not change the existing file extents but only reorganizes the file's index within these extents.
2. The online index reorder process does not move index elements out of blocks that are full (according to the Asso padding factor); it only moves elements into blocks that are not full.
3. Released index blocks are not put into the unused RABN chain.
4. This function is not available in the Adabas Cluster Services or Adabas Parallel Services environments.

```
ADADBS  ONLREORFASSO  FILE = file-number
                      [ASSOPFAC = asso-padding-factor]
                      [PASSWORD = password ]
                      [NOUSERABEND]
                      [TEST]
                      [WAIT]
```

Essential Parameters

FILE: File Number

FILE specifies the file to which the parameters that follow in the statement sequence apply.

Several files and their related parameters may be specified within one ONLREORFASSO operation. In this case, the files are reordered in the specified sequence.

If a component file of an Adabas expanded file is specified, only that file's Associator is reordered; this has no adverse effect on the other component files.

The Adabas checkpoint or security file number must not be specified.

Optional Parameters

ASSOPFAC: Associator Padding Factor

ASSOPFAC defines the Associator block padding factor, which is the percentage of each Associator block *not* used during the reorder process. Specify a value in the range 1-90. The number of bytes free after padding must be greater than the largest descriptor value plus 10.

If this parameter is omitted, the current padding factor in effect for the file is used.

PASSWORD: File Password

If the file is password-protected, use this parameter to specify the password.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

WAIT: Wait for End of Process

Specify WAIT if ADADBS is to wait for the end of the online process before proceeding either with the next function or with termination.

If WAIT is not specified, ADADBS proceeds immediately after initiating the online process.

Example

Initiate an online process that reorders the Associator of file 10 first and then file 11. The Associator padding factor of file 11 is to be 5 percent.

```
ADADBS ONLREORFASSO FILE=10
ADADBS              FILE=11,ASSOPFAC=5
```

75 ONLREORFDATA: Start Online Reorder Data for Files

- Essential Parameters 444
- Optional Parameters 445
- Example 446

The ONLREORFDATA function starts an online process to reorder the Data Storage of specified files.

Notes:

1. The online reorder process does not change the existing file extents but only reorganizes the file's Data Storage records within these extents.
2. This function is not available in the Adabas Cluster Services or Adabas Parallel Services environments.

```
ADADBS  ONLREORFDATA  FILE = file-number
                      [DATAPFAC =  data-padding-factor ]
                      [SORTSEQ = { ISN | de-name } ]
                      [PASSWORD = password ]
                      [NOUSERABEND]
                      [TEST]
                      [WAIT]
```

Essential Parameters

FILE: File Number

FILE specifies the file to which the parameters that follow in the statement sequence apply.

Several files and their related parameters may be specified within one ONLREORFDATA operation. In this case, the files are reordered in the specified sequence.

If a component file of an Adabas expanded file is specified, only that file's Data Storage is reordered; this has no adverse effect on the other component files.

The Adabas checkpoint or security file number must not be specified.

Optional Parameters

DATAPFAC: Data Storage Padding Factor

DATAPFAC specifies the Data Storage padding factor. The number specified represents the percentage of each Data Storage block that remains unused when the file is reordered. A value in the range 1-90 may be specified (see **ADALOD utility** for additional information about setting and using the Data Storage padding factor).

If this parameter is omitted, the current padding factor in effect for the file is used.

SORTSEQ: File Reordering Sequence

SORTSEQ determines the sequence in which the file is processed. If this parameter is omitted, the records are processed in physical sequence.



Note: Records within a single Data Storage block are not sorted according to the specified sequence.

If a descriptor is specified, the file is processed in the logical sequence of the descriptor values. *Do not* use a hyperdescriptor, a phonetic descriptor, a multiple-value field, or a descriptor contained in a periodic group.

If ISN is specified, the file is processed in ascending ISN sequence.

PASSWORD: File Password

If the file is password-protected, use this parameter to specify the password.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

WAIT: Wait for End of Process

Specify WAIT if ADADBS is to wait for the end of the online process before proceeding either with the next function or with termination.

If WAIT is not specified, ADADBS proceeds immediately after initiating the online process.

Example

Initiate an online process that reorders the Data Storage of file 10 first, and then file 11. The Data Storage padding factor of file 11 is to be 5 percent.

```
ADADBS ONLREORFDATA FILE=10
ADADBS              FILE=11,DATAPFAC=5
```

76 ONLREORFILE: Start Online Reorder Associator and Data for Files

- Essential Parameters 448
- Optional Parameters 449
- Example 450

The ONLREORFILE function starts an online process to reorder the Associator and Data Storage of specified files.

Notes:

1. The online reorder process does not change the existing file extents but only reorganizes the file's index and Data Storage records within these extents.
2. This function is not available in the Adabas Cluster Services or Adabas Parallel Services environments.
3. Released index blocks are not put into the unused RABN chain.

```
ADADBS  ONLREORFILE     FILE = file-number
                        [ASSOPFAC = asso-padding-factor]
                        [DATAPFAC = data-padding-factor ]
                        [SORTSEQ = { ISN | de-name } ]
                        [PASSWORD = password ]
                        [NOUSERABEND]
                        [TEST]
                        [WAIT]
```

Essential Parameters

FILE: File Number

FILE specifies the file to which the parameters that follow in the statement sequence apply.

Several files and their related parameters may be specified within one ONLREORFILE operation. In this case, the files are reordered in the specified sequence.

If a component file of an Adabas expanded file is specified, only that file's Associator and Data Storage is reordered; this has no adverse effect on the other component files.

The Adabas checkpoint or security file number must not be specified.

Optional Parameters

ASSOPFAC: Associator Padding Factor

ASSOPFAC defines the new Associator block padding factor, which is the percentage of each Associator block *not* used during the reorder process. Specify a value in the range 1-90. The number of bytes free after padding must be greater than the largest descriptor value plus 10.

If this parameter is omitted, the current padding factor in effect for the file is used.

DATAPFAC: Data Storage Padding Factor

DATAPFAC specifies the new Data Storage padding factor. The number specified represents the percentage of each Data Storage block that remains unused when the file is reordered. A value in the range 1-90 may be specified (see the **ADALOD utility** for additional information about setting and using the Data Storage padding factor).

If this parameter is omitted, the current padding factor in effect for the file is used.

SORTSEQ: File Reordering Sequence

SORTSEQ determines the sequence in which the file is processed. If this parameter is omitted, the records are processed in physical sequence.

Note: Records within a single Data Storage block are not sorted according to the specified sequence.

If a descriptor is specified, the file is processed in the logical sequence of the descriptor values. *Do not* use a hyperdescriptor, a phonetic descriptor, a multiple-value field, or a descriptor contained in a periodic group.

If ISN is specified, the file is processed in ascending ISN sequence.

PASSWORD: File Password

If the file is password-protected, use this parameter to specify the password.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

WAIT: Wait for End of Process

Specify WAIT if ADADBS is to wait for the end of the online process before proceeding either with the next function or with termination.

If WAIT is not specified, ADADBS proceeds immediately after initiating the online process.

Example

Initiate an online process that reorders the Associator and Data Storage of file 10 first, and then file 11. The Associator padding factor of file 10 is to be 5 percent; the Data Storage padding factor of file 11 is to be 10 percent.

```
ADADBS ONLREORFILE FILE=10,ASSOPFAC=5
ADADBS             FILE=11,DATAPFAC=10
```

77 OPERCOM: Issue Adabas Operator Commands

- Using OPERCOM Commands in Cluster Environments 452
- Optional Parameters 452
- Operator Commands 453

The OPERCOM function issues operator commands to the Adabas nucleus.

In an Adabas cluster environment, OPERCOM commands can be directed to a single cluster nucleus or to all active nuclei in the cluster. If a particular nucleus is not specified, the command defaults to the local nucleus.

Adabas issues a message to the operator, confirming command execution.

```
ADADBS  OPERCOM  operator-command
                 [NOUSERABEND]
                 [NUCID = { nuc-id | 0 } ]
                 [TEST]
```

In this section, the discussion of the individual operator commands follows the discussion of the optional parameters, since some of the operator commands behave differently when issued in an Adabas cluster environment.

Using OPERCOM Commands in Cluster Environments

Some ADARUN parameters are *global parameters*; that is, they must have the same values for all nuclei in a cluster. Of these, some are set at session initialization and cannot be changed. Others can be modified on a running system. OPERCOM commands that change these modifiable global parameter values are handled in a special way in cluster environments.

If an Adabas cluster nucleus changes one or more global parameters, that nucleus acquires a *parameter change lock*, makes the changes in its local parameter area, informs the other cluster nuclei of the changes and waits for a reply. The other cluster nuclei make the changes in their own local parameter areas and send an acknowledgment message.

Optional Parameters

GLOBAL: Operate Across All Active Cluster Nuclei

Six OPERCOM commands use the GLOBAL option to operate across all active nuclei in a cluster: ADAEND, CANCEL, FEOFAL, FEOFCL, FEOFPL, and HALT. For example:

```
ADADBS OPERCOM ADAEND, GLOBAL
```

All other OPERCOM commands use the NUCID=0 option to operate across all active nuclei in a cluster.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

NUCID: Cluster Nucleus ID

Any nucleus running in an Adabas nucleus cluster is allowed to run Adabas utilities such as ADADBS.

With certain exceptions, the NUCID parameter allows you to direct the ADADBS OPERCOM commands to a particular nucleus in the cluster for execution, just as though the command had been issued by a locally run ADADBS OPERCOM operation. You can route most OPERCOM commands to all nuclei in a cluster by specifying NUCID=0.

If NUCID is not specified in a cluster environment, the command is routed to the local nucleus.



Note: For ADADBS OPERCOM and Adabas Online System (AOS), a zero value for the NUCID parameter indicates that the command applies to all nuclei in the cluster (global). A nonzero value for the NUCID parameter indicates that the command applies only to the cluster nucleus specified.

TEST: Test Syntax

This parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; nor the validity of values and variables. See **Syntax Checking with the TEST Parameter** for more information on using the TEST parameter in ADADBS functions.

Operator Commands

The following Adabas operator commands can be used in the ADADBS OPERCOM utility function:

ADAEND Command



This command terminates an Adabas session normally. No new users are accepted after this command has been issued. ET logic updating is continued until the end of the current logical transaction for each user. After all activity has been completed as described above, the Adabas session is terminated.

In nucleus cluster environments, the GLOBAL option can be used to terminate the Adabas session in all active cluster nuclei.

ALOCKF Command



Lock a file in advance to ensure that an EXU, EXF, or UTI user will have exclusive control of the specified file. The advance-lock prevents new transactions from using the file. Once all current users have stopped using the file, the exclusive-control user has the lock. Until then, the exclusive-control user must wait.

To remove the advance lock without running the utility, see the RALOCKF command.

This command is not available in single user mode or for a read-only nucleus. It is available in cluster and non-cluster environments.

The following key points should be noted about advance-locks on files:

1. An advance-lock can be set while a file is being used.
2. A command requesting exclusive control (UTI, EXF, or EXU) over an advance-locked file will wait in the command queue until all other users stop using the file before it starts processing the file.
3. Advance-locks are automatically removed when a user gets exclusive control over the file. However, if a file is locked (via the LOCKF, LOCKU, or LOCKX commands), the locks are not removed when a user gets exclusive control over the file. (Locks must be explicitly removed, whereas advance-locks are automatically removed.)
4. Adabas will reject an advance-lock on a file that is already locked (via the LOCKF, LOCKU, LOCKX or ALOCKF commands) but will accept a lock request on an advance-locked file.
5. To ensure you have uninterrupted exclusive control over a file in a situation where you have multiple steps to run that require uninterrupted exclusive control while all steps have

been processed, use a combination of advance-locking the file (ALOCKF), stopping all users of the file (STOPF), and locking the file (LOCKU). An example of this is given later in this section.

6. In the case of expanded files, an ALOCKF command is applied to the anchor file (representing the entire expanded file chain).
7. In a cluster environment, advance-locks are effective in all nuclei of the cluster.

Simple Example

In the following example, issuing the ALOCKF request to advance-lock file 32 ensures that file 32 will be available so the ADALOD UPDATE function can take exclusive control (via a UTI request) of the file for its processing:

```
ADADBS OPERCOM ALOCKF=32
ADALOD UPDATE FILE=32
```

Adabas processing proceeds in the following manner for these utility functions:

1. When the ADADBS OPERCOM ALOCKF request is submitted, file 32 is marked as advance-locked.
2. If there are any active users of file 32, the ADALOD UTI request cannot be granted immediately and will wait for the active users to end their transactions or sessions. Active users continue to issue commands against file 32. However, requests by new users for file 32 are rejected because of the advance-lock on the file.
3. When all active users of file 32 have ended their transactions or sessions, the ADALOD UTI request for exclusive control can be granted. Once exclusive control is established, ADALOD UPDATE processing can occur.

   As part of the successful execution of the ADALOD UTI request, the advance-lock is removed from the file. However, because ADALOD processing now has exclusive control of file 32, other users still cannot access it.

To accelerate the process and limit the wait time for the ADALOD UTI request, you can simply stop all active users of the file by force using the STOPF operator command:

```
ADADBS OPERCOM ALOCKF=32
ADADBS OPERCOM STOPF=32
ADALOD UPDATE FILE=32
```

In this case, the STOPF command will cause the nucleus to back out and stop users of file 32 before the ADALOD UTI request is granted. In addition, the advance-lock request specified by the ALOCKF command will prevent new users from accessing the file until the ADALOD UTI request for exclusive control is processed.

More Complex Example

An advance-lock set by ALOCKF cannot guarantee that multiple job steps in a series get uninterrupted exclusive control over a file, as the advance-lock is removed when the first step obtains exclusive control. Suppose an installation wants to run the following utility sequence:

```
ADAULD UNLOAD FILE=45
ADADBS REFRESH FILE=45
ADALOD UPDATE FILE=45
```

An ALOCKF request to advance-lock file 45 in this case would only work for the ADAULD UNLOAD function, because the ADAULD EXU request for exclusive control of file 45 would remove the advance-lock. If there are active users who try to issue commands against file 45, there is a chance that one of them will execute a command between the UNLOAD and REFRESH steps, or between the REFRESH and UPDATE steps. Such a user may also prevent the REFRESH or UPDATE step from obtaining exclusive control of file 45.

To ensure you have uninterrupted exclusive control over the file in this situation, use a combination of advance-locking the file (ALOCKF), stopping all users of the file (STOPF), and locking the file (LOCKU):

```
ADADBS OPERCOM ALOCKF=45
ADADBS OPERCOM STOPF=45
ADADBS OPERCOM LOCKU=45
ADAULD UNLOAD FILE=45
ADADBS REFRESH FILE=45
ADALOD UPDATE FILE=45
ADADBS OPERCOM UNLOCKU=45
```

In this example, Adabas processing proceeds in the following manner:

1. When the ADADBS OPERCOM ALOCKF request is submitted, file 45 is marked as advance-locked.

   The ADADBS OPERCOM STOPF request causes the nucleus to back out and stop users of file 45. (This step is optional.)

   The ADADBS OPERCOM LOCKU request locks the file more permanently than the ALOCKF request since the LOCKU lock will stay in effect until it is explicitly released.
2. If there are any active users updating file 45, the ADAULD EXU request cannot be granted immediately and will wait for the update users to end their transactions or sessions. Active users may continue to issue commands against file 45. However, requests by new users for file 45 are rejected because of the advance-lock on the file.
3. When all active users of file 45 have ended their transactions or sessions, the ADAULD request for exclusive-update (EXU) control can be granted. Once exclusive-update control is established, ADAULD UNLOAD processing can occur.

As part of the successful execution of the ADAULD EXU request, the advance-lock is removed from the file. When ADAULD completes processing, it releases the EXU control of file 45. However, during and after the ADAULD execution, the LOCKU lock ensures that other users still cannot access the file.

4. The ADADBS utility will issue a UTI request for exclusive control of file 45, which will be granted. ADADBS REFRESH processing will then occur. When it completes, ADADBS will release exclusive control of file 45. However, the LOCKU lock ensures that other users still cannot access it.
5. The ADALOD utility will issue a UTI request for exclusive control of file 45, which will be granted. ADALOD UPDATE processing will then occur. When it completes, ADALOD will release exclusive control of file 45. However, the LOCKU lock ensures that other users still cannot access it.
6. The ADADBS OPERCOM UNLOCKU request explicitly unlocks file 45, making it available for other users.

ASSOSPACEWARN Command

ASSOSPACEWARN=*threshold1*[,*inc1*[,*threshold2*[,*inc2*[,*threshold3*[,*inc3*]]]]]

Use the `ASSOSPACEWARN` command to modify the setting of the `ASSOSPACEWARN` ADARUN parameter while the database is running. ASSOSPACEWARN specifies up to three pairs of threshold percentages and percentage change increments, expressed as percentages of the total Associator (ASSO) portion of the database. When ASSO space use reaches a specified threshold percentage, a space usage message is issued. In addition, when the ASSO space use changes by the increment specified for that threshold, additional space usage messages are issued.

Note: For the purposes of database space notifications, the space used in the Adabas database comprises all database blocks that have been assigned either to some common purpose (for example, DSST, PPT, DLOG area) or to a file in the database. Only space listed in the free space table (FST) is not calculated as used. In particular, free space allocated for some file but not yet used is calculated as used.

The following specifications can be made in the ASSOSPACEWARN command:

Parameter	Description
threshold#	A threshold, expressed as a percentage of the total Associator portion of the database. When ASSO space use exceeds or falls below any of the thresholds set, a space usage message is issued. The default is zero (0). If zero is specified for the first threshold setting, no space messages are issued. When more than one threshold is specified, the second threshold specification must be greater than the first. Likewise, the third threshold specification (if made) must be greater than the second.
inc#	An increment percentage, expressed as a percentage of the total ASSO portion of the database. When ASSO space use changes (increases or decreases) from the associated threshold by the amount given in this increment percentage, another space usage message is issued. The default is five percent (5).

Up to three threshold-increment pairs can be specified.

AUDCONNECT Command

```
ADADBS OPERCOM AUDCONNECT = { dbid | ALL }
```

Use this command to dynamically force a connection attempt to either a specific Auditing Server or Adabas database ID, or to all related Auditing Server or Adabas database IDs.

One of the parameters of the AUDCONNECT command must be specified. There is no default. The parameters are described in the following table:

Parameter	Description
ALL	Forces a reconnection attempt with ALL known Auditing Server or Adabas database IDs.
dbid	The specified Auditing Server or Adabas database ID.

AUDITLOG Command

```
AUDITLOG = {CONSOLE}
```

Use the `AUDITLOG` command to modify the setting of the AUDITLOG ADARUN parameter while the database is running. Valid option is "CONSOLE". Once AUDITLOG has been activated with "CONSOLE", it cannot be changed to "NO" without cycling the nucleus.

In the following example, issuing the AUDITLOG requests activates the AUDITLOG to start writing all utility, and AOS audit data to the console.

```
ADADBS OPERCOM AUDITLOG=CONSOLE
```

AUTOINCASSOSIZE Command



Use the `AUTOINCASSOSIZE` command to modify the setting of the `AUTOINCASSOSIZE` ADARUN parameter while the database is running. AUTOINCASSOSIZE specifies the size, in cylinders or blocks, by which the highest-numbered (last-defined) Associator (ASSO) data set will be increased by an automatic database increase process.

This parameter only has an effect if the AUTOINCASSOTHRESHOLD setting has a non-zero value. A numeric value followed by the letter B indicates the value specifies a number of blocks, which is then rounded up to the next highest amount that fills the final track or other platform-dependent minimum space allocation unit. A numeric value without the suffix B indicates the increment is in cylinders.

If an automatic database increase process is started to increase the ASSO component of the database, the last ASSO container data set will be increased by the number of cylinders or blocks specified by AUTOINCASSOSIZE. (The number of cylinders or blocks added may be less, however, if the limit set by the AUTOINCASSOTOTAL parameter is reached.)

If the last ASSO container data set cannot be increased by the specified size, but only by a lesser amount, the automatic increase process will increase the data set by the amount of space that it can get. Note that one data set can span at most 59 volumes with up to 16 extents on each volume. When these limits are reached, the database should be increased manually by adding a new container data set with the ADADBS ONLADD utility function.

When 4-byte RABNs are used, valid values for this parameter range from zero (0) to 2147483647 cylinders or from zero (0) to 2147483647B blocks. When 3-byte RABNs are used, the limits range from zero (0) to 16775215 blocks or the equivalent in cylinders. A value of "0" or "0B" means that no automatic increase will be performed for ASSO space. The default value for this parameter is zero, which functionally disables auto-increase for ASSO space.

If AUTOINCASSOSIZE is set to a small value, the automatic database increase process may be initiated multiple times and cause the container allocation to reach platform-dependent limits on extents and volumes. Once that occurs, there can be no more auto-increases until the container allocations are consolidated, which requires stopping all nuclei using the container. Alternatively, a manual increase can be used to add another container without stopping the Adabas nucleus.

AUTOINCASSOTHRESHOLD Command



Use the AUTOINCASSOTHRESHOLD command to modify the setting of the AUTOINCASSOTHRESHOLD ADARUN parameter while the database is running. AUTOINCASSOTHRESHOLD specifies the Associator space threshold at which Adabas automatically initiates a database increase process. This threshold is expressed as a percentage of used Associator (ASSO) space to total Associator space available in the database, ASSO space is considered used if it is not listed as free in the Free Space Table (FST).

If the used Associator space exceeds the percentage defined by the specified AUTOINCASSO-THRESHOLD setting, Adabas automatically initiates a database increase process for ASSO space.

Note that the database increase process may take some time, the length of which is related to the size by which the database is increased (the value of the AUTOINCASSOSIZE setting). The AUTOINCASSOTHRESHOLD setting should be configured so that the increase process is given sufficient time to complete before the database fills up, assuming a reasonable space consumption rate.

Once a database increase process has been initiated, another automatic increase for ASSO space may be started only after the first process has finished and the ASSO space usage again or still exceeds the specified threshold.

Valid values for this command are either zero (0) or integers ranging from 50 through 100. A value of zero means that no automatic increase will be performed for ASSO space. A value of 100 means that an automatic increase process will be initiated only when the ASSO space grows full (i.e., no free space is left at all). The default value is zero, which functionally disables auto-increase processes for ASSO space.

AUTOINCASSOTOTAL Command



Use the AUTOINCASSOTOTAL command to modify the setting of the AUTOINCASSOTOTAL ADARUN parameter while the database is running. AUTOINCASSOTOTAL specifies the maximum total size, in cylinders or blocks, of the entire Associator space of the database that is not to be exceeded by automatic database increase processes.

A numeric value followed by the letter B indicates the value specifies a number of blocks, which is then rounded up to the next highest amount that fills the final track or other platform-dependent minimum space allocation unit. A numeric value without a suffix B indicates the increment is in cylinders.

An automatic database increase process for ASSO space will not exceed the size specified by the AUTOINCASSOTOTAL setting. This setting can be used to limit the total amount of disk space used for the database, in order to protect the disk space remaining available for other databases and other users of the system. Note that this setting limits only automatic increases by Adabas; it does not limit increasing the database manually using the ADADBS ADD, INCREASE, ONLADD or ONLINCREASE utility functions.

AUTOINCASSOTOTAL specifies the number of cylinders or blocks beyond which automatic database increase processes will not extend the ASSO space of the database. If the AUTOINCASSOTOTAL setting is less than the current number of cylinders in the ASSO space plus the AUTOINCASSOSIZE value, an automatic increase will increase the ASSO space only by the difference between the AUTOINCASSOTOTAL value and the current size – that is, by less than the increase requested by the AUTOINCASSOSIZE setting. If AUTOINCASSOTOTAL is not greater than the current number of cylinders in ASSO, no further automatic increases will be performed for ASSO space.

When 4-byte RABNs are used, valid values for this parameter range from zero (0) to 2147483647 cylinders or from zero (0) to 2147483647B blocks. When 3-byte RABNs are used, the limits range from zero (0) to 16775215 blocks or the equivalent in cylinders. A value of "0" or "0B" means that no automatic increase will be performed for ASSO space. The default value for this parameter is zero, which functionally disables auto-increase for ASSO space.

The total size of ASSO space may be constrained to less than the AUTOINCASSOTOTAL value for either of the following reasons.

- There may not be enough storage resources to extend to the specified limit.
- When using 4-byte RABNs, a component can have at most 2147483647 blocks, and for 3-byte RABNs the limit is 16777215 blocks.

AUTOINCDATASIZE Command



Use the `AUTOINCDATASIZE` command to modify the setting of the `AUTOINCDATASIZE` ADARUN parameter while the database is running. AUTOINCDATASIZE specifies the size, in cylinders or blocks, by which the highest-numbered (last-defined) Data Storage (DATA) data set will be increased by an automatic database increase process.

This parameter only has an effect if the AUTOINCDATATHRESHOLD has a non-zero value. A numeric value followed by the letter B indicates the value specifies a number of blocks, which is then rounded up to the next highest amount that fills the final track or other platform-dependent minimum space allocation unit. A numeric value without the suffix B indicates the increment is in cylinders.

If an automatic database increase process is started to increase the DATA component of the database, the last DATA container data set will be increased by the number of cylinders or blocks specified by AUTOINCDATASIZE. (The number of cylinders or blocks added may be less, however, if the limit set by the AUTOINCDATATOTAL parameter is reached.)

If the last ASSO container data set cannot be increased by the specified size, but only by a lesser amount, the automatic increase process will increase the data set by the amount of space that it can get. Note that one data set can span at most 59 volumes with up to 16 extents on each volume. When these limits are reached, the database should be increased manually by adding a new container data set with the ADADBS ONLADD utility function.

When 4-byte RABNs are used, valid values for this parameter range from zero (0) to 2147483647 cylinders or from zero (0) to 2147483647B blocks. When 3-byte RABNs are used, the limits range from zero (0) to 16775215 blocks or the equivalent in cylinders. A value of "0" or "0B" means that no automatic increase will be performed for DATA space. The default value for this parameter is zero, which functionally disables auto-increase for DATA space.

If AUTOINCDATASIZE is set to a small value, the automatic database increase process may be initiated multiple times and cause the container allocation to reach platform-dependent limits on extents and volumes. Once that occurs, there can be no more auto-increases until the container allocations are consolidated, which requires stopping all nuclei using the container. Alternatively, a manual increase can be used to add another container without stopping the Adabas nucleus.

AUTOINCDATATHRESHOLD Command



Use the `AUTOINCDATATHRESHOLD` command to modify the setting of the `AUTOINCDATATHRESHOLD` ADARUN parameter while the database is running. AUTOINCDATATHRESHOLD specifies the Data Storage space threshold at which Adabas automatically initiates a database increase process. This threshold is expressed as a percentage of used Data Storage space to total Data Storage space available in the database. DATA space is considered used if it is not listed as free in the Free Space Table (FST).

If the used Data Storage space exceeds the percentage defined by the specified AUTOINCDATATHRESHOLD setting, Adabas automatically initiates a database increase process for DATA space.

Note that the database increase process may take some time, the length of which is related to the size by which the database is increased (the value of the AUTOINCDATASIZE setting). The AUTOINCDATATHRESHOLD setting should be configured so that the increase process is given sufficient time to complete before the database fills up, assuming a reasonable space consumption rate.

Once a database increase process has been initiated, another automatic increase for DATA space may be started only after the first process has finished and the DATA space usage again or still exceeds the specified threshold.

Valid values for this command are either zero (0) or integers ranging from 50 through 100. A value of zero means that no automatic increase will be performed for DATA space. A value of 100 means that an automatic increase process will be initiated only when the DATA space grows full (i.e., no free space is left at all). The default value is zero, which functionally disables auto-increase processes for DATA space.

AUTOINCDATATOTAL Command



Use the `AUTOINCDATATOTAL` command to modify the setting of the `AUTOINCDATATOTAL` ADARUN parameter while the database is running. AUTOINCDATATOTAL specifies the maximum total size, in cylinders or blocks, of the entire Data Storage space of the database that is not to be exceeded by automatic database increase processes.

A numeric value followed by the letter B indicates the value specifies a number of blocks, which is then rounded up to the next highest amount that fills the final track or other platform-dependent minimum space allocation unit. A numeric value without a suffix B indicates the increment is in cylinders.

An automatic database increase process for DATA space will not exceed the size specified by the AUTOINCDATATOTAL setting. This setting can be used to limit the total amount of disk space used for the database, in order to protect the disk space remaining available for other databases and other users of the system. Note that this setting limits only automatic increases by Adabas; it does not limit increasing the database manually using the ADADBS ADD, INCREASE, ONLADD or ONLINCREASE utility functions.

AUTOINCDATATOTAL specifies the number of cylinders or blocks beyond which automatic database increase processes will not extend the DATA space of the database. If

AUTOINCDATATOTAL setting is less than the current DATA size plus the AUTOINCDATASIZE value, an automatic increase will increase the DATA space only by the difference between the AUTOINCDATATOTAL value and the current size – that is, by less than the increase requested by the AUTOINCDATASIZE setting. If the AUTOINCDATATOTAL setting is not greater than the current DATA size, no further automatic increases will be performed for DATA space.

When 4-byte RABNs are used, valid values for this parameter range from from zero (0) to 2147483647 blocks or the equivalent in cylinders. When 3-byte RABNs are used, the limits range from zero (0) to 16775215 blocks or the equivalent in cylinders. A value of "0" or "0B" means that no automatic increase will be performed for DATA space. The default value for this parameter is zero, which functionally disables auto-increase for DATA space.

The total size of DATA space may be constrained to less than the AUTOINCDATATOTAL value for either of the following reasons.

- There may not be enough storage resources to extend to the specified limit.
- When using 4-byte RABNs, a component can have at most 2147483647 blocks, and for 3-byte RABNs the limit is 16777215 blocks.

CANCEL Command

```
CANCEL [, GLOBAL ]
```

Cancel the Adabas session immediately. All command processing is immediately suspended. A pending autorestart is in effect which in turn causes the autorestart routine to be executed during the initialization of the next Adabas session.

In nucleus cluster environments, the GLOBAL option can be used to cancel the Adabas session in all active cluster nuclei.

CLOGMRG Command

```
CLOGMRG = { YES | NO }
```

Switches automatic command log merging (ADARUN CLOGMRG parameter value) on or off in nucleus cluster environments.

The CLOGMRG command is global by definition and affects all nuclei in the cluster. If a NUCID is specified, it is ignored.

CLUFREEUSER Command



Note: The `CLUFREEUSER` command is only valid in cluster environments. It can be issued against the local nucleus only or, with the GLOBAL option, against all active and inactive nuclei in the cluster.

Use the `CLUFREEUSER` command to delete leftover cluster user table elements (UTEs or PLX-USERs) in common storage that are no longer associated with user queue elements (UQEs) in a nucleus.

Parameter	Is...
TNA	TNA specifies a decimal number specifying the timeout value in seconds (*max-time*). PLXUSERs that are not used during the time specified may be deleted if other conditions are fulfilled. If TNA= is not specified, PLXUSERs may be deleted without regard to their recent use.
UID	UID specifies a character string or hexadecimal byte string format user ID (*userid*) as follows: ■ *cccccccc* where the argument is 1-8 letters, digits, or embedded '-' signs without surrounding apostrophes. ■ *'cccccccc'* ↩ where the argument is 1-8 characters with surrounding apostrophes. A character string must be enclosed in apostrophes if it contains characters other than letter, digits, or embedded '-' signs. If a specified character string is less than eight characters long, it is implicitly padded with blanks. If the last eight bytes of a user's 28-byte communication ID match a specific user ID, that user's PLXUSER may be deleted if other conditions are fulfilled. If UID is not specified, PLXUSERs may be deleted regardless of their user IDs.
FORCE	FORCE indicates that leftover PLXUSERs are to be deleted even if the users are due a response code 9 (ADARSP009), subcode 20. If FORCE is not specified, such PLXUSERs are not deleted. Before using the FORCE parameter, ensure that the users owning the PLXUSERs to be deleted will not expect any of their transactions to remain open.
GLOBAL	GLOBAL indicates that leftover PLXUSERs throughout the Adabas cluster are to be deleted if they are no longer associated with UQEs and are eligible according to the other specified parameters. Additionally and subject to the other rules, leftover PLXUSERs are deleted if their assigned nuclei have terminated since their last use. If GLOBAL is not specified, only PLXUSERs assigned to the local nucleus and used since the nucleus start are eligible for deletion.

CLUPUBLPROT Command

```
CLUPUBLPROT={YES | NO}
```

Use the `CLUPUBLPROT` command to dynamically modify the setting of the ADARUN CLUPUBLPROT parameter. For more information, read CLUPUBLPROT Parameter: Publishing of Protection Data.

The CLUPUBLPROT command is only valid in cluster environments. It is global by definition and affects all nuclei in the cluster. If a NUCID is specified, it is ignored.

CT Command

```
CT = timeout-limit
```

Dynamically override the ADARUN CT parameter value; that is, the maximum number of seconds that can elapse from the time an Adabas command has been completed until the results are returned to the user through interregion communication (which depends on the particular operating system being used). The minimum setting is 1; the maximum is 16777215.

In nucleus cluster environments, the CT command is global by definition and affects all nuclei in the cluster. If a NUCID is specified, it is ignored.

DATASPACEWARN Command

```
DATASPACEWARN=threshold1[,inc1[,threshold2[,inc2[,threshold3[,inc3]]]]]
```

Use the `DATASPACEWARN` command to modify the setting of the `DATASPACEWARN` ADARUN parameter while the database is running. DATASPACEWARN specifies up to three pairs of threshold percentages and percentage change increments, expressed as percentages of the total Data Storage (DATA) portion of the database. When Data Storage space use reaches a specified threshold percentage, a space usage message is issued. In addition, when the Data Storage space use changes by the increment specified for that threshold, additional space usage messages are issued.

Note: For the purposes of database space notifications, the space used in the Adabas database comprises all database blocks that have been assigned either to some common purpose (for example, DSST, PPT, DLOG area) or to a file in the database. Only space

listed in the free space table (FST) is not calculated as used. In particular, free space allocated for some file but not yet used is calculated as used.

The following specifications can be made in the DATASPACEWARN command:

Parameter	Description
threshold#	A threshold, expressed as a percentage of the total Data Storage (DATA) portion of the database. When DATA space use exceeds or falls below any of the thresholds set, a space usage message is issued. The default is zero (0). If zero is specified for the first threshold setting, no space messages are issued. When more than one threshold is specified, the second threshold specification must be greater than the first. Likewise, the third threshold specification (if made) must be greater than the second.
inc#	An increment percentage, expressed as a percentage of the total DATA portion of the database. When DATA space use changes (increases or decreases) from the associated threshold by the amount given in this increment percentage, another space usage message is issued. The default is five percent (5).

Up to three threshold-increment pairs can be specified.

DAUQ Command

DAUQ

Display the user queue element (UQE) of each user who has executed at least one Adabas command within the last 15 minutes. Up to 100 user queue elements will be displayed.

In nucleus cluster environments, the NUCID=0 option can be used to display information for all active cluster nuclei. Information is displayed for each nucleus, one after the other.

DCMDSTAT Command

DCMDSTAT

Display statistics about the usage distribution of commands by command type.

Here is sample output from the DCMDSTAT operator command when run using the ADADBS OPERCOM utility function:

```
ADADBS OPERCOM DCMDSTAT

Type              Commands
 --------------------------
 A1/4               527,017
 CL                       3
 ET                      28
 HI                 528,086
 L2/5            20,784,748
 L3/6             9,977,842
 N1/2                   282
 OP                       3
 UC                       3
 REST                     1
 --------------------------
 Total           31,818,013
```

DCQ Command



Display all posted command queue elements (CQEs). Each CQE's user ID, job name, and buffer length is displayed.

In nucleus cluster environments, the NUCID=0 option can be used to display information for all active cluster nuclei. Information is displayed for each nucleus, one after the other. Here is sample output from the DCQ operator command run using the ADADBS OPERCOM utility function:

```
User/Terminal-ID  Jobname   SECUID    Time      Status  Cmd  File   Length
 ------------------------------------------------------------------------
 'Z007Z007'        ABCTPCTA  ABC       13:36:14  Done    L3     27     1664
 'Z019Z019'        ABCTPCTA  ABC       13:36:14  Ready   A1     21     1388
 C90060C839F60000  ABCDCQ    ABC       13:36:14  Active  LA      0      720
```

DDIB Command



Display data integrity block (DIB). This block contains entries indicating which Adabas utilities are active and the resources being used by each utility. The DDIB function can be performed with either an active or an inactive nucleus.

In nucleus cluster environments, the information displayed by the DDIB command is global; the command can be run on any nucleus.

DDSF Command

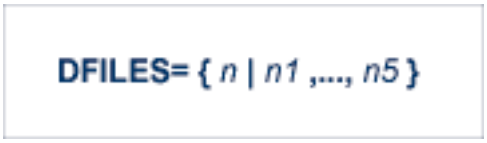


Display Adabas Delta Save Facility Facility (DSF) status. The Adabas nucleus displays the DSF status on the operator console as well as in the ADADBS job protocol.

This function is only available if the nucleus is run with the parameter ADARUN DSF=YES.

In nucleus cluster environments, the information displayed by the DDSF command is global; the command can be run on any nucleus.

DFILES Command



Displays the number of access, update, EXU, and UTI users for the specified files. User types are totaled for each file, and are listed by file. Up to five files can be specified in this command. Up to 798 users are displayed.

In nucleus cluster environments, the NUCID=0 option can be used to display information for all active cluster nuclei. Information is displayed for each nucleus, one after the other. Here is sample output from the DFILES operator command run using the ADADBS OPERCOM utility function:

```
      U E    U      A
      T X     P      C
File  I U      D      C    User   User/Terminal-ID Jobname  SECUID   ETID
-------------------------------------------------------------------------------
  11  1 0      1      1  00008ACB C9108B278FF52000 ABCLOD   ABC

  12  0 1      0      1  00008BF3 C9108CF0F4E09000 ABC      ABC

  23  0 0     14     17  00008EC7 'Z001Z001'       ABCTPCTA ABC      Z01-ETID
                         00008EC8 'Z002Z002'       ABCTPCTA ABC      Z02-ETID
                         00008EC9 'Z003Z003'       ABCTPCTA ABC      Z03-ETID
                         00008ECA 'Z004Z004'       ABCTPCTA ABC      Z04-ETID
                         00008ECB 'Z005Z005'       ABCTPCTA ABC      Z05-ETID
                         00008ECC 'Z006Z006'       ABCTPCTA ABC      Z06-ETID
                         00008ECE 'Z007Z007'       ABCTPCTA ABC      Z07-ETID
```

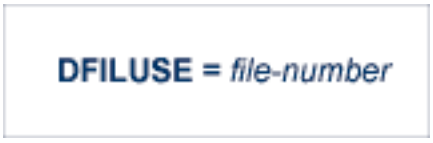
```
                        00008ECF 'Z008Z008'       ABCTPCTA ABC      Z08-ETID
                        00008ED0 'Z009Z009'       ABCTPCTA ABC      Z09-ETID
                        00008ED1 'Z010Z010'       ABCTPCTA ABC      Z10-ETID
                        00008ED2 'Z011Z011'       ABCTPCTA ABC      Z11-ETID
                        00008ED3 'Z012Z012'       ABCTPCTA ABC      Z12-ETID
                        00008ED5 'Z013Z013'       ABCTPCTA ABC      Z13-ETID
                        00008ED8 'Z014Z014'       ABCTPCTA ABC      Z14-ETID
                        00008ED9 'Z015Z015'       ABCTPCTA ABC      Z15-ETID
                        00008EDA 'Z016Z016'       ABCTPCTA ABC      Z16-ETID
                        00008EDB 'Z017Z017'       ABCTPCTA ABC      Z17-ETID
```

DFILESTAT Command



Display statistics for the number of commands by file.

DFILUSE Command



Displays the count of commands processed for the specified file so far during the current session.

In nucleus cluster environments, the NUCID=0 option can be used to display information for all active cluster nuclei. Information is displayed for each nucleus, one after the other.

DHQA Command



Display all hold queue elements (HQEs) and all users of each element.

Here is sample output from the DHQA operator command run using the ADADBS OPERCOM utility function:

```
 File        ISN  Mode    User   User/Terminal-ID  Jobname   SECUID   ETID
 ------------------------------------------------------------------------------
   10          1  SHR   000072A1 'U001U001'        ABCTXG    ABC
                  SHR   000072A2 'U002U002'        ABCTXG    ABC
                  SHR   000072A3 'U003U003'        ABCTXG    ABC
                  SHR   000072A4 'U004U004'        ABCTXG    ABC
                  SHR   000072A5 'U005U005'        ABCTXG    ABC
                  SHR   000072A6 'U006U006'        ABCTXG    ABC
                  SHR   000072A7 'U007U007'        ABCTXG    ABC
                  SHR   000072A8 'U008U008'        ABCTXG    ABC
   11         10  SHR   000072D6 C90F5F4552580000  SILLYJOB
   22          2  EXCL  000072E8 'Z001Z001'        ABCTPCTA  ABC       Z01-ETID
   22          3  EXCL  000072E2 'Z002Z002'        ABCTPCTA  ABC       Z02-ETID
   22          5  EXCL
   22          7  EXCL  000072E9 'Z005Z005'        ABCTPCTA  ABC       Z05-ETID
   22         10  EXCL  000072F0 'Z008Z008'        ABCTPCTA  ABC       Z08-ETID
   22         12  EXCL  000072E3 'Z003Z003'        ABCTPCTA  ABC       Z03-ETID
   22         15  EXCL  000072ED 'Z009Z009'        ABCTPCTA  ABC       Z09-ETID
   23       4783  EXCL  000072F3 'Z011Z011'        ABCTPCTA  ABC       Z11-ETID
...
   29     108191  EXCL  000072E4 'Z004Z004'        ABCTPCTA  ABC       Z04-ETID
   29     113431  EXCL  000072E4 'Z004Z004'        ABCTPCTA  ABC       Z04-ETID
   29     113823  EXCL  000072E4 'Z004Z004'        ABCTPCTA  ABC       Z04-ETID
   29     129953  EXCL  000072E4 'Z004Z004'        ABCTPCTA  ABC       Z04-ETID
   29     164217  EXCL  000072E4 'Z004Z004'        ABCTPCTA  ABC       Z04-ETID
   29     171617  EXCL  000072E4 'Z004Z004'        ABCTPCTA  ABC       Z04-ETID
   29     181868  EXCL  000072E4 'Z004Z004'        ABCTPCTA  ABC       Z04-ETID
   29     197793  EXCL  000072E4 'Z004Z004'        ABCTPCTA  ABC       Z04-ETID
```

DLOCKF Command



Display locked files. The type of lock appears in parentheses following the file number. Possible file types are:

Type	Description
F	The file was locked by LOCKF.
U	The file was locked by LOCKU.
X	The file was locked by LOCKX.
A	The file was advance locked by ALOCKF.

In nucleus cluster environments, the NUCID=0 option can be used to display information for all active cluster nuclei. Information is displayed for each nucleus, one after the other.

DNC Command



Display the number of posted command queue elements (CQEs) waiting to be selected.

In nucleus cluster environments, the NUCID=0 option can be used to display information for all active cluster nuclei. Information is displayed for each nucleus, one after the other.

DNH Command



Display the number of ISNs currently in the hold queue.

In nucleus cluster environments, the NUCID=0 option can be used to display information for all active cluster nuclei. Information is displayed for each nucleus, one after the other.

DNU Command



Display the number of current users.

In nucleus cluster environments, the NUCID=0 option can be used to display information for all active cluster nuclei. Information is displayed for each nucleus, one after the other.

DONLSTAT Command



Note: Not currently available for use with Adabas Parallel Services cluster nuclei.

Display status of each active reorder or invert online process together with the process ID.

In nucleus cluster environments, the NUCID=0 option can be used to display information for all active cluster nuclei. Information is displayed for each nucleus, one after the other.

DPARM Command



Display the Adabas session parameters currently in effect. The parameters are displayed in alphabetic order. In a cluster environment, they are also grouped by nucleus ID (NUCID).

Note: Additional Adabas add-on products and nucleus components will display additional parameters from those shown for a classic nucleus. A sample of the output produced by an ADADBS OPERCOM DPARM run is shown below:

```
ADADBS OPERCOM DPARM

          AOSLOG=YES
          ARNWORKBUFFERS=270
          ASSOSPACEWARN=50,10
          ASYTVS=YES
          AUDITLOG=CONSOLE
          AUTOCQENV=ALL
          AUTOCQTIME=60
          AUTOINCASSOSIZE=500
          AUTOINCASSOTHRESHOLD=80
          AUTOINCASSOTOTAL=10,000
          AUTOINCDATASIZE=1000
          AUTOINCDATATHRESHOLD=80
          AUTOINCDATATOTAL=20,000
          CT=900
          DATASPACEWARN=60,5
          FMXIO=1
          INDEXCROSSCHECK=YES
          INDEXUPDATE=ADVANCED
          INTNAS=3,600
          LARGEPAGE=NO
          LBP= 104,860,416
          LCP=10,000
          LDEUQP=5,000
          LFIOP=1
          LFP=10,240,000
          LI=819,200
          LOGCB=YES
          LOGFB=NO
          LOGGING=NO
          LOGIB=YES
          LOGIO=NO
```

```
          LOGMB=NO
          LOGRB=YES
          LOGSB=NO
          LOGUX=YES
          LOGVB=NO
          LOGVOLIO=NO
          LOGWARN=30
          LP=4,000
          LQ=220,000
          LRPL=104,857,600
          LS=29,952
          LU=64,000
          LWP=1,200,000
          MXTNA=7,200
          MXTSX=7,200
          MXTT=900
          NC=2,000
          NH=5,000
          NISNHQ=1,000
          NONDES=YES
          NPLOGBUFFERS=9
          NQCID=10
          NSISN=51
          NT=30
          NU=10,000
          NWORK1BUFFERS=9
          READONLY=NO
          REFSTPRT=YES
          REVIEW=NO
          RIAFTERUPDATE=NO
          SECUID=WARN
          SMF=YES
          SMFDETAIL=NONE
          SMFINTERVAL=5
          SMFRECNO=255
          SMFSUBSYS=CURR
          SMF89=NO
          TLSCMD=900
          TNAA=3,600
          TNAE=3,600
          TNAX=300
          TT=300
          UPDATECONTROL=NODELAY
          UTIONLY=NO
          V64BIT=YES


 A D A D B S  Terminated normally                    2016-02-24  23:17:00
```

DRES Command



Display the allocated pool space and the highest use level ('high water mark') reached so far during the current interval by record count and by percent for the following resources:

Resource Name	Description	Associated Parameter
AB - Pool	Attached buffer table	NAB
CQ - Pool	Command queue	NC
DUQ - Pool	Unique descriptor pool	LDEUQP
FI - Pool	Internal format buffer pool	LFP
HQ - Pool	Hold queue	NH
Plog-IO B	PLOG I/O buffers	NPLOGBUFFERS
REDO - Pool	Cluster redo pool (only with Adabas cluster environments)	LRDP
RPL - Pool	Replication pool (only with Event Replicator for Adabas environments)	LRPL
SC - Pool	Security pool	LCP
TBI - Pool	Pool for the table of ISNs	LI
TBS - Pool	Pool for the table of sequential commands	LQ
UQ - Pool	User queue	NU
UQF - Pool	User queue file list pool	NU
Wrk1 - IO B	Work I/O buffers	NWORK1BUFFERS
WORK - Pool	Work pool (WP)	LWP
Work Prt1	Work part 1 area (WKP1) **Note:** The maximum pool value of Work part 1 is derived from the LP parameter. It corresponds to the maximum number of blocks a transaction can spend on Work Part 1 before Adabas decides to back it out.	LP
Work Prt1b	Work Part 1 extension (only with Adabas cluster environments)	CLUWORK1CACHE
Work Prt2	Work part 2 area (WKP2)	LWKP2
Work Prt3	Work part 3 area (WKP3)	
XID -Pool	Pool for global transaction IDs (only with Adabas Transaction Manager environments)	NU

The actual values are displayed in nucleus message ADAN28, described in the *Adabas Messages and Codes Manual*.

SESSION: Display High Watermark for the session

By default, ADADBS OPERCOM DRES only returns High Watermark statistics for the current interval, which are High Watermark statistics collected after the last refresh. If ADADBS REFRESHSTATS ALL or ADADBS REFRESHSTATS POOLUSAGE has not been invoked, High Watermark statics for the session are returned, which are collected from the start of the nucleus. See ADADBS REFRESHSTATS for more information about invoking statistic refreshes.

When the `SESSION` parameter is specified, high watermark statistics collected from the start of the nucleus will be displayed, even if an ADADBS REFRESHSTATS has been invoked.

```
ADADBS OPERCOM DRES,SESSION
```

TS: Display timestamp information

When the `TS` parameter is specified, the time and date of the recorded high watermark will be included in the output. The date time will correspond to the interval, or the session, depending on which is specified.

Note: When `TS` parameter is specified, the output is extended to use 121 columns. Normally, the output defaults to 80 columns.

```
ADADBS OPERCOM DRES,TS
OR:
ADADBS OPERCOM DRES,SESSION,TS
```

DSPACE Command



Display space defined and used in ASSO and DATA.

Here is sample output from the DSPACE operator command run using the ADADBS OPERCOM utility function:

```
ADADBS OPERCOM DSPACE

         Database ASSO usage is 80%
         16,101KB of 20,079KB in use
         6,481 of 8,082 blocks in use
         24 of 30 cylinders in use

         Database DATA usage is 71%
         15,983KB of 22,204KB in use
         3,232 of 4,490 blocks in use
         22 of 30 cylinders in use
```

DSTAT Command



Display the current Adabas nucleus operating status.

Note: After issuing a REFRESHSTATS, DSTAT displays the refreshed statistics.

DTH Command



Display thread status.

DUQ Command



Display all active and inactive user queue elements.

Here is sample output from the DUQ operator command run using the ADADBS OPERCOM utility function:

```
  User    User/Terminal-ID  Jobname   SECUID   ETID      Typ  State  Active
-------------------------------------------------------------------------------
00002BD8  'U001U001'        ABCTXG    ABC                 ACC  OK     09:46:03
00002C20  'Z001Z001'        ABCTPCTA  ABC      Z01-ETID   ET   ET     09:46:19
00002C27  'Z002Z002'        ABCTPCTA  ABC      Z02-ETID   ET   TRANS  09:46:18
00002C26  'Z003Z003'        ABCTPCTA  ABC      Z03-ETID   ET   TRANS  09:46:19
00002C2E  'Z004Z004'        ABCTPCTA  ABC      Z04-ETID   ET   ET     09:46:19
00002C13  'Z005Z005'        ABCTPCTA  ABC      Z05-ETID   ET   T-OUT  09:40:31
00002C30  C8F39ABD5A18DE84  ABCDUQ    ABC                 UTI  OK     09:46:19
00000002  'ABC    1'        DAEFCO    ABC                 UTI  OK     08:33:56
```

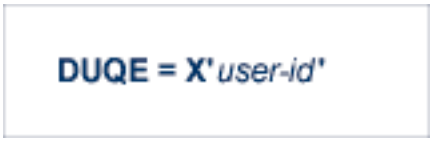
DUQA Command



Display all user queue elements (UQEs).

In nucleus cluster environments, the NUCID=0 option can be used to display information for all active cluster nuclei. Information is displayed for each nucleus, one after the other.

DUQE Command



Display a user queue element for the specified Adabas-assigned user ID as follows:

```
DUQE=X'A3CF2'
```

The user ID must be entered in hexadecimal format. Do not use a job name for the user ID. In nucleus cluster environments, NUCID must always be specified because the user ID is not unique to the cluster. When a user is associated with a time zone this information is displayed.

Here is sample output (when the user is not associated with a time zone) from the DUQE operator command run using the ADADBS OPERCOM utility function:

```
                                                                        Last
  User    User/Terminal-ID  Jobname   SECUID   ETID      Typ  State  Active
 ------------------------------------------------------------------------------
 000001A6  'Z001Z001'        AZUTPCTA  AZU               ET   TRANS  13:41:05

Communication ID  = 000FA10E20980000 4040404040404040 00FB0800 ...
Session start     = 13:38:42
Transaction start = 13:41:04

Number of files   =    8

             File =   21 (ACC)
             File =   22 (ACC)
             File =   23 (ACC)
             File =   25 (UPD)
             File =   26 (UPD)
             File =   27 (UPD)
             File =   28 (ACC)
             File =   29 (UPD)
```

Here is sample output (when the user is associated with a time zone) from the DUQE operator command run using the ADADBS OPERCOM utility function:

```
                                                                          Last
  User    User/Terminal-ID  Jobname   SECUID   ETID      Typ  State  Active
 ------------------------------------------------------------------------------
 000001A6  'Z001Z001'        AZUTPCTA  AZU               ET   TRANS  13:41:05

Time Zone         = AMERICA/DENVER

Communication ID  = 000FA10E20980000 4040404040404040 00FB0800 ...
Session start     = 13:38:42
Transaction start = 13:41:04

Number of files   =    8

             File =   21 (ACC)
             File =   22 (ACC)
             File =   23 (ACC)
             File =   25 (UPD)
             File =   26 (UPD)
             File =   27 (UPD)
             File =   28 (ACC)
             File =   29 (UPD)
```

DUUQE Command



Display utility user queue elements (UQEs). Up to 100 user queue elements will be displayed.

In nucleus cluster environments, the NUCID=0 option can be used to display information for all active cluster nuclei. Information is displayed for each nucleus, one after the other.

DVOLIO Command



Display ASSO and DATA I/Os by volume serial (VOLSER) number. The number of reads and writes to each volume is displayed.

In nucleus cluster environments, the NUCID=0 option can be used to display information for all active cluster nuclei. Information is displayed for each nucleus, one after the other.

DXCACHE Command

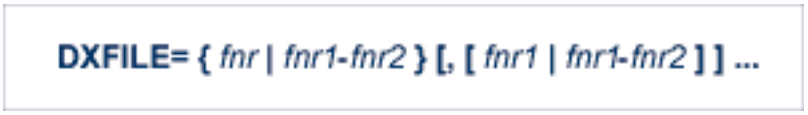


Issue the `DXCACHE` command against cluster nuclei to display the primary cache-related statistics. Statistics are displayed in messages on the console. The kinds of statistics displayed are described in *External Cache Statistics (Cluster Nucleus Only)*, in the *Adabas Cluster Services Operations Guide* or in *Global Cache Statistics (Cluster Nucleus Only)* in the *Adabas Parallel Services Operations Guide*.

The full set of statistics shown in this output is displayed only for users who have the selectable unit Adabas Online System (AOS) installed.

If you have installed only the demo version of AOS delivered with Adabas, only the statistics in the sections *Totals*, *Data Storage*, and *Normal Index* are displayed using this command.

DXFILE Command



Issue the `DXFILE` command against cluster nuclei to display cache-related statistics for files. File numbers can be specified in sequences, separated by commas. Ranges of file numbers can also be specified in the sequence. A maximum of five files can be specified in the command, so if you want to get results for more than five files, specify a range of files instead. The following are valid examples of ADADBS OPERCOM DXFILE specifications:

```
DXFILE=10-99
DXFILE=1,2,10-99
DXFILE=10-99,200
DXFILE=10-99,200-300
```

Statistics are displayed in messages on the console. The kinds of statistics displayed are described in *External Cache Statistics (Cluster Nucleus Only)*, in the *Adabas Cluster Services Operations Guide* or in *Global Cache Statistics (Cluster Nucleus Only)*, in the *Adabas Parallel Services Operations Guide*.

DXLOCK Command



Issue the DXLOCK command against cluster nuclei to display lock-related statistics. Statistics are displayed in messages on the console. The kinds of statistics displayed are described in *External Lock Statistics (Cluster Nucleus Only)*, in the *Adabas Cluster Services Operations Guide* or in *Global Lock Statistics (Cluster Nucleus Only)*, in the *Adabas Parallel Services Operations Guide*.

DXMSG Command



The DXMSG command can be used to display the messaging performance statistics available for Adabas Cluster Services or Adabas Parallel Services. Statistics are displayed in messages on the console. The kinds of statistics displayed are described in *Messaging Statistics*, in the *Adabas Cluster Services Operations Guide* or *Adabas Parallel Services Operations Guide*.

The statistics produced by the DXMSG command can be used to determine the impact of messaging on system performance and to determine how to set the ADARUN MXMSG parameter.

DXSTAT Command



Use `DXSTAT` command against cluster nuclei to display all cache- and lock-related statistics. Statistics are displayed in messages on the console. The kinds of statistics displayed are described in *External Cache Statistics (Cluster Nucleus Only)*, *External Lock Statistics (Cluster Nucleus Only)* and *WORK Data Set I/O Statistics (Cluster Nucleus Only)*, in the *Adabas Cluster Services Operations Guide* and in *Global Cache Statistics (Cluster Nucleus Only)*, *Global Lock Statistics (Cluster Nucleus Only)* and *WORK Data Set I/O Statistics (Cluster Nucleus Only)*, in the *Adabas Parallel Services Operations Guide*.

The full set of external cache statistics shown in this output is displayed only for users who have the selectable unit Adabas Online System (AOS) installed.

If you have installed only the demo version of AOS delivered with Adabas, only the external cache statistics in the sections *Totals*, *Data Storage*, and *Normal Index* are displayed using this command. All file cache statistics for files and all external lock statistics are displayed.

DXWORK Command



Use the `DXWORK` command to display the read and write I/O statistics for the accesses to the WORK datasets of peer nuclei by a particular nucleus in the cluster. The DXWORK command may or may not display I/O statistics for a nucleus' own WORK dataset.

The DXWORK command is only valid in cluster environments. The kinds of statistics displayed are described in *WORK Data Set I/O Statistics (Cluster Nucleus Only)*, in the *Adabas Cluster Services Operations Guide* and in the *Adabas Parallel Services Operations Guide*

DZSTAT Command



Display ZIIP related statistics.

Optional Parameters

Parameter	Description
ALL	Extended Statistics and Scheduling by type of work statistics. The ALL keyword may be specified and will include Extended Statistics and Scheduling by type of work statistics.

This function is only available if the nucleus was started with the parameter ADARUN ZIIP=YES.

In nucleus cluster environments, the NUCID parameter can be used to display information only for the specified nucleus. Otherwise, information is displayed for each active nucleus, one after the other.

The statistics displayed by DZSTAT correspond and are equivalent to the zIIP-related statistics at the end of the nucleus session statistics. See *Understanding the zIIP-Related Statistics* in the *Adabas for zIIP* documentation for detailed information about the various statistical figures.

FEOFAL Command

```
FEOFAL [,GLOBAL]
```

Close the current dual or multiple Audit log and switch to another dual or multiple Audit log. This command is valid only if dual or multiple audit logging is in effect and only for Adabas Audit Servers.

In nucleus cluster environments, use the GLOBAL option to switch the dual or multiple audit log in all cluster nuclei at the same time.

FEOFCL Command

```
FEOFCL [, GLOBAL ]
```

Close the current dual or multiple Command log and switch to the other dual or another multiple Command log. This command is valid only if dual or multiple command logging is in effect.

In nucleus cluster environments, the GLOBAL option can be used to switch the dual or multiple command log in all cluster nuclei at the same time.

FEOFPL Command

```
FEOFPL [, GLOBAL ]
```

Close the current dual or multiple data Protection log and switch to the other dual or another multiple Protection log. This command is valid only if dual or multiple data protection logging is in effect.

In nucleus cluster environments, the GLOBAL option can be used to switch the dual or multiple protection log in all cluster nuclei at the same time.

HALT Command



Stop Adabas session. A BT (backout transaction) command is issued for each active ET logic user. The Adabas session is then terminated; no dumps are produced.

In nucleus cluster environments, the GLOBAL option can be used to halt the Adabas session in all active cluster nuclei.

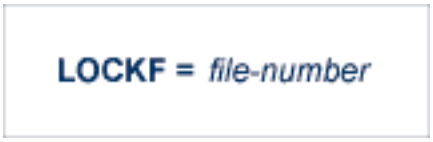
INDEXCROSSCHECK Command



Alter the setting of the ADARUN `INDEXCROSSCHECK` parameter while the nucleus is active. This operator command allows you to dynamically specify whether index crosschecking is turned on or off. Index crosschecking is a process Adabas can complete to ensure that a data storage record it is reading actually matches the descriptor value in the index by which the record was found. For more information, read *INDEXCROSSCHECK Parameter: Data Storage Record Validation*, in the *Adabas Operations Manual*.

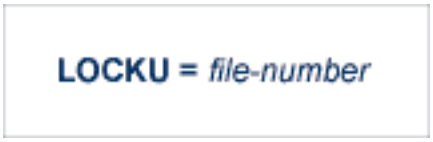
Note: Attempts to change the `INDEXCROSSCHECK` setting (via the `INDEXCROSSCHECK` command) in a cluster environment will be rejected. In a cluster nucleus (CLUSTER=SYSPLEX or CLUSTER=LOCAL), Adabas sets INDEXCROSSCHECK to "YES", overriding any user-specified INDEXCROSSCHECK value, and validates the descriptor value from the index against the corresponding value from the data storage.

LOCKF Command



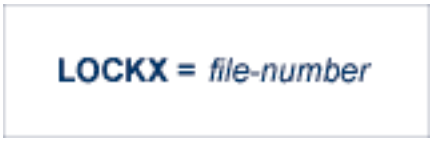
Lock the specified file. The specified file will be locked at all security levels.

LOCKU Command



Lock the specified file for all non-utility use. Adabas utilities can use the file normally.

LOCKX Command



Lock the specified file for all users except EXU or EXF users. EXU and EXF users can use the file normally. The lock is released automatically when an EXU user issues an OP command.

LOGGING Command



Start command logging.

LOGxx Command

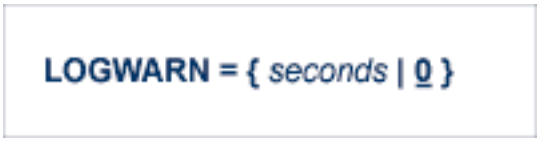


Begin logging as indicated by *xx* for each command logged where *xx* is one of the following:

- CB - the Adabas control block
- FB - the Adabas format buffer
- IB - the Adabas ISN buffer
- IO - Adabas I/O activity
- RB - the Adabas record buffer
- SB - the Adabas search buffer
- UX - user data passed in the seventh parameter of the Adabas parameter list
- VB - the Adabas value buffer

- VOLIO - the extended I/O list for CLOGLAYOUT=5 and CLOGLAYOUT=8

LOGWARN Command



Use the `LOGWARN` command to specify how often the PLOG and CLOG status is checked and resulting alert messages are produced. Valid values range from zero (0) through 2147483647 seconds. The default is 0, indicating that no PLOG or CLOG status checking occurs and no corresponding alert messages are produced. If a non-zero value is specified for LOGWARN, a valid user exit 2 or user exit 12 must also be specified.

MXCANCEL Command



The MXCANCEL command can be used against cluster nuclei to dynamically change (while the Adabas nucleus is running) the setting of the `MXCANCEL` ADARUN parameter for a nucleus. This command can be used to specify the maximum number of seconds for which one cluster nucleus waits for the termination of another nucleus in the cluster that it has canceled for not responding to an internal, intracluster command. Valid values range from 15 through 2147483647.

If the new MXCANCEL value is less than or equal to the current MXCANCELWARN setting, the MXCANCELWARN setting is automatically set to zero (0). (You can then use the MX-CANCELWARN command to dynamically set the MXCANCELWARN setting appropriately.)

For more information about the MXCANCEL setting for a nucleus, read about the ADARUN `MXCANCEL` parameter in the *Adabas Operations Manual*.

MXCANCELWARN Command



The MXCANCELWARN command can be used to dynamically change (while the Adabas nucleus is running) the setting of the `MXCANCELWARN` ADARUN parameter for a nucleus. This command can be used to specify the number of seconds after an intracluster cancellation request is made when the cluster nucleus requesting the cancellation should issue warnings about the

inability of the canceled peer nucleus to terminate. Valid values can be zero (0) or an integer ranging from 4 through the value of MXCANCEL minus 1 (MXCANCEL-1).

The value of the MXCANCELWARN setting must be less than the value of the MXCANCEL setting. If MXCANCELWARN is set to a value greater than or equal to the value specified for MXCANCEL, the value is rejected. If MXCANCELWARN is set to zero, no warnings are issued about canceled cluster nuclei that are slow to terminate.

For more information about the MXCANCELWARN setting for a nucleus, read about the ADARUN `MXCANCELWARN` parameter in the *Adabas Operations Manual*.

MXMSG Command



The MXMSG command can be used to dynamically change (while the Adabas nucleus is running) the setting of the `MXMSG` ADARUN parameter for a cluster nucleus. This command can be used to specify the maximum number of seconds in which one cluster nucleus waits for the response of another nucleus in the cluster to an internal, intracluster command. Valid values range from 15 through 32767.

If the new MXMSG value is less than or equal to the current MXMSGWARN setting, the MXMSGWARN setting is automatically set to zero (0). (You can then use the MXMSGWARN command to dynamically set the MXMSGWARN setting appropriately.)

For more information about the MXMSG setting for a nucleus, read about the ADARUN `MXMSG` parameter in the *Adabas Operations Manual*.

MXMSGWARN Command



The MXMSGWARN command can be used to dynamically change (while the Adabas nucleus is running) the setting of the `MXMSGWARN` ADARUN parameter for a cluster nucleus. This command can be used to specify the number of seconds after an intracluster request is made when the cluster nucleus issuing the intracluster request should issue warnings about the failure of the target cluster nucleus to respond. Valid values can be zero (0) or an integer ranging from 4 through the value of MXMSG minus 1 (MXMSG-1).

The value of MXMSGWARN must be less than the value of MXMSG. If MXMSGWARN is set to a value greater than or equal to the value specified for MXMSG, the value is rejected. If

MXMSGWARN is set to zero, no warnings are issued about cluster nuclei that are slow to respond.

For more information about the MXMSGWARN setting for a nucleus, read about the ADARUN MXMSGWARN parameter in the *Adabas Operations Manual*.

MXSTATUS Command



The MXSTATUS command can be used to dynamically change (while the Adabas nucleus is running) the setting of the MXSTATUS ADARUN parameter for a cluster nucleus. This command can be used to specify the number of seconds between member-level XCF status monitoring heartbeats. Valid values range from 15 through 21474836.

Note: The MXSTATUS command is only used for Adabas Cluster Services and not for Adabas Parallel Services. Adabas Parallel Services does not use XCF and ignores the setting.

The MXSTATUS command can only be used to change the value of the MXSTATUS parameter if the ADARUN MXSTATUS parameter is not zero (0). This command cannot be used to turn status monitoring on (by specifying a nonzero value if the ADARUN MXSTATUS setting is zero) or to turn it off (by specifying a value of zero when the ADARUN MXSTATUS setting is nonzero). However, setting MXSTATUS to a very high value is effectively the same as turning status monitoring off (monitoring will occur very infrequently at least).

For more information about the MXSTATUS setting for a nucleus, read about the ADARUN MXSTATUS parameter in the *Adabas Operations Manual*.

MXWTOR Command



The optional MXWTOR command can be used to dynamically change (while the Adabas nucleus is running) the setting of the MXWTOR ADARUN parameter for a cluster nucleus. This command can be used to specify the maximum number of seconds that a cluster nucleus should wait for an operator response to a self-terminating operator query. Valid values range from 15 through 64800 or zero (0).

If MXWTOR is set to zero, the operator query does not occur (no ADAX9J messages are written) and no operator intervention is expected.

For more information about the MXWTOR setting for a nucleus, read about the ADARUN `MXWTOR` parameter in the *Adabas Operations Manual*.

NOLOGGING Command



Stop or prevent command logging.

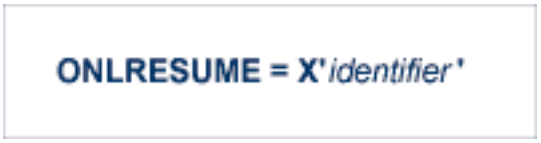
NOLOGxx Command



Stop or prevent logging of *xx* where *xx* is one of the following:

- CB - the Adabas control block
- FB - the Adabas format buffer
- IB - the Adabas ISN buffer
- IO - Adabas I/O activity
- RB - the Adabas record buffer
- SB - the Adabas search buffer
- UX - user data passed in the seventh parameter of the Adabas parameter list
- VB - the Adabas value buffer
- VOLIO - the extended I/O list for CLOGLAYOUT=5 and CLOGLAYOUT=8

ONLRESUME Command



Note: Not currently available for use with Adabas Parallel Services cluster nuclei.

Resume a previously suspended online reorder or invert process.

In a cluster environment, NUCID must always be specified because the online process ID is not unique to the cluster.

ONLSTOP Command

```
ONLSTOP = X'identifier'
```

Note: Not currently available for use with Adabas Parallel Services cluster nuclei.

Stop an online reorder or invert process cleanly. The process continues up to its next interrupt point in order to produce a consistent state, and then terminates after performing all necessary cleanup.

In a cluster environment, NUCID must always be specified because the online process ID is not unique to the cluster.

ONLSUSPEND Command

```
ONLSUSPEND = X'identifier'
```

Note: Not currently available for use with Adabas Parallel Services cluster nuclei.

Suspend an online reorder or invert process. The process continues up to its next interrupt point in order to produce a consistent state, performs a command throwback, and enters a state where it cannot be selected for processing. This command is useful if the online process is consuming too much of the nucleus resources.

In a cluster environment, NUCID must always be specified because the online process ID is not unique to the cluster.

RALOCKF Command

```
RALOCKF = n
```

Remove the advance lock on the specified file (see ALOCKF command) without running the utility.

This command is available in cluster and non-cluster environments.

RALOCKFA Command

```
RALOCKFA
```

Remove the advance lock on all files for which it has been set (see ALOCKF command) without running the utility.

This command is available in cluster and non-cluster environments.

RDUMPST Command

```
RDUMPST
```

Terminate online dump status. This command is normally used if online execution of the ADASAV utility has terminated abnormally.

READONLY Command

```
READONLY = { YES | NO }
```

Note: Not currently available for use with cluster nuclei.

Switches READONLY status on or off.

REVIEW Command

```
REVIEW = { NO | LOCAL | hub-id }
```

Note: Not currently available for use with Adabas Parallel Services cluster nuclei.

Deactivate Adabas Review; change from hub mode to local mode; specify or change the Adabas Review hub with which a nucleus communicates.

RPLCHECK Command



Note: Not valid for use with an Adabas nucleus; only valid for use against an Event Replicator Server. If this command is issued against a database that is not an Event Replicator Server, error messages result.

Use this command to perform the replication cross-check function for all active databases known (defined in one or more subscriptions) to the Event Replicator Server. When this command is run using the ADADBS OPERCOM function, the information about the cross-check function is printed to the ADADBS DDDRUCK data set. The information printed by ADADBS is the same as the information printed by the Event Replicator Server during the cross-check process initiated by the RPLCHECK operator command.

SECUID Command



Use the SECUID command to alter the ADARUN SECUID parameter while the nucleus is active. For complete information about the SECUID ADARUN parameter, read *SECUID Parameter: Security System User ID Requirement Level*, in the *Adabas Operations Manual*.

STOPF Command



Stop all users who are using the specified file. Any open transactions of the stopped users are backed out. Unless PURGE is also specified, a stopped user who returns (by sending a command) receives response code 9 (ADARSP009).

If the optional PURGE parameter is specified, the stopped users are also deleted (their user queue elements are removed from the user queue).

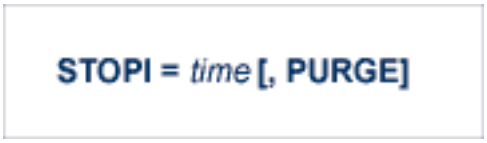
This command does not stop EXF or UTI users.

The following is an example of using the PURGE parameter:

```
ADADBS OPERCOM STOPF=5,PURGE
```

Caution: If Adabas is running with ADARUN OPENRQ=NO (specifying that users are not required to issue an OP as the first command of the session), run the STOPF command with `PURGE` only if you are certain that the users to be deleted are no longer active. If a user with an open transaction is deleted, but then returns (by sending a command), no indication is given about the transaction backout. If the user continues the transaction, logical inconsistencies in the database could occur.

STOPI Command



Stop all users who have not executed a command during the specified time interval (in seconds). Any open transactions of the stopped users are backed out. Unless PURGE is also specified, a stopped user who returns (by sending a command) receives response code 9 (ADARSP009).

This command does not stop EXF or UTI users.

If the optional PURGE parameter is specified, the stopped users are also deleted (their user queue elements are removed from the user queue).

The following is an example of using the PURGE parameter:

```
ADADBS OPERCOM STOPI=3600,PURGE
```

Caution: If Adabas is running with ADARUN OPENRQ=NO (specifying that users are not required to issue an OP as the first command of the session), run the STOPI command with `PURGE` only if you are certain that the users to be deleted are no longer active. If a user with an open transaction is deleted, but then returns (by sending a command), no indication is given about the transaction backout. If the user continues the transaction, logical inconsistencies in the database could occur.

STOPSU Command



Use the STOPSU command to stop and delete all users with the specified security user ID (*secuid*). Any open transactions of the stopped users will be backed out. No response code is issued; the next time the stopped user issues a command, a new user queue element (UQE) is created.

Caution: If Adabas is running with ADARUN OPENRQ=NO (specifying that users are not required to issue an OP as the first command of the session), run the STOPSU command only if you are certain that the users to be deleted are no longer active. If a user with an open transaction is deleted, but then returns (by sending a command), no indication is given about the transaction backout. If the user continues the transaction, logical inconsistencies in the database could occur. If this is a concern, use the STOPSUR command instead.

Note: The STOPSU=*secuid* command is not allowed for online reorder or invert processes. Use the ONLSTOP=X'*identifier*' command instead.

STOPSUR Command



Use the STOPSUR command to stop a user with a specified security user ID (*secuid*). Any open transactions of the stopped user are backed out. The stopped user is only deleted after he or she has issued a subsequent command and response code 22 (ADARSP22), subcode 54 has been issued in response to that command. This response code-subcode combination is used to notify users that their user session resources have been freed. Only after the response code-subcode combination has been issued is the user queue element (UQE) of the stopped user deleted.

Note: The STOPSUR=*secuid* command is not allowed for online reorder or invert processes. Use the ONLSTOP=X'*identifier*' command instead.

STOPU Command



Use the STOPU command to stop and delete the user with the Adabas-assigned user ID (in the form shown in the display commands), or to stop and delete all users with the specified job name (*job-name*). Any open transaction by the stopped users will be backed out. No response code is issued; the next time a stopped user issues a command, a new user queue element (UQE) is created.

Caution: If Adabas is running with ADARUN OPENRQ=NO (specifying that users are not required to issue an OP as the first command of the session), run the STOPU command only if you are certain that the users to be deleted are no longer active. If a user with an open transaction is deleted, but then returns (by sending a command), no indic-

ation is given about the transaction backout. If the user continues the transaction, logical inconsistencies in the database could occur. If this is a concern, use the STOPUR command instead.

Note: The `STOPU=X'userid'` command is not allowed for online reorder or invert processes. See the `ONLSTOP=X'identifier'` command instead.

The user ID must be specified in hexadecimal format; for example:

```
STOPU=X'1CF2' ↩
```

In a cluster environment, NUCID must always be specified because the user ID is not unique to the cluster.

STOPUR Command



Use the STOPUR command to stop a user with the Adabas-assigned user ID (in the form shown in the display command) or to stop a user with the specified job name. Any open transactions of the stopped user are backed out. The stopped user is only deleted after he or she has issued a subsequent command and response code 22 (ADARSP22), subcode 54 has been issued in response to that command. This response code-subcode combination is used to notify users that their user session resources have been freed. Only after the response code-subcode combination has been issued is the user queue element (UQE) of the stopped user deleted.

Note: The STOPUR=X'*userid*' command is not allowed for online reorder or invert processes. Use the ONLSTOP=X'*identifier*' command instead.

The user ID must be specified in hexadecimal format; for example:

```
STOPUR=X'1CF2'
```

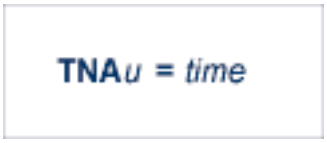
In a cluster environment, NUCID must always be specified because the user ID is not unique to the cluster.

SYNCC Command



Force resynchronization of all ET users on the nucleus. The nucleus waits for all ET users to reach ET status before continuing.

TNA*x* Command



Set non-activity time limit (in seconds) for users where *u* is one of the following:

- A - for access-only (ACC) users
- E - for ET logic users
- X - for exclusive control (EXF/EXU) users

If specified, *time* must be a value greater than zero; it overrides the ADARUN value.

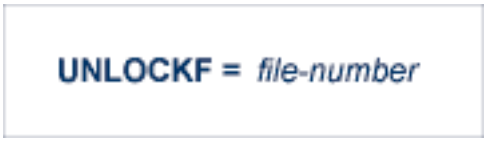
In nucleus cluster environments, the TNAu commands are global by definition and affect all nuclei in the cluster. If a NUCID is specified, it is ignored.

TT Command



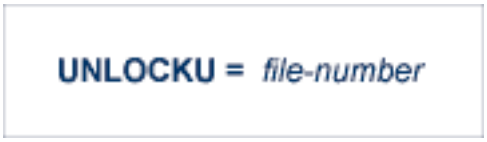
Set transaction time limit (in seconds) for ET logic users. If specified, this value must be greater than zero; it overrides the ADARUN value. In nucleus cluster environments, the TT command is global by definition and affects all nuclei in the cluster. If a NUCID is specified, it is ignored.

UNLOCKF Command



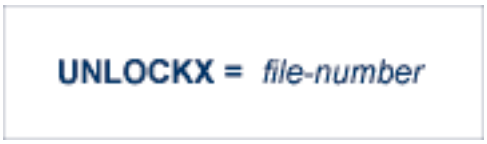
Unlock the specified file and restore its usage to the prelocked status.

UNLOCKU Command



Unlock the specified file for utility use and restore it to its prelocked status for non-utility users.

UNLOCKX Command



Unlock the specified file and restore its usage to the prelocked status.

UTIONLY Command

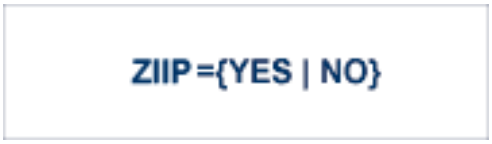


Note: Not currently available for use with Adabas Parallel Services cluster nuclei.

Switch UTIONLY status on or off.

In nucleus cluster environments, the UTIONLY command is global by definition and affects all nuclei in the cluster. If a NUCID is specified, it is ignored.

ZIIP Command



Use the `ZIIP` operator command to turn on or off the use of System z Integrated Information Processors (zIIP) in the Adabas nucleus. Issuing `ZIIP=YES` will tell the Adabas nucleus to run in SRB mode when possible and enable the use of zIIPs.

Issuing `ZIIP=NO` will tell the Adabas nucleus to stay in TCB mode and disable the use of zIIPs.

The ZIIP operator command may only be used in an Adabas nucleus started with `ADARUN ZIIP=YES`.

78 PRIORITY: Change User Priority

- Essential Parameter 500
- Optional Parameters 500
- Example 501

The PRIORITY function may be used to set or change the Adabas priority of a user. A user's priority can range from 0 (the lowest priority) to 255 (the highest priority).

The user is identified by the same user ID provided in the Adabas control block (OP command, Additions 1 field).

```
ADADBS  PRIORITY  USERID = 'user-id'
                  [NOUSERABEND]
                  [PRTY = { n | 255 }]
                  [TEST]
```

Essential Parameter

USERID: User ID

The user ID in the checkpoint file of the user for which priority is to be changed. If a record for this user does not exist, a new one is added to the checkpoint file.

Optional Parameters

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PRTY: User Priority

The priority to be in effect for the user. A value in the range 0 for lowest priority to 255 for the highest priority may be specified. The default is 255. This value will be added to the operating system priority by the interregion communications mechanism.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and vari-

ables. See **Syntax Checking with the TEST Parameter** for more information on using the TEST parameter in ADADBS functions.

Example

```
ADADBS PRIORITY USERID='USER24',PRTY=7
```

Set the priority assignment for the user with the user ID "USER24" to "7".

79 REACTLOG: Reactivating Command Logging

- Optional Parameters 504
- Example 504

The REACTLOG function allows you to reactivate command logging in an active nucleus where it had been disabled previously as a result of an I/O error. The cause of the I/O error needs to be corrected before running this utility function or command logging will simply fail again and will not be reactivated.

```
ADADBS REACTLOG  [NOUSERABEND]
                 [TEST]
```

Optional Parameters

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See **Syntax Checking with the TEST Parameter** for more information on using the TEST parameter in ADADBS functions.

Example

```
ADADBS REACTLOG
```

This example reactivates command logging after the problem that caused it to be disabled has been resolved.

80 RECORDSPANNING: Enable or Disable Record Spanning

- Syntax 506
- Essential Parameters 506
- Optional Parameters 506
- Example 507

The RECORDSPANNING function is used to enable or disable record spanning for a file. When record spanning is enabled, the size of compressed records in a file may exceed the maximum data storage block size.

Syntax

```
ADADBS  RECORDSPANNING  FILE = file-number
                        MODE = { ON | OFF }
                        [PASSWORD = 'password' ]
                        [NOUSERABEND]
                        [TIMELIMIT = { 60  |  timelimit } ]
                        [TEST]
```

Essential Parameters

FILE: File Number

FILE specifies the number of the file for which you want to enable or disable record spanning. Specify the valid file number of the file in the database for which you want to control record spanning.

MODE: Turn Record Spanning On or Off

MODE indicates whether record spanning should be activated for the file or not. Valid values are "ON" and "OFF". A value "ON" turns record spanning on; a value of "OFF" turns record spanning off.

Optional Parameters

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PASSWORD: File Password

The password of the file for which record spanning is to be enabled or disabled. This parameter is required if the file is password-protected. Specify the password between apostrophes ('). In addition, the password must provide update authority for this function to work correctly.

TIMELIMIT: Number of Seconds

The maximum number of seconds the ADADBS RECORDSPANNING function may run. If record spanning is being turned off for a file, Adabas must verify that no spanned records exist in the file. Since this analysis may take some time, you can use the TIMELIMIT parameter to indicate how long you are willing to wait for the ADADBS RECORDSPANNING function to run. If the file scan takes more time than specified by the TIMELIMIT parameter, the function will be aborted and appropriate error messages are issued.

The default is 60 seconds.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See *Syntax Checking with the TEST Parameter* (elsewhere in this section) for more information on using the TEST parameter in ADADBS functions.

Example

In the following example, record spanning is turned on for file 17. The records in file 17 can therefore exceed the boundaries of a data storage block.

```
ADADBS RECORDSPANNING MODE=ON,FILE=17
```

81 RECOVER: Recover Space

- Optional Parameters 510

The RECOVER function recovers allocated space by rebuilding the free space table (FST). The RECOVER function subtracts file, DSST, and alternate RABN extents from the total available space.

```
ADADBS  RECOVER  [NOUSERABEND]
                 [TEST]
```

Optional Parameters

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See **Syntax Checking with the TEST Parameter** for more information on using the TEST parameter in ADADBS functions.

82 REFRESH: Set File to Empty Status

- Essential Parameter 512
- Optional Parameters 512
- Example 513

The REFRESH function sets the file to 0 records loaded, sets the first extent for the address converter, Data Storage, normal index, and upper index to *empty* status, and deallocates other extents.

When the REFRESH function completes successfully, if the file was previously locked automatically by Adabas, the automatic lock is reset. Any locks previously set with the operator commands LOCKU or LOCKF remain set.

```
ADADBS  REFRESH  FILE = file-number
                 [NOUSERABEND]
                 [PASSWORD = 'password ' ]
                 [TEST]
```

Essential Parameter

FILE: File Number

FILE specifies the file that is to be set to *empty* status.

Optional Parameters

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "`utility` TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PASSWORD: File Password

This parameter is required if the file is password-protected.

TEST: Test Syntax

This parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See **Syntax Checking with the TEST Parameter** for more information on using the TEST parameter in ADADBS functions.

Example

File 116 is to be set to empty status.

```
ADADBS REFRESH FILE=116
```

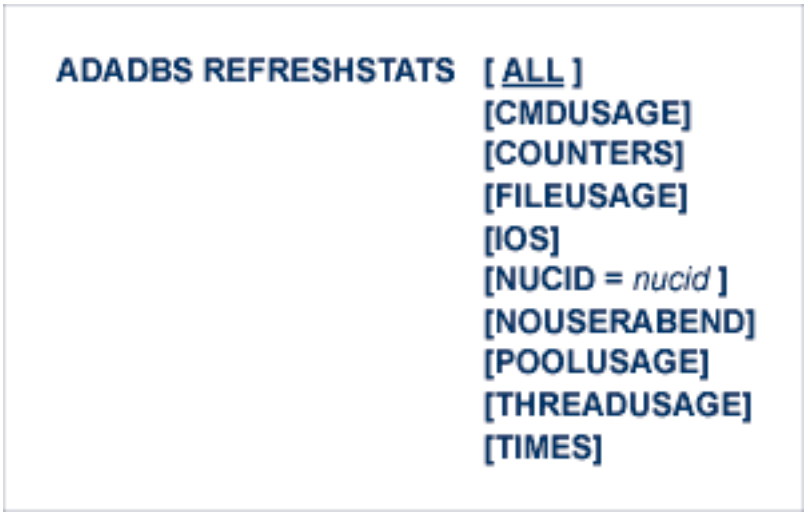
83 REFRESHSTATS: Reset Statistical Values

- Optional Parameters 516
- Example 518

The REFRESHSTATS function resets statistical values maintained by the Adabas nucleus for its current session. Parameters may be used to restrict the function to particular groups of statistical values.

When you invoke REFRESHSTATS, Adabas automatically writes the nucleus shutdown statistics to DDPRINT.

Important: Refreshing Adabas statistical values affects the corresponding Adabas Statistics Facility (ASF) field values. These values, which normally reflect the period from the start of the nucleus, will then refer to the time after the last refresh. ASF users may therefore find it useful to store the nucleus records with the appropriate ASF function before refreshing the values. DSTAT will also display the refreshed statistics.



Optional Parameters

ALL: All Statistical Values

The ALL keyword may be specified as an abbreviation for the combination of CMDUSAGE, COUNTERS, FILEUSAGE, IOS, POOLUSAGE, THREADUSAGE, and TIMES.

If none of the option keywords is specified, ALL is the default option.

CMDUSAGE: Command Usage Counters

The CMDUSAGE parameter is specified to reset the counters for Adabas direct call commands such as L*x*, S*x*, or A1.

COUNTERS: Frequency Counters

The COUNTERS parameter is specified to reset the counter fields for local or remote calls, format translations, format overwrites, Autorestarts, protection log switches, buffer flushes, and command throw-backs.

FILEUSAGE: Count of Commands Per File

The FILEUSAGE parameter is specified to reset the count of commands for each file.

IOS: I/O Counters

The IOS parameter is specified to reset the counter fields for read and write I/Os for the Associator, Data Storage, Work data sets, PLOG, and CLOG, as well as all I/O distribution statistics by volume serial number (VOLSER).

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

NUCID: Cluster Nucleus ID

Any nucleus running in an Adabas nucleus cluster is allowed to run Adabas utilities such as ADADBS. The NUCID parameter allows you to direct the ADADBS REFRESHSTATS function to a particular nucleus in the cluster for execution, just as though the command had been issued by a locally run ADADBS REFRESHSTATS operation.

If you specify NUCID=0, the statistical values are refreshed for all active nuclei in the cluster.

POOLUSAGE: High-Water Marks for Nucleus Pools

The POOLUSAGE parameter is specified to reset the high-water marks for the nucleus pools such as the work pool, the command queue, or the user queue. It is important to note that High Watermark statistics for the session (statistics collected from the start of the nucleus), can still be viewed with the DRES operator command or ADADBS OPERCOM DRES,SESSION parameter.

THREADUSAGE: Count of Commands Per Thread

The THREADUSAGE parameter is specified to reset the count of commands for each Adabas thread.

TIMES: Time Values

The TIMES parameter is specified to reset time statistics, including duration, wait time, and CPU time statistics.

Example

```
ADADBS  REFRESHSTATS
CMDUSAGE,POOLUSAGE,NUCID=3
```

After the shutdown statistics for the Adabas cluster nucleus with NUCID=3 are written to DDPRINT, the command counters and the pool high-water marks for the nucleus are reset.

84 RELEASE: Release Descriptor

- Essential Parameters 520
- Optional Parameters 521
- Example 521

The RELEASE function releases a descriptor from the descriptor space.

This function results in the release of all space currently occupied in the Associator inverted list for this descriptor. This space can then be reused for this file by reordering or ADALOD UPDATE. No changes are made to Data Storage.

When releasing descriptor space for an Adabas expanded file, perform the RELEASE function for *each individual component file* of the expanded file. Each RELEASE operation on a component file causes a message that confirms the change, and returns condition code 4.

This utility function does not need to lock the database file for its use; this function can perform its processing in parallel with active users. This means that you do not need to set the file to read-only status to run this utility function.

It is not possible to release a descriptor which has Security By Value set on it; you must first remove this security option on the descriptor before releasing it. For complete information about Adabas Security, contact your nearest representative.

```
ADADBS  RELEASE  FILE = file-number
                 DESCRIPTOR = 'name'
                 [INPARALLEL={ YES|NO }]
                 [NOUSERABEND]
                 [PASSWORD = 'password' ]
                 [TEST]
```

Essential Parameters

FILE: File Number

FILE specifies the file that contains the descriptor to be released. The file cannot be an Adabas system file.

DESCRIPTOR: Descriptor to Be Released

DESCRIPTOR specifies the descriptor to be released. Any descriptor type can be specified. A descriptor currently being used as the basis for file coupling cannot be specified. If the descriptor being released is an ADAM descriptor, the file is no longer processed as an ADAM file.

Optional Parameters

INPARALLEL: Execute function in parallel

When INPARALLEL=YES (the default) is specified, the function may be executed even if other users are active on the file. Adabas performs the function while all application commands against the file are temporarily quiesced and suspended. The integrity of the file targeted by the ADADBS function is maintained.

With INPARALLEL=NO, the function requires exclusive file control – that is, no applications can be executing against the file. Adabas refuses to execute the function (with response code 48) if other users are active on the file.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PASSWORD: File Password

This parameter is required if the file is password-protected. Specify the password between apostrophes (').

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See **Syntax Checking with the TEST Parameter** for more information on using the TEST parameter in ADADBS functions.

Example

```
ADADBS RELEASE FILE=31,DESCRIPTOR='AA'
```

Descriptor AA in file 31 is released from descriptor status.

85 RENAME: Rename File or Database

- Essential Parameter 524
- Optional Parameters 524
- Examples 525

The RENAME function may be used to change the name assigned to a file or database.

```
ADADBS  RENAME   NAME = 'name'
                 [FILE = file-number ]
                 [INPARALLEL={ YES|NO }]
                 [NOUSERABEND]
                 [PASSWORD = 'password ' ]
                 [TEST]
```

This utility function does not need to lock the database file for its use; this function can perform its processing in parallel with active users. This means that you do not need to set the file to read-only status to run this utility function.

Essential Parameter

NAME: New File Name

NAME is the new name to be assigned to the file. It is specified between apostrophes (for example, 'RESERVATIONS'). A maximum of 16 characters can be used.

Optional Parameters

FILE: File Number

FILE is the number of the file to be renamed: if specified as zero or omitted, the database is renamed.

INPARALLEL: Execute function in parallel

When INPARALLEL=YES (the default) is specified, the function may be executed even if other users are active on the file. Adabas performs the function while all application commands against the file are temporarily quiesced and suspended. The integrity of the file targeted by the ADADBS function is maintained.

INPARALLEL=YES is applicable only when renaming a file (i.e., FILE parameter is specified and nonzero), not when renaming the database.

With INPARALLEL=NO, the function requires exclusive file control – that is, no applications can be executing against the file. Adabas refuses to execute the function (with response code 48) if other users are active on the file.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend

after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PASSWORD: File Password

The password of the file. This parameter is required if the file is password-protected.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See **Syntax Checking with the TEST Parameter** for more information on using the TEST parameter in ADADBS functions.

Examples

The name of file 2 is to be changed to "INVENTORY".

```
ADADBS RENAME FILE=2,NAME='INVENTORY'
```

The database is renamed to "RESERVATIONS".

```
ADADBS RENAME  NAME='RESERVATIONS',FILE=0
```

86 RENUMBER: Change File Number

- Essential Parameter 528
- Optional Parameter 528
- Example 529

The RENUMBER function changes the number of an Adabas file.

```
ADADBS  RENUMBER  FILES = current-number, new-number
                  [NOUSERABEND]
                  [TEST]
```

Essential Parameter

FILES: Current File Number, New File Number

The number currently assigned to the file, and the new number to be assigned to the file. If the new number is assigned to another file, the RENUMBER function will not be performed.

An Adabas system file cannot be used. The file may not be security-protected, may not be coupled to another file, and may not be part of an expanded file.

Optional Parameter

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See **Syntax Checking with the TEST Parameter** for more information on using the TEST parameter in ADADBS functions.

Example

The file number for file 4 is to be changed to 40.

```
ADADBS RENUMBER FILES=4,40
```

87 REPLICATION: Activating or Deactivating Replication

- Essential Parameter 532
- Optional Parameter 532
- Examples 534

The REPLICATION function activates or deactivates replication for an Adabas database file. This function should be used with Adabas databases; it is not valid with Event Replicator Servers.

```
ADADBS  REPLICATION  FILE = file-number
                     {ON | OFF | MODIFY [ACTIVATE | DEACTIVATE] }
                     [DSBI = { ON | OFF } ]
                     [INPARALLEL={ YES | NO }]
                     [KEY = field | NOKEY]
                     [TARGET = dbid ]
```

Essential Parameter

FILE: File Number

The file number of the database file for which replication is to be activated or deactivated. This parameter is required; there is no default.



Note: Replication may not be turned on for an Adabas system file or for a ciphered file.

ON, OFF, MODIFY: Set Replication

One of the parameters `ON`, `OFF`, or `MODIFY` must be set. There is no default. Each parameter is described below:

- `ON` turns replication on for the specified file.
- `OFF` turns replication off for the specified file.
- `MODIFY` allows you to modify one or more of the replication-related values (KEY, TARGET, DSBI) for a replicated file. The `ACTIVATE` and `DEACTIVATE` parameters allow you to change the replication status for a replicated file.

Optional Parameter

ACTIVATE: Activate Replication

The `ACTIVATE` parameter is only allowed when MODIFY is specified. It will activate replication for a file that has replication defined. Replication must already be inactive for the file.

DEACTIVATE: Deactivate Replication

The `DEACTIVATE` parameter is only allowed when MODIFY is specified. It will deactivate replication for a file that has replication defined. Replication must already be active for the file.

DSBI: Define Before Image

The `DSBI` parameter defines whether or not before images of data storage are collected for replication during the update of a record on a file. Following are rules for setting the DSBI parameter:

- Possible values: "ON", "OFF"
- The parameter is optional when "ON" or "MODIFY" is specified.
- The parameter has a default value of "ON" when `ADADBS REPLICATION ON` is specified.
- The parameter is not allowed when `ADADBS REPLICATION OFF` is specified.

For more information about how this parameter is used in Adabas database processing during replication, read *Nucleus Processing* in *Event Replicator for Adabas Concepts*.

INPARALLEL: Execute function in parallel

When INPARALLEL=YES is specified, the function may be executed even if other users are active on the file. Adabas performs the function while all application commands against the file are temporarily quiesced and suspended. The integrity of the file targeted by the ADADBS function is maintained.

With INPARALLEL=NO (the default), the function requires exclusive file control – that is, no applications can be executing against the file. Adabas refuses to execute the function (with response code 48) if other users are active on the file.



Caution: If the REPLICATION function is performed with INPARALLEL=YES, the replication-related settings of the file may change in the middle of a replicated transaction. It may happen that some updates of the transaction get replicated while other updates do not get replicated. Updates performed in the course of a single command always get replicated either completely or not at all.

KEY: Define Primary Key

The `KEY` parameter defines the primary key for replication. Following are rules for setting the `KEY` parameter:

- The parameter is optional when `ON` or `MODIFY` is specified.
- The parameter is not allowed when `OFF` is specified.
- The parameter may *not* be specified when `OFF` or `NOKEY` is specified.



Note: The field name specified must be a descriptor on the file. Note that "descriptor" in this case is used generically, as the field may be a descriptor, subdescriptor, superdescriptor, etc.

NOKEY: Remove Primary Key Setting

The `NOKEY` parameter removes the primary key setting when MODIFY is specified. Following are rules for setting the `NOKEY` parameter:

- The parameter is not allowed when `OFF` is specified.
- The parameter is the default setting when `ON` is specified and neither `NOKEY` nor `KEY` is specified.
- The parameter may *not* be specified when `KEY` is specified.

TARGET: Event Replicator Target ID

`TARGET` defines the Event Replicator target database ID. This parameter is required when `ON` is specified, is optional when `MODIFY` is specified, and is not allowed when `OFF` is specified.

Examples

```
ADADBS REPLICATION FILE=33,ON,TARGET=206,KEY=AA
```

or

```
ADADBS REPLICATION FILE=70,OFF
```

or

```
ADADBS REPLICATION FILE=71,MODIFY
ADADBS ACTIVATE
```

or

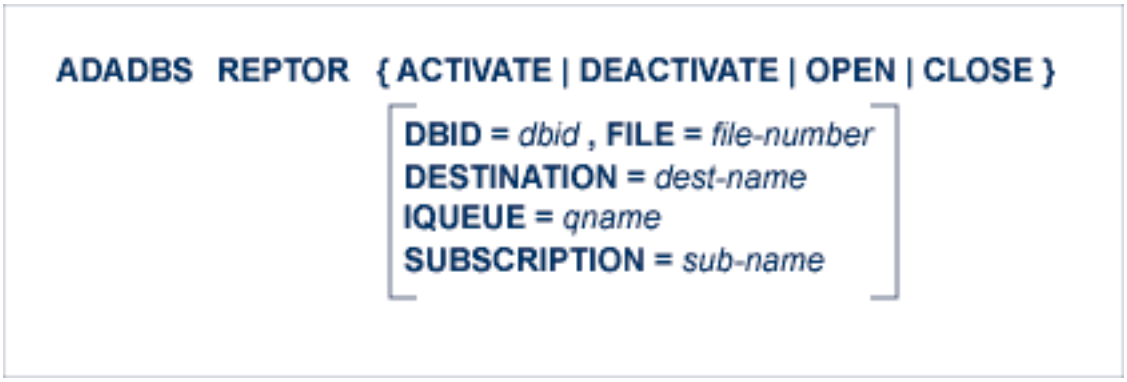
```
ADADBS REPLICATION FILE=72,MODIFY
ADADBS DEACTIVATE
```

88 ADADBS REPTOR: Activate, Deactivate, Open, or Close

Event Replicator Resources

- Essential Parameters 536
- Optional Parameters 537
- Examples 538

The ADADBS REPTOR function provides activation and deactivation control of Event Replicator resources. This function should be used with Event Replicator Servers; it is not valid with other Adabas databases.



This chapter covers the following topics:

Essential Parameters

ACTIVATE: Event Replicator Server Resource Activation Request

The `ACTIVATE` parameter requests activation for the specified file, database ID, destination, or subscription.

One of the parameters, `ACTIVATE`, `DEACTIVATE`, `OPEN`, or `CLOSE` must be set. There is no default.

DEACTIVATE: Event Replicator Server Resource Deactivation Request

The `DEACTIVATE` parameter requests deactivation for the specified file, database ID, destination, or subscription.

One of the parameters, `ACTIVATE`, `DEACTIVATE`, `OPEN`, or `CLOSE` must be set. There is no default.

OPEN: Event Replicator Server Resource Open Request

The `OPEN` parameter requests that an unavailable destination or input queue be opened. When this parameter is specified, either the `DESTINATION` or `IQUEUE` parameter must be specified.

One of the parameters, `ACTIVATE`, `DEACTIVATE`, `OPEN`, or `CLOSE` must be set. There is no default.

CLOSE: Event Replicator Server Resource Close Request

The `CLOSE` parameter requests that an available destination or input queue be closed. When this parameter is specified, either the `DESTINATION` or `IQUEUE` parameter must be specified.

One of the parameters, `ACTIVATE`, `DEACTIVATE`, `OPEN`, or `CLOSE` must be set. There is no default.

Optional Parameters

DBID: Replicated DBID

The DBID parameter, when specified without the FILE parameter, will activate replication for any inactive files or deactivate replication for any active files for a given DBID. When specified with the FILE parameter, the DBID parameter identifies the database in which the file specified by the FILE parameter resides.

The DBID parameter is mutually exclusive with the DESTINATION, IQUEUE, and SUBSCRIPTION parameters.

DESTINATION: Replication Destination

The DESTINATION parameter can only be specified when the ACTIVATE, DEACTIVATE, OPEN, or CLOSE parameters are specified. The DESTINATION parameter supplies the name of the destination that should be activated, deactivated, opened, or closed. The destination specified must be defined to the Event Replicator Server.

If "ACTIVATE" is specified, the destination must already be inactive; if "DEACTIVATE" is specified, the destination must already be activated.

If "OPEN" is specified, the destination must be in an unavailable state; if "CLOSE" is specified, the destination must be in an available state.

The DESTINATION parameter is mutually exclusive with the DBID, FILE, IQUEUE, and SUBSCRIPTION parameters.

FILE: Replicated File

The FILE parameter will activate or deactivate replication for a single file in a specific database. A corresponding DBID parameter must also be specified.

If "ACTIVATE" is specified, the file must already be deactivated; if "DEACTIVATE" is specified, the file must already be activated.

The FILE parameter is mutually exclusive with the DESTINATION, IQUEUE, and SUBSCRIPTION parameters.

IQUEUE: Replication Input Queue

The IQUEUE parameter can only be specified when the OPEN or CLOSE parameters are specified. It supplies the name of the input queue (IQUEUE) that should be opened or closed and must be defined to the Event Replicator Server.

If "OPEN" is specified, the input queue must be in an unavailable state; if "CLOSE" is specified, the input queue must be in an available state.

The IQUEUE parameter is mutually exclusive with the DBID, DESTINATION, FILE, and SUBSCRIPTION parameters.

SUBSCRIPTION: Replication Subscription

The subscription specified for the SUBSCRIPTION parameter must be defined to the Event Replicator Server. It specifies the name of the subscription definition to use.

If "ACTIVATE" is specified, the subscription must already be deactivated. If "DEACTIVATE" is specified, the subscription must already be activated.

The SUBSCRIPTION parameter is mutually exclusive with the DBID, FILE, IQUEUE, and DESTINATION parameters.

Examples

The following example requests that the Event Replicator Server activate all inactive files for database 232.

```
ADADBS REPTOR ACTIVATE,DBID=232
```

The following example requests that the Event Replicator Server activate file 2 on database 232. The file is currently inactive.

```
ADADBS REPTOR ACTIVATE,DBID=232,FILE=2
```

The following example requests that the Event Replicator Server activate the destination defined by the DEST0001 destination. The destination is currently inactive.

```
ADADBS REPTOR ACTIVATE,DESTINATION=DEST0001
```

The following example requests that the Event Replicator Server activate the subscription defined by the SUBS0001 subscription. The subscription is currently inactive.

```
ADADBS REPTOR ACTIVATE,SUBSCRIPTION=SUBS0001
```

The following example requests that the Event Replicator Server deactivate all active files for database 232.

```
ADADBS REPTOR DEACTIVATE,DBID=232
```

The following example requests that the Event Replicator Server deactivate file 2 on database 232. The file is currently active.

```
ADADBS REPTOR DEACTIVATE,DBID=232,FILE=2
```

The following example requests that the Event Replicator Server deactivate the destination defined by the DEST0001 destination. The destination is currently active.

```
ADADBS REPTOR DEACTIVATE,DESTINATION=DEST0001
```

The following example requests that the Event Replicator Server deactivate the subscription defined by the SUBS0001 subscription. The subscription is currently active.

```
ADADBS REPTOR DEACTIVATE,SUBSCRIPTION=SUBS0001
```

The following example requests that the Event Replicator Server open (restart) destination DEST0001. Each output task will be asked to process the destination if that output task does not already have the destination open for processing.

```
ADADBS REPTOR OPEN,DESTINATION=DEST0001
```

The following example requests that the Event Replicator Server close destination DEST0001. Each output task will be asked to close the destination if that output task does not already have the destination closed for processing.

```
ADADBS REPTOR CLOSE,DESTINATION=DEST0001
```

The following example requests that the Event Replicator Server open (restart) input queue IQUEUE01. Each input task will be asked to open the input queue if that input task does not already have the input queue open for processing.

```
ADADBS REPTOR OPEN,IQUEUE=IQUEUE01
```

The following example requests that the Event Replicator Server close input queue IQUEUE01. Each input task will be asked to close the input queue if that input task does not already have the input queue closed for processing.

```
ADADBS REPTOR CLOSE,IQUEUE=IQUEUE01
```

89 RESETDIB: Reset Entries in Active Utility List

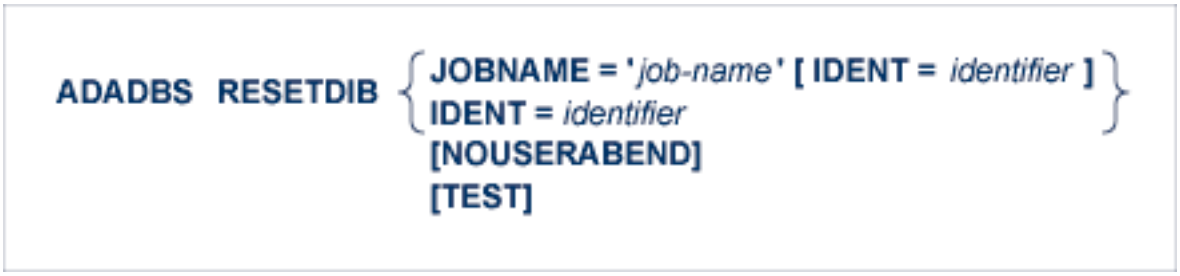
- Essential Parameters 542
- Optional Parameters 542
- Examples 543

The RESETDIB function resets entries in the active utility list (that is, the data integrity block or DIB).

Adabas maintains a list of the files used by each Adabas utility in the DIB. The DDIB operator command (or Adabas Online System) may be used to display this block to determine which jobs are using which files. A utility removes its entry from the DIB when it terminates normally. If a utility terminates abnormally (for example, the job is canceled by the operator), the files used by that utility remain *in use*. The DBA may release any such files with the RESETDIB function.

Note: The RESETDIB function can be executed either with or without an active nucleus.

To remove a DIB from an abended ADAORD REORDB, REORDATA, REORASSO, ADADBS RESETDIB has to run without an active nucleus.



Essential Parameters

JOBNAME: Job Name

This parameter specifies the name of the job whose entry is to be reset. If it is not unique, the IDENT parameter must also be specified.

IDENT: Utility Execution Identifier

A unique number that identifies a utility execution. It may be specified alone or to qualify a job name when the same name has been used for various utility executions. The identifier may be obtained using the operator command DDIB or Adabas Online System.

Optional Parameters

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "`utility` TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See **Syntax Checking with the TEST Parameter** for more information on using the TEST parameter in ADADBS functions.

Examples

The entry in the DIB block for job "JOB1" is to be deleted.

```
ADADBS RESETDIB JOBNAME='JOB1'
```

The entry in the DIB block for "JOB2" with IDENT=127 is to be deleted.

```
ADADBS RESETDIB
JOBNAME='JOB2',IDENT=127
```

90 RESETPPT: Reset PPT Blocks

- Syntax 547
- Essential Parameters 547
- Optional Parameters 547

The RESETPPT function resets the PPT blocks on the Associator data set. The PPT blocks contain information about the PLOG, CLOG, and work data sets that the nucleus is using. This information is used to perform validity checks on the work, PLOG, and CLOG data sets and is also used to determine if an autorestart is necessary.

Resetting the PPT is not without some risks:

- An attempt to reset the PPT when an autorestart is pending will prevent the nucleus from being able to recover from the current failure.
- An attempt to reset the PPT when PLOGs and CLOGs remain to be copied will disable all validity checks on the PLOG and CLOG data sets until new PPT information is written. This will, therefore, allow a data set to be overwritten, losing data.

Because of these risks, we recommend that you use the RESETPPT function with extreme caution and only with direction from your Customer support representative.



Caution: The nucleus must be down before you run this function and this function must be run in MODE=MULTI. Otherwise, errors will result.

Syntax

```
ADADBS  RESETPPT  [NOUSERABEND]
                  [TEST]
```

Essential Parameters

There are no required parameters.

Optional Parameters

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See *Syntax Checking with the TEST Parameter* (elsewhere in this section) for more information on using the TEST parameter in ADADBS functions.

91 SPANCOUNT: Count Spanned Records

- Syntax 550
- Essential Parameters 550
- Optional Parameters 550
- Example 551

The SPANCOUNT function counts and displays the number of records in a file that are spanned.

Syntax

```
ADABDS   SPANCOUNT    FILE = file-number
                      [NOUSERABEND]
                      [TIMELIMIT = {60 | timelimit } ]
                      [TEST]
```

Essential Parameters

FILE: File Number

FILE specifies the number of the file whose spanned records are to be counted. Specify the valid file number of the file in the database for which you want spanned records counted.

Optional Parameters

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See *Syntax Checking with the TEST Parameter* (elsewhere in this section) for more information on using the TEST parameter in ADADBS functions.

TIMELIMIT: Number of Seconds

The maximum number of seconds the ADADBS SPANCOUNT function may run. If the function takes more time than specified by the TIMELIMIT parameter, the function will be aborted and appropriate error messages are issued.

The default is 60 seconds.

Example

In the following example, the spanned records in file 17 are counted.

```
ADADBS SPANCOUNT FILE=17
```

The results might look like this:

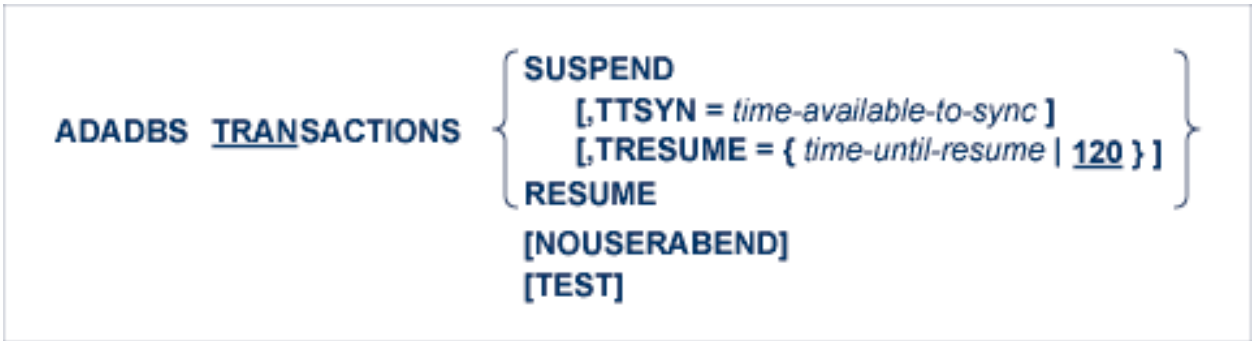
Records not spanned	15132
Primary records	19345
Secondary records	43897

92 TRANSACTIONS: Suspend and Resume Update

Transaction Processing

- Essential Parameters 555
- Optional Parameters 555
- Example 556

The TRANSACTIONS function may be used to suspend and resume update transaction processing; that is, to reach a quiesced state that could be a recoverable starting point.



Once the SUSPEND function has been submitted, new update transactions are held in the user queue. Executing transactions are allowed to finish if they can do so within the time allotted by the TTSYN parameter. Any transactions that exceed this time are backed out. In a cluster environment, all cluster nuclei are likewise quiesced.

Note: If you are trying to use the SUSPEND or RESUME functions of ADADBS TRANSACTIONS and if Natural Security is being used to log on, the FSEC Natural profile parameter for the FSEC Natural Security system file should be set with the RO (read-only) setting. If RO is not specified for the FSEC parameter, you will not be able to log onto Natural while Adabas is suspended because some of the logon programs in Natural Security require updates to the FSEC.

Once the quiesce is successful, the buffers are flushed for all nuclei so that the DASD files are current with the content of the buffers. A checkpoint SYNC-73 is written and ADADBS is notified.

At this point, you may run a third-party fast backup product such as IBM's FlashCopy or StorageTek's SnapShot to copy off the database; that is, copy pointers to the data created by the fast backup product in the electronic memory of the array storage device.

Caution: We do not recommend using such a database fast copy as a substitute for a regular Software GmbH database (or delta) save. Not only does Software GmbH have no control over the data sets that are included in the database fast copy, but it also cannot vouch for the success of the fast copy. Moreover, delta saves cannot sensibly be run on a copy of the database, as the DSF status change effected by the delta save would occur on the database copy instead of the original.

If the COPY completes before the TRESUME timeout and the RESUME function is issued, the nucleus writes a SYNS-74 checkpoint, leaves the suspended state and resumes update processing. The database was in a valid state over the whole duration of the COPY process.

If the COPY does not complete before the TRESUME timeout, Adabas automatically leaves the suspended state and resumes update processing. If the RESUME function is issued subsequently, Adabas rejects it with a response code and ADADBS terminates abnormally with an error message.

This means that whatever COPY has been produced while update processing was suspended is invalid and must not be used, because Adabas may have resumed updating the database while the COPY process was still in progress.

If the so-created copy of the database is used for recovery, removing the need to restore the database as of the time of the COPY, the subsequent regenerate should be started at the SYNC-73 checkpoint written at the end of the SUSPEND function.

Important: In a job where a SUSPEND function is followed by other job steps and then by a RESUME function, none of the job steps in between should be update-type commands or functions; otherwise, job execution will stall until the nucleus times out the suspended state.

Essential Parameters

SUSPEND: Suspend Transactions and Quiesce the Database

Use this parameter to suspend update transaction processing and quiesce the database.

RESUME: Resume Transaction Processing that was Previously Suspended

Use this parameter to resume update transaction processing that was previously suspended. If this parameter is used while Adabas is not in a suspended state or is no longer in a suspended state, this function terminates with an error.

Optional Parameters

TRESUME

Use this parameter to specify the amount of time in seconds the system is to remain quiesced after being suspended before the nucleus automatically resumes normal update transaction processing. If this parameter is not specified, the default is 120 seconds and the maximum is 86400 seconds or about 24 hours. The count begins when the nucleus has been successfully quiesced.

TTSYN

Use this parameter to specify the maximum amount of time the nucleus is to wait for all ET users to reach ET status before it forcibly ends and backs out update transactions that are still running in order to quiesce the system. If this parameter is not specified, the default is the ADARUN TT value.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See **Syntax Checking with the TEST Parameter** for more information on using the TEST parameter in ADADBS functions.

Example

Quiesce a database allowing 300 seconds for the currently running update transactions to finish and 150 seconds thereafter for the suspension to last before Adabas automatically resumes normal processing:

```
ADADBS TRANSACTIONS SUSPEND,TTSYN=300,TRESUME=150
```

93 UNCOUPLE: Uncouple Files

- Essential Parameter 558
- Optional Parameters 558
- Example 559

The UNCOUPLE function is used to eliminate the coupling relationship between two files.

```
ADADBS  UNCOUPLE   FILES = number, number
                   [NOUSERABEND]
                   [PASSWORD = 'password ' ]
                   [TEST]
```

Essential Parameter

FILES: Files to Be Uncoupled

FILES specifies the two files to be uncoupled.

Optional Parameters

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PASSWORD: File Password

PASSWORD specifies the security password for one or both files, and is required if either of the files is password-protected. If both files are password-protected, the password applies to both files. The password must be enclosed in single quotation marks.

TEST: Test Syntax

This parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See **Syntax Checking with the TEST Parameter** for more information on using the TEST parameter in ADADBS functions.

Example

Files 62 and 201 are to be uncoupled. One or both are protected with the password "PAIR05".

```
ADADBS UNCOUPLE
FILES=62,201,PASSWORD='PAIR05'
```

94 UNDELDE: Undeleting a Logically Deleted Descriptor

- Essential Parameters 562
- Optional Parameters 562
- Example 563

The UNDELDE function undeletes a logically deleted descriptor. It removes the logically deleted status for the descriptor. Once, UNDELDE has been run for a descriptor, it can be used as a search descriptor.

```
ADADBS  UNDELDE   FILE = file-number
                  DESCRIPTOR = descriptor-name
                  [NOUSERABEND]
                  [PASSWORD = password ]
                  [TEST]
```

Essential Parameters

FILE: File Number

FILE specifies the file from which the descriptor was logically deleted. Specify a decimal value.

Descriptor: Descriptor Name

DESCRIPTOR identifies the descriptor that was logically deleted and needs to be undeleted. Specify a valid descriptor name.

Optional Parameters

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PASSWORD: File Password

The password of the file from which the descriptor was logically deleted. This parameter is required if the file is password-protected. Specify the password between apostrophes (').

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and vari-

ables. See **Syntax Checking with the TEST Parameter** for more information on using the TEST parameter in ADADBS functions.

Example

The following example undeletes the logically deleted field AA as a descriptor field in file 12 (which requires password XYZPSWD).

```
ADADBS UNDELDE FILE=12 DESCRIPTOR=AA PASSWORD=XYZPSWD
```

95 UNDELFN: Logically Undelete Fields

- Essential Parameter 566
- Optional Parameters 566
- Example 567

The UNDELFN function allows you to logically undelete fields in an Adabas database file that were previously logically deleted using the ADADBS DELFN utility function. Undeleting the fields reestablishes the fields in the FDT for the file.

```
ADADBS UNDELFN  FILE = file-number
                FIELDLIST = ' field-list '
                [PASSWORD = password ]
                [NOUSERABEND]
                [TEST]
```

Essential Parameter

FILE: File Number

FILE specifies the database file number from which the fields should be logically undeleted. The file number may not be the number of a large object (LOB) file; fields in LOB files cannot be deleted, so you should have no reason to undelete them.

FIELDLIST: List of Fields

FIELDLIST specifies a list of one or more fields. At least one field must be specified. If more than one field will be undeleted, separate the field names with commas (,). A maximum of 800 fields may be specified. A field may only be listed once in an ADADBS UNDELFN run.

Descriptor fields cannot be included in the list (they cannot be logically deleted, so there should be no reason to undelete them). Likewise, parent fields of subdescriptor, superdescriptor, hyperdescriptor, phonetic descriptor, and collating descriptor fields cannot be listed.

Optional Parameters

PASSWORD: File Password

PASSWORD specifies the password of the file containing fields to be logically undeleted. This parameter is required if the file is password-protected.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "`utility` TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables. See **Syntax Checking with the TEST Parameter** for more information on using the TEST parameter in ADADBS functions.

Example

In the following example, fields AA, AB, and AC are logically undeleted from file 12.

```
ADADBS UNDELFN FILE=12,FIELDLIST='AA,AB,AC'
```

96 JCL Requirements and Examples

- Collation with User Exit 570
- Requirements to Run ADADBS JCL 570

This section describes the job control information required to run ADADBS on the z/OS operating system and shows an example job stream.

Collation with User Exit

If a collation user exit is to be used during ADADBS ONLINVERT execution, the ADARUN CDXnn parameter must be specified for the utility run.

Used in conjunction with the universal encoding support (UES), the format of the collation descriptor user exit parameter is

```
ADARUN CDXnn= exit-name
```

where

nn	is the number of the collation descriptor exit, a two-digit decimal integer in the range 01-08 inclusive.
exit-name	is the name of the user routine that gets control at the collation descriptor exit; the name can be up to 8 characters long.

Only one program may be specified for each collation descriptor exit. Up to 8 collation descriptor exits may be specified (in any order). See the *Adabas DBA Reference* documentation for more information.

Requirements to Run ADADBS JCL

Data Set	DD Name	Storage	More Information
Associator	DDASSORn	disk	Required only for OPERCOM DDIB or RESETDIB functions with inactive nucleus
ADADBS messages	DDDRUCK	printer	*Messages and Codes*
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADARUN parameters	DDCARD	reader	*Operations*
ADADBS parameters	DDKARTE	reader	

ADADBS JCL Example

Refer to ADADBS in the JOBS data set for this example.

```
//ADADBS    JOB
//*
//*     ADADBS:
//*        DATA BASE SERVICES (BATCH)
//*
//DBS       EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD  <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <=== WORK
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADADBS,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADADBS      REFRESH FILE=1
/*
```

VII ADADEF Utility: Define a Database

This chapter covers the following topics:

Functional Overview
DEFINE: Defining a Database and Checkpoint File
MODIFY: Change Database Options
NEWWORK: Defining a Work File
JCL Requirements and Examples

97 Functional Overview

- Database Components 576
- Checkpoint File 576

The following database characteristics are defined with ADADEF:

- database name and ID
- database components (Associator, Data Storage, and Work)
  - device type
  - size
- checkpoint system file
- database default encodings

Database Components

Each database component (Associator, Data Storage, and Work) must be formatted by the ADAFRM utility before it is defined with ADADEF. The ADADEF utility may also be used to define a new Work data set for an existing database.

Systems using the Recovery Aid feature require a recovery log (RLOG) data set, which must first be formatted with the ADAFRM utility, and then defined using the ADARAI utility.

Checkpoint File

Adabas uses the checkpoint system file to store checkpoint data and user data provided with the Adabas CL and ET commands. It is required and must be specified using the ADADEF DEFINE (database) function.

98 DEFINE: Defining a Database and Checkpoint File

- Essential Parameters 579
- Optional Parameters 580
- Examples 584

The database and the checkpoint file must be defined at the same time.

The database parameters include the required ASSOSIZE, DATASIZE, and WORKSIZE parameters and the optional (non-indented) parameters ASSODEV through WORKDEV shown in the syntax diagram.

The FILE=...,CHECKPOINT,... statement is also required for database definition. The checkpoint file parameters (indented under the FILE statement in the syntax diagram) should be specified immediately following the FILE statement. See the **examples**.

```
ADADEF  DEFINE   ASSOSIZE = size-list
                 DATASIZE = size-list
                 WORKSIZE = size
                 FILE = file-number , CHECKPOINT
                       DSSIZE = size
                       MAXISN = maximum-number-of-records-expected
                       [ACRABN = starting-rabn ]
                       [ASSOPFAC = { Associator-padding-factor | 10 } ]
                       [ASSOVOLUME = ' Associator-extent-volume ' ]
                       [DATAPFAC = { Data-Storage-padding-factor | 10 } ]
                       [DATAVOLUME = ' Data-Storage-extent-volume ' ]
                       [DSDEV = device-type ]
                       [DSRABN = starting-rabn ]
                       [DSREUSE = { NO | YES } ]
                       [ISNSIZE = { 3 | 4 } ]
                       [MAXDS = maximum-Data-Storage-secondary-allocation ]
                       [MAXNI = maximum-normal-index-secondary-allocation ]
                       [MAXUI = maximum-upper-index-secondary-allocation ]
                       [NAME = { ' file-name ' |CHECKPOINT } ]
                       [NIRABN = starting-rabn ]
                       [NISIZE = size ]
                       [UIRABN = starting-rabn ]
                       [UISIZE = size ]
                 [ASSODEV = { device-type-list | ADARUN-device } ]
                 [DATADEV = { device-type-list | ADARUN-device } ]
                 [DBIDENT = { database-id | ADARUN-dbid } ]
                 [DBNAME = { database-name | GENERAL-DATABASE } ]
                 [FACODE = { alpha-EBCDIC-key | 37 } ]
                 [FWCODE = { wide-key | 4095 } ]
                 [MAXFILES = { maximum-number-of-files | 255 } ]
                 [NOUSERABEND]
                 [OVERWRITE]
                 [RABNSIZE = { 3 | 4 } ]
                 [REPTOR = YES | NO ]
                 [TIMEZONE = 'string' ]
                 [UACODE = { alpha-ASCII-key | 437 } ]
                 [UES = { YES | NO } ]
                 [UWCODE = wide-key ]
                 [WORKDEV = device-type-list ]
```

Essential Parameters

ASSOSIZE/ DATASIZE/ WORKSIZE: Database Size

ASSO-/DATA-/WORKSIZE specifies the number of blocks or cylinders to be assigned to the Associator, Data Storage, or Work. A block value must be followed by a "B"; otherwise, the value is assumed to be cylinders.

If the Associator or Data Storage is to be contained on more than one data set, the size of each data set must be specified. If a companion ASSODEV or DATADEV parameter specifies two or more extents, the equivalent ASSOSIZE or DATASIZE parameter must specify the extent sizes as positional operands in the corresponding order (see the **examples**).

The minimum WORKSIZE allowed is 300 blocks.



Note: If ASSOSIZE or DATASIZE is not specified, the ADADEF DEFINE function will not execute. If WORKSIZE is not specified, the function will allocate three (3) cylinders to the Work data set. Because 3 cylinders are usually not enough to start the database, WORKSIZE is considered to be a required parameter.

DSSIZE: Data Storage Size

DSSIZE specifies the number of blocks or cylinders to be assigned to checkpoint/Data Storage. For blocks, the value specified must be followed by a "B" (for example, DSSIZE=80B).

The size of the checkpoint file specified with the DSSIZE and MAXDS parameters depends on

- the amount of ET data to be stored;
- the number of utility runs for which checkpoint information is to be retained;
- the number of user IDs.

FILE . . . CHECKPOINT Parameter

The FILE...CHECKPOINT parameter indicates the file number to be used for the checkpoint system file. This parameter is required; the file number must be 5000 or lower.

Adabas uses the checkpoint system file to store checkpoint data and user data provided with the Adabas CL and ET commands.

MAXISN: Highest ISN to be Used

The highest ISN that may be assigned to the file. The value specified is used to determine the space allocation for the address converter. When determining the MAXISN, consider the importance of ET data and checkpoint data to your site.

Adabas considers ET data to be more important than checkpoint data. As soon as the ET data ISN range in the checkpoint system file is exhausted, the first checkpoint ISN is deleted and given to the ET data. This is an ongoing process. As soon as the MAXISN is reached, a new address converter extent is allocated and given to the checkpoint data. You can delete checkpoint data piece by piece using the Adabas Online System function DELCP.

Note: The way the checkpoint handles data is subject to change in a future release of Adabas.

Optional Parameters

ACRABN/ DSRABN/ NIRABN/ UIRABN: Starting RABN

These parameters may be used to cause allocation for their respective areas to begin with the specified RABN:

- ACRABN for the address converter
- DSRABN for Data Storage
- NIRABN for the normal index
- UIRABN for the upper index

ASSODEV/ DATADEV/ WORKDEV: Device Type

ASSO-/DATA-/WORKDEV specify the device type(s) to be assigned to the Associator, Data Storage, and Work. These parameters are required only if the device type to be used is different from that specified with the ADARUN DEVICE parameter.

WORKDEV, if specified, can only be one device type. If the Associator (ASSODEV) or Data Storage (DATADEV) is to be contained on more than one data set, the device type for each data set must be specified, even if both extents are on the ADARUN DEVICE type.

Space allocation for specified device types must be given in companion ASSOSIZE, DATASIZE, and WORKSIZE parameters on this or another ADADEF statement in the same job. If a ASSODEV or DATADEV parameter specifies more than one extent on the same or different device types (DATADEV=3390,8390, for example), the companion ASSOSIZE or DATASIZE parameter must specify the related extent sizes in corresponding order.

ASSOPFAC/ DATAPFAC: Padding Factor

ASSOPFAC defines the percentage of space in each Associator RABN block to be reserved for later entries (padding space). This space is used for later descriptor extensions or ISN additions. The percentage value specified, which can range 1-90, should be large enough to avoid the overhead caused when block overflow forces splitting of an existing address block into two new blocks. If ASSOPFAC is not specified, ADADEF assumes a padding factor of 10%.

DATAPFAC defines the percentage of space in each Data Storage RABN block to reserve for later entries (padding space). This space is used when changes to an existing data record cause it to need more space in the block; an updated record that no longer fits in the existing block must be moved to another block. The percentage value specified, which can range 1-90, should be large enough to avoid the overhead caused when block overflow forces splitting of an existing address block into two new blocks. If DATAPFAC is not specified, ADADEF assumes a padding factor of 10%.

ASSOVOLUME/ DATAVOLUME: Extent Volume



Note: Values for ASSOVOLUME and DATAVOLUME must be enclosed in apostrophes.

ASSOVOLUME specifies the volume on which the file's Associator space (that is, the AC, NI, and UI extents) is to be allocated.

DATAVOLUME specifies the volume on which the file's Data Storage space (DS extents) are allocated.

If the requested number of blocks cannot be found on the specified volume, ADADEF retries the allocation while disregarding the ASSOVOLUME or DATAVOLUME parameter value.

If ACRABN, UIRABN, or NIRABN is specified, ADADEF ignores the ASSOVOLUME value when allocating the corresponding extent type.

If DSRABN is specified, DATAVOLUME is ignored for the related file.

If ASSOVOLUME and/or DATAVOLUME are not specified, the file's Associator and/or Data Storage space, respectively, is allocated according to ADADEF's default allocation rules.

DBIDENT: Database Identifier

DBIDENT specifies the identification number to be assigned to the database. A value in the range 1-65535 may be specified. If this parameter is omitted, the value specified with the ADARUN DBID parameter is used.

If multiple databases are to be established, the DBIDENT parameter is required in order to uniquely identify each database.

DBNAME: Database Name

DBNAME is the name to be assigned to the database. This name appears in the title of the Database Status Report produced by the ADAREP utility. A maximum of 16 characters may be specified. Enclose the name in single quotation marks if the name includes any special characters other than dashes, or if the name contains embedded blanks.

If this parameter is omitted, a default value of "GENERAL-DATABASE" is assigned.

DSDEV: Device Type for Data Storage

DSDEV specifies the device type to be used for the checkpoint file's Data Storage. There is no default value; if DSDEV is not specified, an arbitrary device type is used.

DSREUSE: Storage Reusage

DSREUSE indicates whether space which becomes available in the checkpoint file is to be reused. The default is YES.

FACODE: Encoding for Alphanumeric Fields

The FACODE parameter specifies the default encoding for alphanumeric fields for all files in the database. The encoding must be derived from EBCDIC encoding; that is, X'40' is the space character. Modal or shift-type double-byte character set (DBCS) encodings are supported; fixed type DBCS (DBCS-only) encodings are not supported. The default encoding key is 37.

The purpose of the database-wide setting is to serve as a default when loading files. Once loaded, the encoding for a file is stored in its FCB.

You can change the default encoding set in this parameter using the ADADEF MODIFY function. Changing the database-wide setting does not affect files already loaded.

FWCODE: Encoding for Wide-Character Fields

The FWCODE parameter specifies the default encoding for wide-character (W) format fields for all files in the database. The default encoding is 4095; that is, Unicode.

The FWCODE parameter can be used to set a wide-character encoding that defines the superset of code points of all user encodings. For example, Unicode encompasses about 50,000 code points as opposed to Host-DBCS and Shift-JIS with about 10,000 code points each.

The purpose of the database-wide setting is to serve as a default when loading files. Once loaded, the encoding for a file is stored in its FCB.

You can change the default encoding set in this parameter using the ADADEF MODIFY function. Changing the database-wide setting does not affect files already loaded.

ISNSIZE: 3- or 4-Byte ISN

ISNSIZE indicates whether ISNs in the file are 3 or 4 bytes long. The default is 3 bytes.

MAXDS/ MAXNI/ MAXUI: Maximum Secondary Allocation

MAXDS/NI/UI specify the maximum number of blocks per secondary extent for Data Storage, the normal index, and the upper index, respectively. The value specified may either be in blocks and then be followed by a "B" (for example, MAXDS=8000B) or cylinders.

MAXFILES: Highest File Number

MAXFILES specifies the maximum number of files that can be loaded into the database. The minimum value for this parameter is 3. The highest value permitted is 5000 or one less than the ASSOR1 block size, whichever is lower. For example, 2543 is the highest MAXFILES value for a database whose ASSOR1 is stored on a 3390 DASD.

The value specified determines the number of file control blocks and field definition tables to be allocated when the database is being established. Each file control block requires one Associator block and each field definition table requires four Associator blocks.

If this parameter is omitted, a value of 255 is assigned.

Once the database has been established, the value for MAXFILES may be changed only by executing the REORASSO or REORDB functions of the ADAORD utility.

NAME: Name of the Checkpoint File

NAME specifies the name for the checkpoint file being defined. This name appears on the Database Status Report produced by the ADAREP utility. The maximum number of characters permitted is 16. The default file name is CHECKPOINT.

NISIZE: Normal Index Size

NISIZE specifies the number of blocks or cylinders to be assigned to the normal index. For blocks, the value specified must be followed by "B" (for example, NISIZE=80B).

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

OVERWRITE: Overwrite Existing Database

Specify OVERWRITE to write over an existing database. OVERWRITE cannot be specified when creating a database with newly formatted data sets.

RABNSIZE: 3- or 4-Byte RABN

RABNSIZE specifies the length of RABNs in the database. Specify 3 for 24-bit RABNs or 4 for 31-bit RABNs. The default is 3.

REPTOR: Set the Event Replicator Server

The `REPTOR` parameter is used with ADADEF DEFINE to specify whether a normal Adabas database is being defined, or whether an Event Replicator Server is being defined. Possible values are "YES" or "NO" (the default).

Specify `REPTOR=NO` for a normal Adabas database. Specify `REPTOR=YES` for an Event Replicator Server.

TIMEZONE: Set the default time zone

The TIMEZONE parameter is used with ADADEF DEFINE to set the default time zone of the database.

Possible values are blank (default) or a string which specifies the name of a valid Adabas-supported time zone. For information about the time zone names in the Adabas time zone library, read *Supported Time Zones*, in the *Adabas DBA Tasks Manual*.



Note: When the default time zone is used, the ADADEF utility job must include the TZINFO data definition statement (DD statement) identifying the TZ library.

UACODE: User Encoding for Alphanumeric Fields

The parameter UACODE specifies the default encoding for alphanumeric fields for ASCII users. The encoding must be derived from ASCII encoding; that is, X'20' is the space character. Encodings for multiple-byte character sets are supported. The default encoding is 437.

The UACODE value is not stored in the file being loaded.

You can override the default encoding set in this parameter for a user session using the OP command. You can change it generally using the ADADEF MODIFY function.

UES: Universal Encoding Support

Setting the parameter UES activates universal encoding support for the database. Any valid xxCODE parameter (FACODE, FWCODE, UACODE, UWCODE) implicitly sets UES=YES.

To deactivate UES, you must explicitly set UES=NO.

You can change the default setting of this parameter generally using the ADADEF MODIFY function.

UISIZE: Upper Index Size

UISIZE specifies the number of blocks or cylinders to be assigned to the upper index. For blocks, the value specified must be followed by "B" (for example, UISIZE=80B).

UWCODE: User Encoding for Wide-Character Fields

The UWCODE parameter specifies the user encoding for wide-character (W) format fields. If the parameter is not specified, the default value is the current value of FWCODE.

The purpose of the database-wide setting is to serve as a default when loading files. Once loaded, the encoding for a file is stored in its FCB.

You can override the default encoding set in this parameter for a user session using the OP command. You can change the default setting generally using the ADADEF MODIFY function. Changing the database-wide setting does not affect files already loaded.

Examples

Example 1:

```
ADADEF DEFINE
ADADEF  ASSOSIZE=200,DATASIZE=600,WORKSIZE=50
ADADEF  DBIDENT=1,DBNAME=DATABASE-1
ADADEF  MAXFILES=150
ADADEF  FILE=1,CHECKPOINT
ADADEF  NAME='DB1-CHECKPOINT',MAXISN=5000
ADADEF  DSSIZE=2,NISIZE=50B,UISIZE=10B
```

The Associator, Data Storage and Work sizes are equal to 200, 600 and 50 cylinders, respectively. The numeric identifier for the database is 1 and the database name is DATABASE-1. The maximum number of files (and the highest file number) that may be loaded into the database is 150. File 1 is to be reserved for the Adabas checkpoint file. The name of the first system file is to be DB1-CHECKPOINT. The Data Storage size for this file is to be 2 cylinders; the normal index size 50 blocks; the upper index size 10 blocks; and the MAXISN is to be 5000.

Example 2:

```
ADADEF DEFINE
ADADEF  ASSODEV=3390,DATADEV=3390,3390,WORKDEV=3390
ADADEF  ASSOSIZE=100,DATASIZE=200,300,WORKSIZE=25
ADADEF  DBIDENT=2,DBNAME='DATABASE_2'
ADADEF  MAXFILES=255
ADADEF  FILE=255,CHECKPOINT,MAXISN=5000
ADADEF  DSSIZE=3,NISIZE=100B,UISIZE=20B
```

The Associator is to be contained on a 3390 device type, and occupies 100 cylinders. Data Storage comprises two data sets: the first data set is 200 cylinders contained on the first DATADEV (3390) device type, and the second data set is 300 cylinders contained on the second DATADEV (3390) device type. The Work space is 25 cylinders on the WORKDEV device (3390).

The numeric identifier for the database is 2, and the database name is DATABASE_2. A maximum of 255 files may be loaded into the database. An Adabas checkpoint file is loaded during this step.

99 MODIFY: Change Database Options

- Optional Parameters 588
- Examples 590

The MODIFY function is used to modify properties set for the database using ADADEF DEFINE. At least one of the optional parameters must be specified.

Changing the FACODE, FWCODE, or UWCODE parameters does not affect files already loaded since the actual encoding of their fields is stored in the FCB. The purpose of the database-wide setting is to serve as a default when loading files.

```
ADADEF  MODIFY  [FACODE = alpha-EBCDIC-key ]
                [FWCODE = wide-key ]
                [NOUSERABEND]
                [REPTOR = YES | NO ]
                [TIMEZONE = 'string' ]
                [UACODE = { alpha-ASCII-key | 437 } ]
                [UES = { YES | NO } ]
                [UWCODE = wide-key ]
```

Optional Parameters

FACODE: Encoding for Alphanumeric Fields

The FACODE parameter specifies the default encoding for alphanumeric fields for all files in the database. The encoding must be derived from EBCDIC encoding; that is, X'40' is the space character. Modal or shift-type double-byte character set (DBCS) encodings are supported; fixed type DBCS (DBCS-only) type encodings are not supported. The default encoding key is the current setting.

The purpose of the database-wide setting is to serve as a default when loading files. Once loaded, the encoding for a file is stored in its FCB. Changing the database-wide setting does not affect files already loaded.

FWCODE: Encoding for Wide-Character Fields

The FWCODE parameter specifies the default encoding for wide-character (W) format fields for all files in the database. The default encoding is the current setting.

The FWCODE parameter can be used to set a wide-character encoding that defines the superset of code points of all user encodings. For example, Unicode encompasses about 50,000 code points as opposed to Host-DBCS and Shift-JIS with about 10,000 code points each.

The purpose of the database-wide setting is to serve as a default when loading files. Once loaded, the encoding for a file is stored in its FCB. Changing the database-wide setting does not affect files already loaded.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend

after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

REPTOR: Set the Event Replicator Server

The REPTOR parameter is used with ADADEF MODIFY to indicate whether the database that is running is an Adabas database or an Event Replicator Server. Possible values are "YES" or "NO" (default).

For a normal Adabas database, the REPTOR parameter must always be set to "NO". When an Event Replicator Server is running, the REPTOR parameter must be set to "YES".

TIMEZONE: Set the default time zone

The TIMEZONE parameter is used with ADADEF MODIFY to set the default time zone of the database.

Possible values are blank (default) or a string which specifies the name of a valid Adabas-supported time zone. For information about the time zone names in the Adabas time zone library, read *Supported Time Zones*, in the *Adabas DBA Tasks Manual*.



Note: When the default time zone is used, the ADADEF utility job must include the TZINFO data definition statement (DD statement) identifying the TZ library.

UACODE: User Encoding for Alphanumeric Fields

The parameter UACODE specifies the default encoding for alpha fields for ASCII users. The encoding must be derived from ASCII encoding; that is, X'20' is the space character. Encodings for multiple-byte character sets is supported. The default encoding is the current setting.

The UACODE setting is not stored in the loaded file. You can override this encoding for a user session with the OP command.

UES: Universal Encoding Support

The parameter UES can be used to enable or disable universal encoding support for an existing database. Disabling is only possible if no files are loaded with wide-character (W) format fields.

Any valid xxCODE parameter (FACODE, FWCODE, UACODE, UWCODE) implicitly sets UES=YES.

To deactivate UES, you must explicitly set UES=NO.

UWCODE: User Encoding for Wide-Character Fields

The UWCODE parameter specifies the user encoding for wide-character (W) format fields. If the parameter is not specified, the default value is the current setting.

The purpose of the database-wide setting is to serve as a default when loading files. Once loaded, the encoding for a file is stored in its FCB. Changing the database-wide setting does not affect files already loaded.

You can override the default encoding for a user session with the OP command.

Examples

Example 1:

Disable universal encoding support for an existing database. The database contains no files with wide (W) format.

```
ADADEF MODIFY UES=NO
```

Example 2:

Change the default encoding for wide-character (W) format fields for all files in the database from the current setting to code page 835 (traditional Chinese host double byte including 6204 user-defined characters).

```
ADADEF MODIFY FWCODE=835
```

Files already loaded are not affected by this change since the actual encoding of their fields is stored in the FCB. The purpose of the database-wide setting is to serve as a default when loading files.

Example 3:

Set US Mountain Standard Time (MST) as the default time zone of the database.

```
ADADEF MODIFY TIMEZONE='America/Denver'
```

Example 4:

Remove the default time zone from the database.

```
ADADEF MODIFY TIMEZONE=' '
```

100 NEWWORK: Defining a Work File

- Essential Parameter 592
- Optional Parameters 592
- Example 593

The following parameters are used for Work data set definition:

```
ADADEF  NEWWORK   WORKSIZE = size
                  [NOUSERABEND]
                  [WORKDEV = device-type-list ]
```

Notes:

1. The Adabas nucleus must not be active during this function, and the old Work must be specified in the JCL.
2. The ADADEF NEWWORK function cannot be executed if a pending autorestart exists.

Essential Parameter

WORKSIZE: Work Data Set Size

The number of blocks or cylinders to be assigned to the Work data set.

Optional Parameters

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

WORKDEV: Device Type

The device type to be assigned to the new Work data set.

This parameter is required only if the device type to be used is different from that specified by the ADARUN DEVICE parameter.

Example

A new Work data set is defined with a size of 50 cylinders. The device type is obtained from the ADARUN DEVICE parameter.

```
ADADEF NEWWORK
ADADEF WORKSIZE=50
```

101 JCL Requirements and Examples

- Requirements to Run ADADEF JCL 596

This section describes the job control information required to run ADADEF on the z/OS operating system and shows examples of job streams.

Requirements to Run ADADEF JCL

Data Set	DD Name	Storage	More Information
Associator	DDASSORn	disk	
Data Storage	DDDATARn	disk	
Work (Current)	DDWORKR1 DDWORKR4	disk	
ADARUN parameters	DDCARD	reader	*Operations*
ADADEF parameters	DDKARTE	reader	
ADARUN messages	DDPRINT	printer	*Messages and Codes*
Time zone file	TZINFO	disk	Required with TIMEZONE parameter
ADADEF messages	DDDRUCK	printer	*Messages and Codes*

ADADEF JCL Examples

Define Database

```
//ADADEF    JOB
//*
//*   ADADEF:
//*      DEFINE THE PHYSICAL LAYOUT OF THE DATABASE
//*      DEFINE THE NUCLEUS SYSTEM FILE: CHECKPOINT FILE
//*
//DEF       EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD      <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <=== WORK
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADADEF,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADADEF DEFINE DBNAME=EXAMPLE-DB,DBIDENT=YYYYY
ADADEF        ASSOSIZE=100,DATASIZE=200,WORKSIZE=40
ADADEF        MAXFILES=120
```

```
*
ADADEF FILE=19,CHECKPOINT
ADADEF   NAME='CHECKPOINT',MAXISN=5000
ADADEF   DSSIZE=100B,NISIZE=3B,UISIZE=3B
/*
```

Refer to ADADEF in the JOBS data set for this example.

Define New Work

```
//ADADEFNW  JOB
//*
//*    ADADEF: DEFINE NEW WORK
//*
//DEF       EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD      <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <=== WORK
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADADEF,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADADEF NEWWORK WORKSIZE=60,WORKDEV=eeee
/*
```

Refer to ADADEFNW in the JOBS data set for this example.

VIII ADADRU Utility: Database Storage Usage Reports

The ADADRU utility currently can be used to produce summary and detail reports about database storage usage.

This chapter covers the following topics:

Functional Overview
SPACEUSAGE: Database File Space Usage
JCL Requirements and Examples

102 Functional Overview

- Spanned Record Handling 602
- Syntax Checking with the TEST Parameter 604
- Global OUTPUT and ACODE Parameter Usage 605

The ADADRU utility currently includes a single utility function, ADADRU SPACEUSAGE, that you can use to produce summarized and detailed reports about database file space usage.

ADADRU verifies that the files specified in the utility run are not in an inconsistent state after being processed simultaneously by utilities such as ADALOD UPDATE or ADAORD, that heavily update or reorganize the database. Updates to the file are permitted while the ADADRU utility is processing. The utility reads through the Associator (ASSO) and Data Storage (DATA) containers for each file requested, and gathers information about file extents, and RABN usage statistics like RABN used space, and RABN free space.

ADADRU can report on an individual file in a database, specifically selected files in the database, or all the files in a database. The individual file processing can also be limited to ASSO information, DATA information or a combination of ASSO and DATA RABNs.

Reports are generated for every file requested in the ADADRU run. If a file resides on more than one device type, file reports are produced for every device type.

If you request detail reports (if parameter PFILESPACE, PASSOSPACE, or PDATASPACE is set to "DETAIL"), the "File ASSO/DATA Detail Report" is produced, as appropriate. The report can be directed to a specific file using the DD statement with the name DDDETAIL. (For more information, read ***JCL Requirements and Examples*** , elsewhere in this chapter. Detail reports may be very large because information may be printed for every RABN in a file (or a specific RABN range).

Spanned Record Handling

During its sequential read of RABNs, if ADADRU detects the primary segment of a spanned record in the Data Storage area, it will use the secondary address converter (AC2) to find and read all secondary segments of the spanned record to determine the full record length and number of its segments. The search for the secondary segments starts by searching and reading the corresponding AC2 block, determining the Data Storage RABN containing this secondary segment, reading the Data Storage RABN, and finally searching for the secondary spanned record identified by the RABN. When all segments of the spanned record are found, ADADRU will start its processing with the RABN of the primary segment.

ADADRU does not have any special parameters for spanned record handling.

Spanned records are reported in the ADADRU output in the following ways.

1. The FCB file options section of the report describes whether the file supports the use of spanned records. The "Spanned Records Support" statistic in this section of the ADADRU report provides this information ("Yes" means spanned records are supported by the file; "No" indicates they are not supported).

```
File options (from FCB):
------------------------
Date / Time Loaded ..................... 2010-04-28 10:01:49.79
Date / Time of last update ............. 2010-05-04 07:07:08.47
Date / Time of last physical restructure 2010-04-28 10:01:49.79

ISN Reusage ............................ No
DATA Storage Reusage ................... Yes
MIXDSDEV ............................... No
Spanned Records Support ................ Yes
Index compressed ....................... No
2-Byte MU/PE Index ..................... No
Highest Index Level .................... 3

LOB fields ............................. No
LOB file ............................... No
Expanded ............................... No
```

2. A section providing spanned record statistics is included in the "Data Storage Summary" section of the report only if the file supports spanned records. Here is a sample of this section:

```
Spanned Record:
   Average record length ..............................  15072
   Minimum record length ..............................  15068
   Maximum record length ..............................  15076
   Number of records with minimum length ..............  100
   Number of records with maximum length ..............  100
   Average numbers of segments ........................  3
   Minimum numbers of segments ........................  3
   Maximum numbers of segments ........................  3
   Number of Primary Spanned Records ..................  499
   Number of Secondary Spanned Records.................  998
   Number of AC2 blocks read ..........................  2
   Data Storage blocks with secondary segments read ....  998
```

3. A section describing the spanned record length and increment is included in the "Distribution Reports Data Storage" section of the report only if the file supports spanned records. If the file supports spanned records but does not contain any spanned records, ADADRU prints warning message DRU163W instead and sets the return code to "4" (ADARSP004). Here is a sample of this section:

```
Spanned records length; Increment = Auto
-------------------------------------------------
I        Length       I         Number           I
I      of record      I       of records         I
I---------------------I--------------------------I
I   15055 -  15080    I              499         I
-------------------------------------------------
```

4. In the "Data Storage Detail" section of the report the column header "Spanned" is always printed, regardless of whether or not the file supports spanned records. The "Spanned" column is split into two columns called "Num" and "Flg". The "Num" column provides the number of spanned record segments in each Data Storage RABN. The "Flg" columns values can be "P", "S", "L", or a blank, identifying the segment of the spanned record stored at the RABN:
   - "P" indicates that the current RABN contains the primary segment of the spanned record.
   - "S" indicates that the current RABN contains a secondary segment of the spanned record.
   - "L" indicates that the current RABN contains the last segment of the spanned record.
   - A blank indicates that the current RABN does not contain any segment of a spanned record.

   Here is a sample of this section of the report:

```
File 5                                                                  Page 1     ↩
------------------------------------------------------------------------------- ↩
I   RABN   I   %    I Block  I Free space I Num I  Spanned  I Record lengths  I ↩
I         I  Used  I Length I------------I Rec I------------I-----------------I ↩
I          I        I        I Gap I Pad  I     I Num I Flg I Min I Max I Avg I ↩
I----------I--------I--------I-----I------I-----I------I-----I-----I-----I-----I ↩
        350   100       5064      0     0      1    1    P     5060  5060  5060 ↩
        351   100       5064      0     0      1    1    S     5060  5060  5060 ↩
...
```

Syntax Checking with the TEST Parameter

The ADADRU functions include a syntax-checking-only mode. When the TEST parameter is specified, the ADADRU function syntax is checked, but the function is not performed.

```
ADADRU SPACEUSAGE TEST
```

Global OUTPUT and ACODE Parameter Usage

The OUTPUT and ACODE parameters can be specified globally for all ADADRU SPACEUSAGE reports requested in the run or they can be specified within a specific ADADRU SPACEUSAGE statement, where they would pertain only to that specific SPACEUSAGE report request. The syntax for declaring them globally is the same as the syntax within the ADADRU SPACEUSAGE statement:

```
ADADRU OUTPUT = 'XML [ ,UTF8 ] '
                [ ACODE= '{ GCB | dbcodepg } [ ,{ GCB | FCB | fncodepg } [ ,{ GCB | cpcodepg } ] ]' ]
```

In the following example, the OUTPUT and ACODE parameters apply to all ADADRU SPACEUSAGE reports in the run. In this example two ADADRU SPACEUSAGE reports are requested; both would return XML output:

```
AADRU OUTPUT=(XML,UTF8),ACODE=...
    ADADRU SPACEUSAGE FILE=...
    ADADRU SPACEUSAGE FILE=...
```

In the following example, the OUTPUT parameter applies only to the second ADADRU SPACEUSAGE report in the run. In this example two ADADRU SPACEUSAGE reports are requested; only the second report return XML output:

```
ADADRU SPACEUSAGE FILE=...
    ADADRU SPACEUSAGE FILE=...,OUTPUT='XML'...
```

103 SPACEUSAGE: Database File Space Usage

- SPACEUSAGE File Group Processing 608
- Syntax 612
- Sample Reports 619

The ADADRU SPACEUSAGE utility function can be used to produce summarized and detailed reports about database file space usage. The database files space usage reports show information such as:

- RABN space usage;
- RABN free space for either or both the Associator (normal index) or Data Storage, with an indication of the available padding space inside the free space;
- Free blocks;
- Average file record lengths; and
- Average file record counts.

A summary report is always produced by the ADADRU SPACEUSAGE utility function. You can request that detail reports also be produced by specifying "DETAIL" for the PFILESPACE, PASSOSPACE, or PDATASPACE parameters; this will produce detail reports in addition to the summary report.

You can use the information in these reports when deciding if database or file level reorganization is necessary. They can be used to help you optimize block sizes, decrease file fragmentation, decrease file space usage, and increase the overall performance of the database files.

SPACEUSAGE File Group Processing

The FILE parameter of the ADADRU SPACEUSAGE utility function is an optional parameter that identifies the files that should be processed by the utility. More than one FILE parameter can be specified in an ADADRU SPACEUSAGE request. If the FILE parameter is omitted, all files in the database are processed for the report. Each FILE parameter specifies a *file group*. ADADRU SPACEUSAGE reports are produced for each file group; one report set (summary report only or a combination of detail and summary reports) is produced for each file group in an ADADRU SPACEUSAGE run.

For each instance of the FILE parameter (each file group), specific FROMRABN, TORABN, PFILESPACE, PASSOSPACE, PDATASPACE, SAMPLING, INCRUSAGE, INCRNUMREC, INCRRECLEN, OUTPUT, LOBGROUP, EXPGROUP and ERRLIM parameters may be specified, although none of these parameters are required. When specified, these parameter should immediately follow the FILE parameter (file group) to which they apply.

When ADADRU processes a SPACEUSAGE report request for multiple file groups, it analyzes all parameters of all file groups up to the next ADADRU SPACEUSAGE specification or to the end of the parameter list. File processing is started only after consolidation of all parameters from all file groups of the current SPACEUSAGE function. Processing is normally performed by file, in ascending order by file number. The exception is expanded files which are processed in logical sequence of their parts when the EXPGROUP parameter specified.

Every file is processed only once with one consolidated list of parameters and the resulting reports represent those consolidated parameters. Consider the following example:

```
ADADRU SPACEUSAGE  FILE=2-5,7,9,15-20,PDATASPACE=SUMMARY,INCRUSAGE=5
ADADRU    FILE=3,PDATASPACE=DETAIL,INCRUSAGE=2
```

The result of a run of these ADADRU SPACEUSAGE specifications are as follows:

- No report is produced for Associator (ASSO) usage.
- For all specified files, except repeated file 3, only the summary report for Data Storage (DATA) usage is produced, using the parameter setting of INCRUSAGE=5.
- For repeated file 3, both the summary and detailed reports are produced for Data Storage (DATA) usage, using the parameter setting of INCRUSAGE=2. In this case, the summary report is produced because one is always produced when a detail report is requested.

During consolidation, the general rule is to choose the more detailed value of every parameter. A list of the consolidated parameters used in reporting is printed in front of the report.

The rest of this section describes how specific ADADRU SPACEUSAGE parameters are consolidated:

- SAMPLING Parameter Consolidation
- INCRUSAGE Parameter Consolidation
- INCRNUMREC/INCRRECLEN Parameter Consolidation
- FROMRABN/TORABN Parameter Consolidation
- DETAIL/SUMMARY Parameter Consolidation
- LOBGROUP/EXPGROUP Parameter Consolidation
- Consolidation Example

SAMPLING Parameter Consolidation

If the same file is specified in several file groups and different values of the SAMPLING parameter are included in these file groups, then the *largest* specified SAMPLING value is used in the consolidated parameters for the file. If SAMPLING is omitted in one group and is included in another group, then the specified value of the SAMPLING parameter is used in the consolidated parameters for the file. The default value is "100".

INCRUSAGE Parameter Consolidation

If the same file is specified in several file groups and different values of the INCRUSAGE parameter are included in these file groups, then the *smallest* specified value is used in the consolidated parameters for the file. If INCRUSAGE is omitted in one group and is included in another group, then the specified value of the INCRUSAGE parameter is used in the consolidated parameters for the file. The default value is "5".

INCRNUMREC/INCRRECLEN Parameter Consolidation

If the same file is specified in several file groups and different values of the INCRNUMREC or INCRRECLEN parameters are included in these file groups, then the *smallest* specified values are used in the consolidated parameters for the file. If INCRNUMREC or INCRRECLEN is omitted in one group and is included in another group, then the specified value of the INCRNUMREC/INCRRECLEN parameter is used in the consolidated parameters for the file. The default value is "AUTO".

FROMRABN/TORABN Parameter Consolidation

The FROMRABN and TORABN parameters are designed to limit the volume of processed information and the size of the produced report. You can specify several FROMRABN/TORABN groups for ASSO and DATA in the same ADADRU SPACEUSAGE run. Where specifications overlap, they are combined into one range. Otherwise, the ranges are concatenated in parameters. The maximum number of FROMRABN/TORABN ranges that can be processed for one file is eight.

If the FROMRABN/TORABN parameters are omitted in one file group (indicating that all RABNs should be processed), but are specified in other file groups for the same file, the specified FROMRABN/TORABN ranges are used for the file. Subsequent file groups including the file will only be allowed to add specific RABNs -- requests to reset the range to all RABNs will be ignored.

ADADRU SPACEUSAGE processes only RABNs belonging to the file. If some RABNs from specified FROMRABN/TORABN ranges are outside the range for any given file extent, ADADRU will not check them. If none of the specified FROMRABN/TORABN ranges belong to given file, ADADRU issues an error message.

DETAIL/SUMMARY Parameter Consolidation

If SUMMARY and DETAIL values are specified differently for the same file in different file groups, both parameters are used for the report.

LOBGROUP/EXPGROUP Parameter Consolidation

If the LOBGROUP or EXPGROUP parameters are specified for the file in any file group, they are used in the consolidated parameters for the file.

Consolidation Example

In the following example, file 5 is specified in five file groups in the same ADADRU run.

```
ADADRU SPACEUSAGE FILE=5,INCRUSAGE=15,PASSOSPACE=DETAIL
ADADRU            FROMRABN=100,TORABN=250

ADADRU            FILE=5,INCRUSAGE=10,PDATASPACE=SUMMARY
ADADRU            FROMRABN=1000,TORABN=1250

ADADRU            FILE=5,PASSOSPACE=SUMMARY,
ADADRU            FROMRABN=150,TORABN=500

ADADRU            FILE=5,INCRUSAGE=2,PASSOSPACE=SUMMARY
ADADRU            FROMRABN=750,TORABN=1500

ADADRU            FILE=5,INCRUSAGE=25,PDATASPACE=SUMMARY
ADADRU            INCRNUMREC=100
```

As a result, file 5 will be processed with the following consolidated parameters:

```
ASSO:

Report=DETAIL + SUMMARY
INCRUSAGE=2
FROMRABN=100,TORABN=500
 FROMRABN=750,TORABN=1500
 + other parameters with default values.

DATA:

Report=SUMMARY
INCRUSAGE=10
INCRNUMREC=100
FROMRABN=1000,TORABN=1250
+ other parameters with default values
```

Syntax

```
ADADRU SPACEUSAGE  [FILE = { file | file1 - file2 [, file | file1 - file2 ]... } ]
                       [PFILESPACE = { SUMMARY | DETAIL} ]
                       [PASSOSPACE = { SUMMARY | DETAIL}]
                         [FROMRABN = blknum ] [TORABN = blknum ]
                       [PDATASPACE = { SUMMARY | DETAIL}]
                         [FROMRABN = blknum ] [TORABN = blknum ]
                       [SAMPLING = { percentage | 100 } ]
                       [INCRUSAGE = { percentage | 5 } ]
                       [INCRNUMREC = { records | calculated } ]
                       [INCRRECLEN = { bytes | calculated } ]
                       [OUTPUT = 'XML [ ,UTF8 ] ' ]
                         [ ACODE= '{ GCB | dbcodepg } [ ,{ GCB | FCB | fncodepg } [ ,{ GCB | cpcodepg } ] ]' ]
                       [LOBGROUP]
                       [EXPGROUP]
                       [ERRLIM = { error-threshold-count | 1 } ]
                   [NOUSERABEND]
                   [TEST]
                   [ABEND34]
```

This section describes the syntax and parameters of the SPACEUSAGE function.

- Essential Parameters
- Optional Parameters and Subparameters
- Examples

Essential Parameters

There are no required parameters for the ADADRU SPACEUSAGE utility function. If no parameters are specified, summary space usage reports are produced for both the Associator and Data Storage areas of all files in the database. In this case, no XML output is produced and any LOB group chains that might be present in the database files are processed and reported in their physical file sequences. Expanded files processed are presented in the report in the same logical sequence as they are in the expanded file chain, starting with the anchor file. The defaults for all other parameters are used.

Optional Parameters and Subparameters

ABEND34: Change User Abend 35 to 34

This optional parameter can be used to change a user abend 35 to user abend 34 if a utility error occurs. This ensures that a dump is produced when the utility terminates abnormally.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADADRU run.

ACODE: Identify Code Page for XML UTF-8 Output

Use the ACODE parameter only if the UTF8 option is specified on the OUTPUT parameter. This parameter identifies the code page to which the database name, file names, and checkpoint names should be converted. This parameter is specified in three parts, in a specific order, and with each part separated by a comma from the others:

```
ACODE='database-name-codepage,file-name-codepage,checkpoint-name-codepage'
```

- The *database-name-codepage* can either be the literal "GCB" or a valid code page number (*dbcodepg*).
- The *file-name-codepage* can either be the literal "GCB", the literal "FCB", or a valid code page number (*fncodepg*).
- The *checkpoint-name-codepage* can either be the literal "GCB" or a valid code page number (*cpcodepg*)

A setting of "GCB" for any part indicates that the code page for that part should be acquired from the code page in the GCB (the GCB ACODE setting). A setting of "FCB" for the *file-name-codepage* indicates that the code page for converting file names in the XML document to UTF-8 should be acquired from the code page setting in the FCB (the FCB ACODE setting).

If code pages for all three parts of the ACODE parameter are being specified, they must be specified in the order shown above. However, if you only need to specify the code page of the database name conversion, you only need to specify the first part. If you only need to specify the code page of the database name and file name conversions, you only need to specify the first two parts. If you do not need to specify the code page for an earlier part in the sequence, simply specify a comma for that part instead. For example, if you only need to specify the code page setting for the checkpoint name UTF-8 conversion, you might specify:

```
ACODE=',,37'
```

Note the two commas specified prior to the checkpoint code page number, 37. These identify the placement of the blank database name and file name code page specifications.

If the ACODE parameter is specified but the OUTPUT parameter is omitted, a setting of `OUTPUT='XML,UTF8'` is assumed.

ERRLIM: Maximum Number of Errors Accepted

This optional parameter specifies the maximum number of errors that can occur before ADADRU processing terminates. Valid values range from 1 through 255. The default is 1.

The ERRLIM parameter is defined within a file group.

EXPGROUP: Expanded File Chain Processing

Specify this optional parameter within a file group if you want expanded files processed and presented in the report in the same logical sequence as they are in the expanded file chain, starting with the anchor file. If just one file from the expanded file chain is specified in the file group, but the EXPGROUP parameter is specified as well, ADADRU will process the every file from expanded files chain.

If this parameter is not specified, expanded files are processed without looking for expanded file associations. The EXPGROUP option is the default when all files in a database are requested for the ADADRU SPACEUSAGE report.

If no expanded files are encountered in the files specified for an ADADRU SPACEUSAGE run, the presence of the EXPGROUP parameter has no effect on the report.

The EXPGROUP parameter is defined within a file group.

FILE: Files for the Report

When specified, the FILE parameter must be the first parameter specified after the SPACEUSAGE keyword in an ADADRU SPACEUSAGE utility job. If you omit this optional parameter, all files in the database are analyzed and included in the SPACEUSAGE report; this is the default.

Use the FILE parameter to identify files in the database that should be analyzed for the SPACEUSAGE report. You can specify a single file number, a series of file numbers, or both single files and file series in an ADADRU SPACEUSAGE run. A maximum of 32 single files or file series can be specified in the FILE parameter.

When you specify a series of file numbers, use dashes between the file numbers. The following example would select files 2, 3, 4, and 5 for the report.

```
FILE=2-5 ↩
```

When you specify multiple single file numbers or a combination of single file numbers and file series, separate them with commas. The following example would select files 2, 3, 4, 5, 11, 13, 14, 15, 16, and 18 for the report.

```
FILE=2-5,11,13-16,18
```

Overlapping or repeated file numbers are allowed. The following example is a valid specification:

```
FILE=2-8,1-4,5,6-11
```

In this example, files 2, 3, 4, 5, 6, 7, and 8 are requested in more than one of the FILE parameter specifications. The repeated files will be presented in the SPACEUSAGE report only once, in sorted order.

The FILE parameter can be specified more than once in ADADRU SPACEUSAGE request. Each FILE parameter specifies a *file group*. For each instance of the FILE parameter (each file group), specific FROMRABN, TORABN, PFILESPACE, PASSOSPACE, PDATASPACE, SAMPLING, INCRUSAGE, INCRNUMREC, INCRRECLEN, OUTPUT, LOBGROUP, EXPGROUP and ERRLIM parameters may be specified, although none of these parameters are required. When specified, these parameter should immediately follow the FILE parameter (file group) to which they apply. For more information, read ***SPACEUSAGE File Group Processing***, elsewhere in this section.

FROMRABN: Data Storage Block Number

The RABN of the Data Storage or Associator block where report processing should begin. It is often paired with the TORABN parameter to specify a range of RABN blocks.

If the FROMRABN parameter is omitted and the TORABN parameter is specified, processing is performed from the beginning of the file to the block specified by the TORABN parameter. If the FROMRABN parameter is specified and the TORABN parameter is omitted, processing is performed from the FROMRABN block to the end of the file. The FROMRABN parameter can only be specified as a subparameter of the PASSOSPACE or PDATASPACE parameters.

The FROMRABN parameter is applicable only if a file group consists of a single file. Only one FROMRABN/TORABN range can be specified for a single file group.

The maximum number of FROMRABN/TORABN ranges that can be specified for the same file in a single ADADRU SPACEUSAGE run is eight. If more than one FROMRABN/TORABN range is specified for a single file, the ranges are consolidated. For more information, read ***FROMRABN/TORABN Parameter Consolidation***, elsewhere in this section.

INCRNUMREC: Increment for Distribution Report's Number of Elements per Block Distribution

This optional parameter specifies the increment for the number of elements per block section of the Distribution reports produced about ASSO and DATA file space usage. For more information, review the Distribution report sample in ***Sample Report***, elsewhere in this section.

Valid values range from 1 through 10000. If this parameter is omitted, ADADRU will calculate this increment automatically, with the goal of providing a report in the best reviewable form, with a total number of lines less than 100.

The INCRNUMREC parameter is defined within a file group.

INCRRECLEN: Increment for DATA Distribution Report's Record Length Area

This optional parameter specifies the increment for record lengths used in the record length section of the Distribution reports produced about DATA file space usage. Lines with zero values (number of records with lengths of zero) are not printed in the report. For more information, review the DATA Distribution report sample in ***Sample Report***, elsewhere in this section.

This parameter is only valid for Data Storage reports; you will not find a record length section in the Distribution reports about ASSO file space usage.

Valid values range from 6 through 5000. If this parameter is omitted, ADADRU will calculate this increment automatically, with the goal of providing a report in the best reviewable form, with a total number of lines less than 100.

The INCRRECLEN parameter is defined within a file group.

INCRUSAGE: Percent Increment for Distribution Report's Block Filling Level Area

This optional parameter specifies the percentage increment for the block filling level of the Distribution reports produced about ASSO and DATA file space usage. Valid values range from 1 through 50 percent; the default is 5. For more information, review the Distribution report sample in ***Sample Report***, elsewhere in this section.

The INCRUSAGE parameter is defined within a file group.

LOBGROUP: Large Object Group Processing

Specify this optional parameter if you want LOB files processed. Files are handled in their physical file sequences (i.e., the file with the lowest number is processed first regardless of whether it is a LOB or base file).

If this parameter is not specified, LOB files are processed without looking for LOB file associations. The LOBGROUP option is the default when all files in a database are requested for the ADADRU SPACEUSAGE report.

If no LOB files are encountered in the files specified for an ADADRU SPACEUSAGE run, the presence of the LOBGROUP parameter has no effect on the report.

The LOBGROUP parameter is defined within a file group.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADADRU run.

OUTPUT: Create XML Output

This optional parameter should be used when you want to create an XML version of the database report in addition to the normal one. The format of the XML report can be either EBCDIC or converted to UTF-8:

If you specify:	The XML report format will be:
OUTPUT='XML'	EBCDIC
OUTPUT='XML,UTF8'	Converted to UTF-8 (including all tags). When UTF-8 format is requested, you may also use the ACODE parameter to specify the specific code page to which the database base name, file names, and checkpoint names should be converted.

When the OUTPUT parameter is specified, additional DD statements (DDXMLR1 and, optionally, DDXMLR2) must be specified in the ADAREP job.

- DDXMLR1 is required and identifies the data set to which the XML version of the base database report should be written.
- DDXMLR2 is used to store the XML version of any checkpoint information in the report; it is therefore only required if checkpoint information is requested for the report via the CPLIST or CPEXLIST parameters.

On z/OS, the data sets specified by these DD statements must be defined with DCB parameters RECFM=VB, LRECL=512.

The resulting XML documents exactly match the information produced in the regular report, including any filtering done for the report (for example if NOFDT or NOPPT are requested). Once the XML document is created, it can be transferred to a Windows or UNIX machine and you can use your own code or other tools to transform it.

The XML document is structured as defined by an XML schema definition file (*adabas.xsd*) provided in the online Community Discussion Forum for Adabas (***https://tech.forums.softwareag.com/t/files-for-processing-adarep-xml-output-ada822/92450***). A style sheet (*adabas.xsl*) is also provided for your use at the same location. You can transfer these XML files from the Adabas Forum to any system (probably Windows or UNIX) using automated tools or user-written programs in the programming language of your choice (Natural, Java, .Net, C, Perl, or Python).

PASSOSPACE: Process Only ASSO File Areas

The PFILESPACE, PASSOSPACE, and PDATASPACE parameters are mutually exclusive. If none of these parameters are specified, PFILESPACE=SUMMARY is used as a default.

The PASSOSPACE parameter indicates that only the normal index blocks (NI blocks) of the Associator (ASSO) file area should be processed for the ADADRU SPACEUSAGE report. Using the SUMMARY and DETAIL values of the PASSOSPACE parameter, you can indicate whether a detailed report (DETAIL) should be produced; a summary report (SUMMARY) is always produced. If DETAIL is specified, both a detail report and a summary report are produced; if DETAIL is not specified, only a summary report is produced.

The PASSOSPACE parameter is defined within a file group.

In addition, if a single file is being processed, you can use the FROMRABN and TORABN parameters to specify an RABN range that should be processed.

PDATASPACE: Process Only DATA File Areas

The PFILESPACE, PASSOSPACE, and PDATASPACE parameters are mutually exclusive. If none of these parameters are specified, PFILESPACE=SUMMARY is used as a default.

The PDATASPACE parameter indicates that only the Data Storage (DATA) file area should be processed for the ADADRU SPACEUSAGE report. Using the SUMMARY and DETAIL values of the PDATASPACE parameter, you can indicate whether a detailed report (DETAIL) should be produced; a summary report (SUMMARY) is always produced. If DETAIL is specified, both a detail report and a summary report are produced; if DETAIL is not specified, only a summary report is produced.

The PDATASPACE parameter is defined within a file group.

In addition, if a single file is being processed, you can use the FROMRABN and TORABN parameters to specify an RABN range that should be processed.

PFILESPACE: Process Both ASSO and DATA File Areas

The PFILESPACE, PASSOSPACE, and PDATASPACE parameters are mutually exclusive. If none of these parameters are specified, PFILESPACE=SUMMARY is used as a default.

The PFILESPACE parameter indicates that both the normal index blocks (NI blocks) of the Associator (ASSO) file area and the Data Storage (DATA) file area should be processed for the ADADRU SPACEUSAGE report. Using the SUMMARY and DETAIL values of the PFILESPACE parameter, you can indicate whether a detailed report (DETAIL) should be produced; a summary report (SUMMARY) is always produced. If DETAIL is specified, both a detail report and a summary report are produced; if DETAIL is not specified, only a summary report is produced.

The PFILESPACE parameter is defined within a file group. If PFILESPACE=SUMMARY, all files are processed.

SAMPLING: Percentage of Blocks Read for Summary Reports

This optional parameter specifies the percentage of RABN blocks that should be read to generate summary reports. Valid values range from 10 through 100 percent. The default is 100, which indicates that all file RABNs should be processed to generated summary reports.

The SAMPLING parameter is defined within a file group.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested. See **Syntax Checking with the TEST Parameter** for more information about using the TEST parameter in ADADRU functions.

The NOUSERABEND, TEST, and ABEND34 parameters affect the processing of the entire ADADRU run.

TORABN: Ending Data Storage Block Number

The RABN of the Data Storage or Associator block where report processing should end. It is often paired with the FROMRABN parameter to specify a range of RABN blocks.

If the FROMRABN parameter is omitted and the TORABN parameter is specified, processing is performed from the beginning of the file to the block specified by the TORABN parameter. If the FROMRABN parameter is specified and the TORABN parameter is omitted, processing is performed from the FROMRABN block to the end of the file. The TORABN parameter can only be specified as a subparameter of the PASSOSPACE or PDATASPACE parameters.

The TORABN parameter is applicable only if a file group consists of a single file. Only one FROMRABN/TORABN range can be specified for single file group.

The maximum number of FROMRABN/TORABN ranges that can be specified for the same file in a single ADADRU SPACEUSAGE run is eight. If more than one FROMRABN/TORABN range is specified for a single file, the ranges are consolidated. For more information, read ***FROMRABN/TORABN Parameter Consolidation***, elsewhere in this section.

Examples

In the following example, detail and summary space usage reports will be produced for both the ASSO and DATA areas for files 5 and 9. In the Distribution report, a block filling level increment of 10 percent will be used.

```
ADADRU SPACEUSAGE FILE=5,9,PFILESPACE=DETAIL,INCRUSAGE=10
```

In the following example, detail and summary space usage reports will be produced for only the DATA area of file 7, starting at RABN 10048. In the Distribution report, a block filling level increment of 8 percent will be used.

```
ADADRU SPACEUSAGE FILE=7,INCRUSAGE=8,FROMRABN=10048,PDATASPACE=DETAIL
```

In the following example, detail and summary space usage reports will be produced for only the DATA area of file 7. In addition, XML output is generated for detail and summary reports and converted to UTF-8 format:

```
ADADRU SPACEUSAGE FILE=7,PDATASPACE=DETAIL,OUTPUT='XML,UTF8'
```

To see a sample of these reports, read ***Sample Report***, elsewhere in this section.

Sample Reports

If you request detail reports (if parameter PFILESPACE, PASSOSPACE, or PDATASPACE is set to "DETAIL"), the "File ASSO/DATA Detail Report" is produced, as appropriate. The report can be directed to a specific file using the DD statement with the name DDDETAIL. (For more information, read ***JCL Requirements and Examples*** , elsewhere in this section. Detail reports may be very large because information may be printed for every RABN in a file (or a specific RABN range).

In the *Data Storage Detail* section of the report the column header "Spanned" is always printed, regardless of whether or not the file supports spanned records. The "Spanned" column is split into

two columns called "Num" and "Flg". The "Num" column provides the number of spanned record segments in each Data Storage RABN. The "Flg" column values can be "P", "S", "L", or a blank, identifying the segment of the spanned record stored at the RABN:

- "P" indicates that the current RABN contains the primary segment of the spanned record.
- "S" indicates that the current RABN contains a secondary segment of the spanned record.
- "L" indicates that the current RABN contains the last segment of the spanned record.
- A blank indicates that the current RABN does not contain any segment of a spanned record.

This section provides some sample ADADRU sample reports. The ADADRU SPACEUSAGE parameters used in each sample report are shown at the beginning of the report.

- Sample 1: Parameter Consolidation for Files
- Sample 2: Large Object (LOB) File Processing
- Sample 3: SAMPLING Parameter Use
- Sample 4: FILE Parameter Processing
- Sample 5: EXPGROUP Parameter Processing
- Sample 6: XML Output for Detail and Summary Report

Sample 1: Parameter Consolidation for Files

This sample report shows parameters INCRUSAGE, INCRNUMREC, INCRECLEN and FROMRABN/TORABN for files 1 and 5 being consolidated. ADADRU builds a list of the RABNs that must be processed based on the FROMRABN/TORABN parameters. If some RABNs belong to more than one FROMRABN/TORABN range, the ranges are combined into one range. ADADRU examines all of the settings of the INCRUSAGE, INCRNUMREC, and INCRRECLEN parameters for the file group and chooses the most detailed value for every parameter. ADADRU processes the files and produces the reports only after all parameters have been consolidated as appropriate.

```
A D A D R U   V8.2  SM8   DBID = 00821  Started          2012-05-11  12:37:52    ↩
                                                                               ↩
 Parameters:                                                                   ↩
 -----------                                                                   ↩
                                                                               ↩
 ADADRU                   NOUSERABEND                                          ↩
                                                                               ↩
 ADADRU SPACEUSAGE FILE=5,PASSOSPACE=DETAIL,FROMRABN=2072,TORABN=2075          ↩
 ADADRU                   INCRUSAGE=7,INCRNUMREC=11,INCRRECLEN=33              ↩
```

```
                                                                                ↩
ADADRU            FILE=5,PASSOSPACE=DETAIL,FROMRABN=2086,TORABN=2090            ↩
                                                                                ↩
ADADRU            FILE=1,PDATASPACE=DETAIL,FROMRABN=21,TORABN=23                ↩
ADADRU                   INCRUSAGE=6,INCRNUMREC=12,INCRRECLEN=29                ↩
                                                                                ↩
ADADRU            FILE=5,PDATASPACE=DETAIL,FROMRABN=351,TORABN=353              ↩
ADADRU                   INCRUSAGE=9,INCRNUMREC=15,INCRRECLEN=40                ↩
                                                                                ↩
ADADRU            FILE=5,PDATASPACE=DETAIL,FROMRABN=355,TORABN=360              ↩
                                                                                ↩
ADADRU            FILE=1,PASSOSPACE=DETAIL,FROMRABN=990,TORABN=995              ↩
                                                                                ↩
ADADRU            FILE=1,PASSOSPACE=DETAIL,FROMRABN=993,TORABN=999              ↩
                                                                                ↩
ADADRU            FILE=5,PDATASPACE=DETAIL,FROMRABN=359,TORABN=365              ↩
ADADRU                   INCRUSAGE=3,INCRNUMREC=10,INCRRECLEN=21                ↩
                                                                                ↩
ADADRU            FILE=5,PDATASPACE=DETAIL,FROMRABN=620,TORABN=622              ↩
                                                                                ↩
ADADRU            FILE=5,PDATASPACE=DETAIL,FROMRABN=624,TORABN=625              ↩
                                                                                ↩
ADADRU            FILE=1,PDATASPACE=DETAIL,FROMRABN=23,TORABN=27                ↩
                                                                                ↩
ADADRU            FILE=1,PDATASPACE=DETAIL,FROMRABN=30,TORABN=33                ↩
ADADRU                   INCRUSAGE=3,INCRNUMREC=22,INCRRECLEN=44                ↩
```

```
                                                                                ↵
                                                                                ↵
1                                                                               ↵
                                                                                ↵
 DRU203I, ******* SPACEUSAGE-1 started *******     2012-05-11 12:37:52          ↵
                                                                                ↵
                                                                                ↵
                                                                                ↵
 File group number 1.  FILE=5                                                   ↵
 -----------------                                                              ↵
 DRU169I, ASSO RABNs 2073 (x'00000819') - 2075 (x'0000081B')                    ↵
          from specified RABN range are outside                                 ↵
          of extents for file 5, or these RABNs are not ASSO NI RABNs.          ↵
          Only ASSO NI RABN(s) which belongs to the file will be processed.     ↵
          Run ADAREP utility to obtain information about file extents.          ↵
                                                                                ↵
                                                                                ↵
 File group number 2.  FILE=5                                                   ↵
 -----------------                                                              ↵
                                                                                ↵
 File group number 3.  FILE=1                                                   ↵
 -----------------                                                              ↵
                                                                                ↵
 File group number 4.  FILE=5                                                   ↵
 -----------------                                                              ↵
                                                                                ↵
```

```
File group number 5.  FILE=5                                                    ↩
-----------------                                                               ↩
                                                                                ↩
File group number 6.  FILE=1                                                    ↩
-----------------                                                               ↩
                                                                                ↩
File group number 7.  FILE=1                                                    ↩
-----------------                                                               ↩
                                                                                ↩
File group number 8.  FILE=5                                                    ↩
-----------------                                                               ↩
                                                                                ↩
File group number 9.  FILE=5                                                    ↩
-----------------                                                               ↩
                                                                                ↩
File group number 10.  FILE=5                                                   ↩
-----------------                                                               ↩
                                                                                ↩
File group number 11.  FILE=1                                                   ↩
-----------------                                                               ↩
                                                                                ↩
File group number 12.  FILE=1                                                   ↩
-----------------                                                               ↩
                                                                                ↩
DRU200I, The input parameters accepted.                                         ↩
                                                                                ↩
```

```
                                                                                ↩
1                                                                               ↩
                                                                                ↩
1                                                                               ↩
 ********************************************************************************↩
 ********************************************************************************↩
 ********************************************************************************↩
 ****                                                                    ****    ↩
 ****             F I L E      1 ( EMPLOYEES        )                   ****    ↩
 ****                                                                    ****    ↩
 ****                       2012-05-11 12:37:52                          ****    ↩
 ********************************************************************************↩
 ********************************************************************************↩
 ********************************************************************************↩
                                                                                ↩
                                                                                ↩
 Consolidated input parameters:                                                 ↩
 ------------------------------                                                 ↩
  File number = 1                                                               ↩
            ERRLIMIT      = 1                                                   ↩
            LOB-GROUP     = No                                                  ↩
            EXP-GROUP     = No                                                  ↩
      ASSO  Report type   = SUMMARY + DETAIL                                    ↩
            SAMPLING      = 100    (10 of every 10 records read)                ↩
            INCRUSAGE     = 5                                                   ↩
            INCRNUMREC    = Auto                                                ↩
```

```
            FROM/TO RABN  = 990          - 999                                ↩
      DATA  Report type   = SUMMARY + DETAIL                                  ↩
            SAMPLING      = 100    (10 of every 10 records read)              ↩
            INCRUSAGE     = 3                                                 ↩
            INCRNUMREC    = 12                                                ↩
            INCRRECLEN    = 29                                                ↩
            FROM/TO RABN  = 21           - 27                                 ↩
            FROM/TO RABN  = 30           - 33                                 ↩
                                                                              ↩
1                                                                             ↩
                                                                              ↩
 File options (from FCB):                                                     ↩
 ------------------------                                                     ↩
 Date / Time Loaded ..................... 2009-10-05 13:36:16.33              ↩
 Date / Time of last update ............. 2011-02-12 10:09:37.80              ↩
 Date / Time of last physical restructure 2009-10-05 13:36:16.33              ↩
                                                                              ↩
 ISN Reusage ............................ Yes                                  ↩
 DATA Storage Reusage ................... Yes                                  ↩
 MIXDSDEV ............................... Yes                                  ↩
 Spanned Records Support ................ Yes                                  ↩
 Index compressed ....................... No                                   ↩
 2-Byte MU/PE Index ..................... No                                   ↩
 Highest Index Level .................... 3                                    ↩
                                                                              ↩
 LOB fields ............................. No                                   ↩
```

```
 LOB file ............................. No                                    ↩
 Expanded ............................. No                                    ↩
                                                                             ↩
1                                                                            ↩
 I/////////////////////////////////////////////////////////////I              ↩
 II                                                           II              ↩
 II     ASSOCIATOR SUMMARY REPORT OF NORMAL INDEX FOR FILE 1      II          ↩
 II                                                           II              ↩
 I/////////////////////////////////////////////////////////////I              ↩
                                                                             ↩
 Physical extents list                                                       ↩
 ------------------------------------------------------------------          ↩
 I Extent I Type I          From           To  I % unused I Device I          ↩
 I        I      I          RABN          RABN I  blocks  I  type  I          ↩
 I--------I------I-----------------------------I----------I--------I          ↩
 I     1  I  NI  I           986          1255 I     77   I  3390  I          ↩
 I        I      I                             I          I        I          ↩
 ------------------------------------------------------------------          ↩
                                                                             ↩
 Range of RABNs being processed according                                    ↩
 to consolidated input parameters                                            ↩
 (unused RABNs at the end of extents excluded)                               ↩
 ----------------------------------------------                              ↩
 I Extent I Type I          From           To  I                             ↩
 I        I      I          RABN          RABN I                             ↩
 I--------I------I-----------------------------I                             ↩
```

```
I     1  I  NI  I           990           999 I                                ↩
I        I      I                            I                                ↩
-----------------------------------------------                                ↩
                                                                               ↩
File ......................... 1                                              ↩
Device Type .................. 3390                                           ↩
Block Size ................... 2544                                           ↩
ASSO Padding factor .......... 10%   (254 bytes)                               ↩
                                                                               ↩
                                                                               ↩
Number of allocated blocks ..............................  270                 ↩
Number of used blocks ...................................  62                  ↩
Number of blocks read (SAMPLING = 100%) .................  10                  ↩
Number of unused blocks at the end of extents ............  208                 ↩
Number of unused blocks within the used extents ..........  0                   ↩
Percent of blocks with free space less than padding ......  0                   ↩
Average percent of block filling (unused blocks excluded)   79                  ↩
Number of elements in blocks read .......................  1691                ↩
Average number of elements per block ....................  169                 ↩
Average element length ..................................  11                  ↩
Minimum element length ..................................  8                   ↩
Maximum element length ..................................  64                  ↩
Number of elements with minimum length ..................  9                   ↩
Number of elements with maximum length ..................  1                   ↩
                                                                               ↩
1                                                                              ↩
```

```
                                                                                ↩
I================================================I                              ↩
II                                              II                              ↩
II    DISTRIBUTION REPORTS ASSO NORMAL INDEX    II                              ↩
II                    File = 1                  II                              ↩
II             Device type = 3390               II                              ↩
I================================================I                              ↩
                                                                                ↩
Block filling level; Increment = 5%                                             ↩
------------------------------------------------                                ↩
I      Range of       I    Number of blocks      I                              ↩
I   percent filled    I  filled in this range    I                              ↩
I--------------------I-------------------------I                               ↩
I      26 -     30    I               1         I                              ↩
I      51 -     55    I               1         I                              ↩
I      86 -     90    I               8         I                              ↩
------------------------------------------------                                ↩
                                                                                ↩
------------------------------------------------                                ↩
I   Percent ranges    I  Number of blocks with   I                              ↩
I   of free space     I free space in this range I                              ↩
I--------------------I-------------------------I                               ↩
I       6 -     10    I               7         I                              ↩
I      11 -     15    I               1         I                              ↩
I      46 -     50    I               1         I                              ↩
I      71 -     75    I               1         I                              ↩
```

```
 ------------------------------------------------                                ↩
                                                                                 ↩
1                                                                                ↩
 Number of elements per block; Increment = Auto                                  ↩
 ------------------------------------------------                                ↩
 I  Number of elements I          Number          I                              ↩
 I     per block       I        of blocks         I                              ↩
 I---------------------I--------------------------I                              ↩
 I      52 -     54    I               1          I                              ↩
 I     100 -    102    I               1          I                              ↩
 I     166 -    168    I               1          I                              ↩
 I     175 -    177    I               3          I                              ↩
 I     178 -    180    I               2          I                              ↩
 I     241 -    243    I               1          I                              ↩
 I     244 -    246    I               1          I                              ↩
 ------------------------------------------------                                ↩
                                                                                 ↩
1                                                                                ↩
 I////////////////////////////////////////////////////////////////I              ↩
 II                                                              II              ↩
 II          DATA STORAGE SUMMARY REPORT FOR FILE 1              II              ↩
 II                                                              II              ↩
 I////////////////////////////////////////////////////////////////I              ↩
                                                                                 ↩
 Physical extents list                                                           ↩
 ----------------------------------------------------------------               ↩
```

```
I Extent I Type I          From           To  I % unused I Device I
I        I      I          RABN          RABN I  blocks  I  type  I
I--------I------I-----------------------------I----------I--------I
I     1  I  DS  I             6           155 I     69   I  3390  I
I        I      I                             I          I        I
-------------------------------------------------------------------


Range of RABNs being processed according
to consolidated input parameters
(unused RABNs at the end of extents excluded)
-----------------------------------------------
I Extent I Type I          From           To  I
I        I      I          RABN          RABN I
I--------I------I-----------------------------I
I     1  I  DS  I            21            27 I
I     2  I  DS  I            30            33 I
I        I      I                             I
-----------------------------------------------


File ......................... 1
Device Type .................. 3390
Block Size.................... 5064
DATA Storage Padding factor .. 10%   (506 bytes)


Number of allocated blocks ..............................  150
```

```
Number of used blocks ..................................  47                ↩
Number of blocks read (SAMPLING = 100%) ................  11                ↩
Number of unused blocks at the end of extents ..........  103               ↩
Number of unused blocks within the used extents ........  4                 ↩
Percent of blocks with free space less than padding .....  9                 ↩
Average percent of block filling (unused blocks excluded)  56                ↩
Number of records in blocks read .......................  175               ↩
   Number of Normal records ............................  175               ↩
   Number of Spanned records ...........................  0                 ↩
Number of blocks with Spanned records ..................  0                 ↩
Average number of records per block ....................  16                ↩
                                                                             ↩
Normal records:                                                              ↩
   Average record length ...............................  179               ↩
   Minimum record length ...............................  126               ↩
   Maximum record length ...............................  244               ↩
   Number of records with minimum length ...............  1                 ↩
   Number of records with maximum length ...............  1                 ↩
                                                                             ↩
Spanned Record:                                                              ↩
   Average record length ...............................  0                 ↩
   Minimum record length ...............................  0                 ↩
   Maximum record length ...............................  0                 ↩
   Number of records with minimum length ...............  0                 ↩
   Number of records with maximum length ...............  0                 ↩
   Average numbers of segments .........................  0                 ↩
```

```
   Minimum numbers of segments ..........................  0                  ↩
   Maximum numbers of segments ..........................  0                  ↩
   Number of Primary Spanned Records ....................  0                  ↩
   Number of Secondary Spanned Records...................  0                  ↩
   Number of AC2 blocks read ............................  0                  ↩
   Data Storage blocks with secondary segments read ......  0                  ↩
                                                                               ↩
1                                                                              ↩
                                                                               ↩
 I================================================I                            ↩
 II                                              II                            ↩
 II       DISTRIBUTION REPORTS DATA STORAGE      II                            ↩
 II                   File = 1                   II                            ↩
 II             Device type = 3390               II                            ↩
 I================================================I                            ↩
                                                                               ↩
 Block filling level; Increment = 3%                                           ↩
 ------------------------------------------------                              ↩
 I      Range of       I    Number of blocks      I                            ↩
 I   percent filled    I  filled in this range    I                            ↩
 I--------------------I--------------------------I                            ↩
 I      85 -     87    I               2         I                            ↩
 I      88 -     90    I               4         I                            ↩
 I      91 -     93    I               1         I                            ↩
 ------------------------------------------------                              ↩
                                                                               ↩
```

```
 ------------------------------------------------                                ↩
 I   Percent ranges    I  Number of blocks with   I                             ↩
 I   of free space     I free space in this range I                             ↩
 I---------------------I--------------------------I                             ↩
 I       7 -      9    I                1         I                             ↩
 I      10 -     12    I                4         I                             ↩
 I      13 -     15    I                2         I                             ↩
 ------------------------------------------------                                ↩
                                                                                 ↩
1                                                                                ↩
 Number of records per block; Increment = 12 records                             ↩
 ------------------------------------------------                                ↩
 I  Number of records  I          Number          I                             ↩
 I     per block       I        of blocks         I                             ↩
 I---------------------I--------------------------I                             ↩
 I      13 -     24    I                2         I                             ↩
 I      25 -     36    I                5         I                             ↩
 ------------------------------------------------                                ↩
                                                                                 ↩
1                                                                                ↩
 Normal records length; Increment = 29 bytes                                     ↩
 ------------------------------------------------                                ↩
 I        Length       I          Number          I                             ↩
 I      of record      I        of records        I                             ↩
 I---------------------I--------------------------I                             ↩
 I     117 -    145    I                7         I                             ↩
```

```
 I     146 -    174    I               83         I                              ↩
 I     175 -    203    I               47         I                              ↩
 I     204 -    232    I               33         I                              ↩
 I     233 -    261    I                5         I                              ↩
 ------------------------------------------------                                 ↩
                                                                                  ↩
 DRU163W, WARNING: No data for "Spanned records length" report                    ↩
                                                                                  ↩
                                                                                  ↩
1                                                                                 ↩
                                                                                  ↩
1                                                                                 ↩
 ************************************************************************         ↩
 ************************************************************************         ↩
 ************************************************************************         ↩
 ****                                                              ****         ↩
 ****             F I L E      5 ( FILE5            )              ****         ↩
 ****                                                              ****         ↩
 ****                      2012-05-11 12:37:52                     ****         ↩
 ************************************************************************         ↩
 ************************************************************************         ↩
 ************************************************************************         ↩
                                                                                  ↩
                                                                                  ↩
 Consolidated input parameters:                                                   ↩
 ------------------------------                                                   ↩
```

```
  File number = 5                                                              ↩
            ERRLIMIT      = 1                                                  ↩
            LOB-GROUP     = No                                                 ↩
            EXP-GROUP     = No                                                 ↩
      ASSO  Report type   = SUMMARY + DETAIL                                   ↩
            SAMPLING      = 100    (10 of every 10 records read)               ↩
            INCRUSAGE     = 7                                                  ↩
            INCRNUMREC    = 11                                                 ↩
            FROM/TO RABN  = 2072         - 2072                                ↩
            FROM/TO RABN  = 2086         - 2087                                ↩
      DATA  Report type   = SUMMARY + DETAIL                                   ↩
            SAMPLING      = 100    (10 of every 10 records read)               ↩
            INCRUSAGE     = 3                                                  ↩
            INCRNUMREC    = 10                                                 ↩
            INCRRECLEN    = 21                                                 ↩
            FROM/TO RABN  = 351          - 353                                 ↩
            FROM/TO RABN  = 355          - 365                                 ↩
            FROM/TO RABN  = 620          - 622                                 ↩
            FROM/TO RABN  = 624          - 625                                 ↩
                                                                               ↩
1                                                                              ↩
                                                                               ↩
 File options (from FCB):                                                      ↩
 ------------------------                                                      ↩
 Date / Time Loaded ..................... 2010-04-28 10:01:49.79               ↩
 Date / Time of last update ............. 2010-05-04 07:07:08.47               ↩
```

```
 Date / Time of last physical restructure 2010-04-28 10:01:49.79                ↩
                                                                                ↩
 ISN Reusage ........................... No                                     ↩
 DATA Storage Reusage .................. Yes                                    ↩
 MIXDSDEV .............................. No                                     ↩
 Spanned Records Support ............... Yes                                    ↩
 Index compressed ...................... No                                     ↩
 2-Byte MU/PE Index .................... No                                     ↩
 Highest Index Level ................... 3                                      ↩
                                                                                ↩
 LOB fields ............................ No                                     ↩
 LOB file .............................. No                                     ↩
 Expanded .............................. No                                     ↩
                                                                                ↩
1                                                                               ↩
 I/////////////////////////////////////////////////////////////////I            ↩
 II                                                               II            ↩
 II     ASSOCIATOR SUMMARY REPORT OF NORMAL INDEX FOR FILE 5      II            ↩
 II                                                               II            ↩
 I/////////////////////////////////////////////////////////////////I            ↩
                                                                                ↩
 Physical extents list                                                          ↩
 ---------------------------------------------------------------------          ↩
 I Extent I Type I          From           To  I % unused I Device I            ↩
 I        I      I          RABN          RABN I  blocks  I  type  I            ↩
 I--------I------I-----------------------------I----------I--------I            ↩
```

```
I     1  I  NI  I          2072          2072 I      0   I  3390  I
I     2  I  NI  I          2086          2096 I     81   I  3390  I
I        I      I                            I          I        I
-------------------------------------------------------------------


Range of RABNs being processed according
to consolidated input parameters
(unused RABNs at the end of extents excluded)
-----------------------------------------------
I Extent I Type I          From           To  I
I        I      I          RABN          RABN I
I--------I------I-----------------------------I
I     1  I  NI  I          2072          2072 I
I     2  I  NI  I          2086          2087 I
I        I      I                            I
-----------------------------------------------


File ......................... 5
Device Type .................. 3390
Block Size ................... 2544
ASSO Padding factor .......... 10%   (254 bytes)



Number of allocated blocks ..............................  12
Number of used blocks ...................................  3
Number of blocks read (SAMPLING = 100%) .................  3
```

```
 Number of unused blocks at the end of extents .............  9                    ↩
 Number of unused blocks within the used extents ...........  0                    ↩
 Percent of blocks with free space less than padding ......  0                     ↩
 Average percent of block filling (unused blocks excluded)   40                    ↩
 Number of elements in blocks read ........................  4                     ↩
 Average number of elements per block .....................  1                     ↩
 Average element length ...................................  772                   ↩
 Minimum element length ...................................  406                   ↩
 Maximum element length ...................................  1056                  ↩
 Number of elements with minimum length ...................  1                     ↩
 Number of elements with maximum length ...................  1                     ↩
                                                                                   ↩
1                                                                                  ↩
                                                                                   ↩
 I================================================I                                ↩
 II                                              II                                ↩
 II    DISTRIBUTION REPORTS ASSO NORMAL INDEX    II                                ↩
 II                    File = 5                  II                                ↩
 II             Device type = 3390               II                                ↩
 I================================================I                                ↩
                                                                                   ↩
 Block filling level; Increment = 7%                                               ↩
 ------------------------------------------------                                  ↩
 I      Range of       I    Number of blocks      I                                ↩
 I   percent filled    I  filled in this range    I                                ↩
 I--------------------I-------------------------I                                 ↩
```

```
I      29 -     35    I                1         I                                ↩
I      36 -     42    I                1         I                                ↩
I      43 -     49    I                1         I                                ↩
-------------------------------------------------                                ↩
                                                                                 ↩
-------------------------------------------------                                ↩
I   Percent ranges    I  Number of blocks with   I                               ↩
I   of free space     I free space in this range I                               ↩
I---------------------I--------------------------I                               ↩
I      50 -     56    I                1         I                                ↩
I      57 -     63    I                1         I                                ↩
I      64 -     70    I                1         I                                ↩
-------------------------------------------------                                ↩
                                                                                 ↩
1                                                                                ↩
 Number of elements per block; Increment = 11 elements                           ↩
 ------------------------------------------------                                ↩
 I  Number of elements I          Number          I                              ↩
 I     per block       I        of blocks         I                              ↩
 I---------------------I--------------------------I                              ↩
 I       1 -     11    I                3         I                              ↩
 ------------------------------------------------                                ↩
                                                                                 ↩
1                                                                                ↩
 I////////////////////////////////////////////////////////////////I               ↩
 II                                                              II              ↩
```

```
II          DATA STORAGE SUMMARY REPORT FOR FILE 5               II
II                                                               II
I/////////////////////////////////////////////////////////////////I

Physical extents list
-----------------------------------------------------------------
I Extent I Type I          From           To  I % unused I Device I
I        I      I          RABN          RABN I  blocks  I  type  I
I--------I------I-----------------------------I----------I--------I
I     1  I  DS  I           350           466 I      0   I  3390  I
I     2  I  DS  I           617          2381 I     18   I  3390  I
I        I      I                             I          I        I
-----------------------------------------------------------------

Range of RABNs being processed according
to consolidated input parameters
(unused RABNs at the end of extents excluded)
-----------------------------------------------
I Extent I Type I          From           To  I
I        I      I          RABN          RABN I
I--------I------I-----------------------------I
I     1  I  DS  I           351           353 I
I     2  I  DS  I           355           365 I
I     3  I  DS  I           620           622 I
I     4  I  DS  I           624           625 I
I        I      I                             I
```

```
-------------------------------------------- ↩
                                             ↩
File ......................... 5             ↩
Device Type .................. 3390          ↩
Block Size.................... 5064          ↩
DATA Storage Padding factor .. 0%    (0 bytes) ↩
                                             ↩
                                             ↩
Number of allocated blocks ..............  1882 ↩
Number of used blocks ...................  1571 ↩
Number of blocks read (SAMPLING = 100%) .  19   ↩
Number of unused blocks at the end of extents ..... 311 ↩
Number of unused blocks within the used extents ..... 1 ↩
Percent of blocks with free space less than padding ..... 0 ↩
Average percent of block filling (unused blocks excluded) 94 ↩
Number of records in blocks read ........  76   ↩
   Number of Normal records .............  71   ↩
   Number of Spanned records ............  5    ↩
Number of blocks with Spanned records ...  16   ↩
Average number of records per block .....  5    ↩
                                             ↩
Normal records:                              ↩
   Average record length ................  147  ↩
   Minimum record length ................  21   ↩
   Maximum record length ................  271  ↩
   Number of records with minimum length ..... 35 ↩
```

```
   Number of records with maximum length .................  36                  ↩
                                                                                 ↩
Spanned Record:                                                                  ↩
   Average record length ................................  15072                ↩
   Minimum record length ................................  15068                ↩
   Maximum record length ................................  15076                ↩
   Number of records with minimum length .................  1                    ↩
   Number of records with maximum length .................  1                    ↩
   Average numbers of segments ...........................  3                    ↩
   Minimum numbers of segments ...........................  3                    ↩
   Maximum numbers of segments ...........................  3                    ↩
   Number of Primary Spanned Records .....................  5                    ↩
   Number of Secondary Spanned Records....................  10                   ↩
   Number of AC2 blocks read .............................  1                    ↩
   Data Storage blocks with secondary segments read ......  10                   ↩
                                                                                 ↩
1                                                                                ↩
                                                                                 ↩
I================================================I                               ↩
II                                              II                               ↩
II       DISTRIBUTION REPORTS DATA STORAGE      II                               ↩
II                   File = 5                   II                               ↩
II             Device type = 3390               II                               ↩
I================================================I                               ↩
                                                                                 ↩
Block filling level; Increment = 3%                                              ↩
```

```
 ------------------------------------------------                                ↩
 I      Range of       I    Number of blocks      I                             ↩
 I   percent filled    I  filled in this range    I                             ↩
 I---------------------I--------------------------I                             ↩
 I      97 -     99    I                9         I                             ↩
 I     100 -    100    I                9         I                             ↩
 ------------------------------------------------                               ↩
                                                                                ↩
 ------------------------------------------------                               ↩
 I   Percent ranges    I  Number of blocks with   I                             ↩
 I   of free space     I free space in this range I                             ↩
 I---------------------I--------------------------I                             ↩
 I       1 -      3    I               18         I                             ↩
 ------------------------------------------------                               ↩
                                                                                ↩
1                                                                               ↩
 Number of records per block; Increment = 10 records                            ↩
 ------------------------------------------------                               ↩
 I  Number of records  I          Number           I                            ↩
 I     per block       I        of blocks          I                            ↩
 I---------------------I--------------------------I                             ↩
 I       1 -     10    I               16         I                             ↩
 I      21 -     30    I                2         I                             ↩
 ------------------------------------------------                               ↩
                                                                                ↩
1                                                                               ↩
```

```
Normal records length; Increment = 21 bytes
-------------------------------------------------
I        Length       I          Number          I
I      of record      I        of records        I
I--------------------I-------------------------I
I       1 -     21    I               35        I
I     253 -    273    I               36        I
-------------------------------------------------


DRU163W, WARNING: No data for "Spanned records length" report



1
************************************************************************
************************************************************************
************************************************************************
****                                                            ****
****       F i l e s   s t a t i s t i c a l   r e p o r t      ****
****                                                            ****
************************************************************************
************************************************************************
************************************************************************

------------------------------------------------------------------------------
I F I  Duration I     ASSOCIATOR (NI)    I      DATA STORAGE      I ParmI Run I
I i IHH:MM:SS.thI-----------------------I-----------------------I-----I-----I
```

```
 I l I           I  Number  I E I Extents I  Number  I E I Extents I E W I E W I  ↩
 I e I           I of RABNs I r I---------I of RABNs I r I---------I r n I r n I  ↩
 I   I           I   read   I r I FCB Run I   read   I r I FCB Run I r g I r g I  ↩
 I---I-----------I----------I---I----I----I----------I---I----I----I--I--I--I--I  ↩
    1 00:00:00.06         10   0    1    1         11   0    2    2  0  0  0  1   ↩
    5 00:00:00.10          3   0    2    2         19   0    4    4  0  0  0  1   ↩
                                                                                  ↩
s1                                                                                ↩
                                                                                  ↩
 DRU205I, ******* SPACEUSAGE-1 ended *******     CC = 04                          ↩

DDDETAIL SPACEUSAGE-1
1File options (from FCB):                                                         ↩
 ------------------------                                                         ↩
 Date / Time Loaded ..................... 2009-10-05 13:36:16.33                 ↩
 Date / Time of last update ............. 2011-02-12 10:09:37.80                 ↩
 Date / Time of last physical restructure 2009-10-05 13:36:16.33                  ↩
                                                                                  ↩
 ISN Reusage ............................ Yes                                      ↩
 DATA Storage Reusage ................... Yes                                      ↩
 MIXDSDEV ............................... Yes                                      ↩
 Spanned Records Support ................ Yes                                      ↩
 Index compressed ....................... No                                       ↩
 2-Byte MU/PE Index ..................... No                                       ↩
 Highest Index Level .................... 3                                        ↩
                                                                                  ↩
```

```
 LOB fields ............................ No                                  ↩
 LOB file .............................. No                                  ↩
 Expanded .............................. No                                  ↩
                                                                            ↩
                                                                            ↩
 I//////////////////////////////////////////////////////////////I           ↩
 II                                                            II            ↩
 II    ASSOCIATOR DETAIL REPORT OF NORMAL INDEX FOR FILE 1      II           ↩
 II                                                            II            ↩
 I//////////////////////////////////////////////////////////////I           ↩
                                                                            ↩
 Device Type ...........  3390                                             ↩
 Block Size ............. 2544                                             ↩
 ASSO Padding Factor .... 10% (254 bytes)                                   ↩
                                                                            ↩
1File 1                                                 Page 1              ↩
 ------------------------------------------------------------------------   ↩
 I   RABN   I   %     I Block  I Free Space I Num  I Element lengths I       ↩
 I          I  Used   I length I------------I elem I-----------------I       ↩
 I          I         I        I Gap I Pad  I      I Min I Max I Avg I       ↩
 I----------I---------I--------I-----I------I------I-----I-----I-----I       ↩
         990     90      2278      12    254    175    13    13    13       ↩
         991     90      2278      12    254    175    13    13    13       ↩
         992     28       705    1585    254     54    13    13    13       ↩
         993     90      2277      13    254    180     8    34    12       ↩
         994     90      2283       7    254    175     8    38    13       ↩
```

```
        995     89      2276      14    254    178     8    34    12
        996     90      2277      13    254    168     8    64    13
        997     51      1287    1003    254    100     8    22    12
        998     90      2286       4    254    245     9    15     9
        999     90      2283       7    254    241     9    15     9


I//////////////////////////////////////////////////////////////////I
II                                                                II
II            DATA STORAGE DETAIL REPORT FOR FILE 1               II
II                                                                II
I//////////////////////////////////////////////////////////////////I

Device Type ...........  3390
Block Size ............. 5064
DATA Padding Factor .... 10% (506 bytes)

1File 1                                                    Page 1
--------------------------------------------------------------------------------
I   RABN   I   %    I Block  I Free space I Num I  Spanned  I Record lengths  I
I          I  Used  I Length I------------I Rec I-----------I-----------------I
I          I        I        I Gap I Pad  I     I Num I Flg I Min I Max I Avg I
I----------I--------I--------I-----I------I-----I-----I-----I-----I-----I-----I
          21    88       4476     82   506     27    0          126   208   165
          22    91       4613      0   451     28    0          136   196   164
```

```
          23    88       4475     83   506     26    0          143   204   171   ↵
          24    87       4382    176   506     25    0          153   205   175   ↵
          25    89       4528     30   506     27    0          143   193   167   ↵
          26    89       4522     36   506     22    0          170   244   205   ↵
          27    86       4357    201   506     20    0          205   239   217   ↵
                                                                                  ↵
                                                                                  ↵
1File options (from FCB):                                                         ↵
 ------------------------                                                         ↵
 Date / Time Loaded ..................... 2010-04-28 10:01:49.79                  ↵
 Date / Time of last update ............. 2010-05-04 07:07:08.47                  ↵
 Date / Time of last physical restructure 2010-04-28 10:01:49.79                  ↵
                                                                                  ↵
 ISN Reusage ............................ No                                       ↵
 DATA Storage Reusage ................... Yes                                      ↵
 MIXDSDEV ............................... No                                       ↵
 Spanned Records Support ................ Yes                                      ↵
 Index compressed ....................... No                                       ↵
 2-Byte MU/PE Index ..................... No                                       ↵
 Highest Index Level .................... 3                                        ↵
                                                                                  ↵
 LOB fields ............................. No                                       ↵
 LOB file ............................... No                                       ↵
 Expanded ............................... No                                       ↵
                                                                                  ↵
                                                                                  ↵
```

```
I/////////////////////////////////////////////////////////////////I
II                                                               II
II    ASSOCIATOR DETAIL REPORT OF NORMAL INDEX FOR FILE 5        II
II                                                               II
I/////////////////////////////////////////////////////////////////I

Device Type ...........  3390
Block Size ............. 2544
ASSO Padding Factor .... 10% (254 bytes)

1File 5                                                Page 1
-------------------------------------------------------------------------
I   RABN   I   %     I Block  I Free Space I Num  I Element lengths I
I          I  Used   I length I------------I elem I-----------------I
I          I         I        I Gap I Pad  I      I Min I Max I Avg I
I----------I---------I--------I-----I------I------I-----I-----I-----I
       2072     48      1213    1077    254      2   406   804   605
       2086     42      1059    1231    254      1  1056  1056  1056
       2087     32       825    1465    254      1   822   822   822


I/////////////////////////////////////////////////////////////////I
II                                                               II
II            DATA STORAGE DETAIL REPORT FOR FILE 5              II
II                                                               II
```

```
 I//////////////////////////////////////////////////////////////I
 Device Type ...........  3390
 Block Size ..............  5064
 DATA Padding Factor ....  0% (0 bytes)
1File 5                                                    Page 1
 ------------------------------------------------------------------------------
 I   RABN   I   %    I Block  I Free space I Num I  Spanned  I Record lengths  I
 I          I  Used  I Length I------------I Rec I-----------I-----------------I
 I          I        I        I Gap I Pad  I     I Num I Flg I Min I Max I Avg I
 I----------I--------I--------I-----I------I-----I-----I-----I-----I-----I-----I
         351   100       5064      0     0      1    1   S     5060  5060  5060
         352    99       4994     70     0      3    1   L       21  4948  1663
         355    99       5038     26     0      5    1   L       21  4950  1006
         356   100       5064      0     0      1    1   P     5060  5060  5060
         357   100       5064      0     0      1    1   S     5060  5060  5060
         358    99       5040     24     0      5    1   L       21  4952  1007
         359   100       5064      0     0      1    1   P     5060  5060  5060
         360   100       5064      0     0      1    1   S     5060  5060  5060
         361    99       5042     22     0      5    1   L       21  4954  1007
         362   100       5064      0     0      1    1   P     5060  5060  5060
         363   100       5064      0     0      1    1   S     5060  5060  5060
         364    99       5044     20     0      5    1   L       21  4956  1008
         365    99       5050     14     0     26    0           21   271   194
         620    99       5044     20     0      5    1   L       21  4956  1008
```

```
        621    98       4987     77     0     23    0            21   271   216   ↩
        622   100       5064      0     0      1    1   P     5060  5060  5060   ↩
        624    98       4952    112     0      1    1   L     4948  4948  4948   ↩
        625   100       5064      0     0      1    1   P     5060  5060  5060   ↩
                                                                                 ↩
```

Sample 2: Large Object (LOB) File Processing

This sample shows ADADRU SPACEUSAGE processing of LOB files. Both file 6 and file 8 are LOB files. In the FILE parameter for file 6, the LOBGROUP parameter is omitted, so its base file is not processed. However, a LOBGROUP parameter is specified for file 8, so its base file is processed. ADADRU processes files only after all parameters are consolidated.

```
 A D A D R U   V8.2  SM8   DBID = 00821  Started           2012-05-11  12:37:52

Parameters:
-----------

ADADRU SPACEUSAGE FILE=6,PFILESPACE=DETAIL
ADADRU            FILE=8,PFILESPACE=DETAIL,LOBGROUP


1

DRU203I, ******* SPACEUSAGE-3 started *******     2012-05-11 12:37:52



File group number 1.  FILE=6
-----------------

File group number 2.  FILE=8
-----------------

DRU200I, The input parameters accepted.


LOB GROUPS:
---------------------------
I File with  I Associated I
I LOB fields I  LOB file  I
I------------I------------I
I         9  I         8  I
I            I            I
---------------------------
```

```
1

1
 ************************************************************************
 ************************************************************************
 ************************************************************************
 ****                                                                ****
 ****             F I L E      6 ( BASE-FILE        )               ****
 ****                                                                ****
 ****                       2012-05-11 12:37:52                      ****
 ************************************************************************
 ************************************************************************
 ************************************************************************

 Consolidated input parameters:
 ------------------------------
  File number = 6
            ERRLIMIT      = 1
            LOB-GROUP     = No
            EXP-GROUP     = No
      ASSO  Report type   = SUMMARY + DETAIL
            SAMPLING      = 100    (10 of every 10 records read)
            INCRUSAGE     = 5
            INCRNUMREC    = Auto
            FROM/TO RABN  = 2100         - 2100
            FROM/TO RABN  = 2113         - 2118
            FROM/TO RABN  = 2125         - 2128
      DATA  Report type   = SUMMARY + DETAIL
            SAMPLING      = 100    (10 of every 10 records read)
            INCRUSAGE     = 5
            INCRNUMREC    = Auto
            INCRRECLEN    = Auto
            FROM/TO RABN  = 2382         - 2383

1

 File options (from FCB):
 ------------------------
 Date / Time Loaded ..................... 2010-07-01 11:13:43.11
 Date / Time of last update ............. 2010-07-06 07:05:02.60
 Date / Time of last physical restructure 2010-07-01 11:13:43.11

 ISN Reusage ............................ No
 DATA Storage Reusage ................... No
 MIXDSDEV ............................... No
 Spanned Records Support ................ Yes
 Index compressed ....................... Yes
 2-Byte MU/PE Index ..................... No
 Highest Index Level .................... 3

 LOB fields ............................. Yes. Base file. LOB file = 7
```

```
 LOB file ............................. No
 Expanded ............................. No

1
 I/////////////////////////////////////////////////////////////////I
 II                                                               II
 II     ASSOCIATOR SUMMARY REPORT OF NORMAL INDEX FOR FILE 6      II
 II                                                               II
 I/////////////////////////////////////////////////////////////////I

 Physical extents list
 ---------------------------------------------------------------------
 I Extent I Type I          From           To  I % unused I Device I
 I        I      I          RABN          RABN I  blocks  I  type  I
 I--------I------I-----------------------------I----------I--------I
 I     1  I  NI  I          2100          2100 I      0   I  3390  I
 I     2  I  NI  I          2113          2118 I      0   I  3390  I
 I     3  I  NI  I          2125          2130 I     33   I  3390  I
 I        I      I                             I          I        I
 ---------------------------------------------------------------------

 Range of RABNs being processed according
 to consolidated input parameters
 (unused RABNs at the end of extents excluded)
 -----------------------------------------------
 I Extent I Type I          From           To  I
 I        I      I          RABN          RABN I
 I--------I------I-----------------------------I
 I     1  I  NI  I          2100          2100 I
 I     2  I  NI  I          2113          2118 I
 I     3  I  NI  I          2125          2128 I
 I        I      I                             I
 -----------------------------------------------

 File ......................... 6
 Device Type .................. 3390
 Block Size ................... 2544
 ASSO Padding factor .......... 50%   (1272 bytes)


 Number of allocated blocks ..............................  13
 Number of used blocks ...................................  11
 Number of blocks read (SAMPLING = 100%) .................  11
 Number of unused blocks at the end of extents ...........  2
 Number of unused blocks within the used extents .........  0
 Percent of blocks with free space less than padding ......  18
 Average percent of block filling (unused blocks excluded)   17
 Number of elements in blocks read .......................  409
 Average number of elements per block ....................  37
 Average element length ..................................  11
 Minimum element length ..................................  7
 Maximum element length ..................................  256
```

```
 Number of elements with minimum length ..................  398
 Number of elements with maximum length ..................  8

1

 I================================================I
 II                                              II
 II    DISTRIBUTION REPORTS ASSO NORMAL INDEX    II
 II                   File = 6                   II
 II             Device type = 3390               II
 I================================================I

 Block filling level; Increment = 5%
 ------------------------------------------------
 I      Range of       I    Number of blocks      I
 I   percent filled    I  filled in this range    I
 I--------------------I-------------------------I
 I       1 -      5    I                1         I
 I       6 -     10    I                8         I
 I      51 -     55    I                2         I
 ------------------------------------------------

 ------------------------------------------------
 I   Percent ranges    I  Number of blocks with   I
 I   of free space     I free space in this range I
 I--------------------I-------------------------I
 I      41 -     45    I                2         I
 I      86 -     90    I                8         I
 I      96 -    100    I                1         I
 ------------------------------------------------

1
 Number of elements per block; Increment = Auto
 ------------------------------------------------
 I  Number of elements I          Number          I
 I     per block       I        of blocks         I
 I---------------------I-------------------------I
 I       1 -      3    I                9         I
 I     199 -    201    I                2         I
 ------------------------------------------------

1
 I//////////////////////////////////////////////////////////////I
 II                                                            II
 II          DATA STORAGE SUMMARY REPORT FOR FILE 6            II
 II                                                            II
 I//////////////////////////////////////////////////////////////I

 Physical extents list
 ------------------------------------------------------------------
 I Extent I Type I          From           To  I % unused I Device I
 I        I      I          RABN          RABN I  blocks  I  type  I
```

```
I--------I------I-----------------------------I----------I--------I
I     1  I  DS  I          2382          3881 I     99   I  3390  I
I        I      I                               I          I        I
-------------------------------------------------------------------

Range of RABNs being processed according
to consolidated input parameters
(unused RABNs at the end of extents excluded)
-----------------------------------------------
I Extent I Type I          From           To  I
I        I      I          RABN          RABN I
I--------I------I-----------------------------I
I     1  I  DS  I          2382          2383 I
I        I      I                               I
-----------------------------------------------

File ......................... 6
Device Type .................. 3390
Block Size.................... 5064
DATA Storage Padding factor .. 0%    (0 bytes)


Number of allocated blocks ..............................  1500
Number of used blocks ...................................  2
Number of blocks read (SAMPLING = 100%) .................  2
Number of unused blocks at the end of extents ...........  1498
Number of unused blocks within the used extents .........  0
Percent of blocks with free space less than padding ......  0
Average percent of block filling (unused blocks excluded)   78
Number of records in blocks read ........................  3
   Number of Normal records .............................  2
   Number of Spanned records ............................  1
Number of blocks with Spanned records ...................  2
Average number of records per block .....................  2

Normal records:
   Average record length ................................  19
   Minimum record length ................................  15
   Maximum record length ................................  23
   Number of records with minimum length ................  1
   Number of records with maximum length ................  1

Spanned Record:
   Average record length ................................  7850
   Minimum record length ................................  7850
   Maximum record length ................................  7850
   Number of records with minimum length ................  1
   Number of records with maximum length ................  1
   Average numbers of segments ..........................  2
   Minimum numbers of segments ..........................  2
   Maximum numbers of segments ..........................  2
   Number of Primary Spanned Records ....................  1
```

```
    Number of Secondary Spanned Records...................  1
    Number of AC2 blocks read ............................  1
    Data Storage blocks with secondary segments read ......  1

1

 I================================================I
 II                                              II
 II       DISTRIBUTION REPORTS DATA STORAGE      II
 II                   File = 6                   II
 II             Device type = 3390               II
 I================================================I

 Block filling level; Increment = 5%
 ------------------------------------------------
 I      Range of       I    Number of blocks     I
 I   percent filled    I  filled in this range   I
 I---------------------I-------------------------I
 I      71 -     75    I                1        I
 I      81 -     85    I                1        I
 ------------------------------------------------

 ------------------------------------------------
 I   Percent ranges    I  Number of blocks with  I
 I   of free space     I free space in this range I
 I---------------------I-------------------------I
 I      16 -     20    I                1        I
 I      26 -     30    I                1        I
 ------------------------------------------------

1
 Number of records per block; Increment = Auto
 ------------------------------------------------
 I  Number of records  I          Number         I
 I     per block       I        of blocks        I
 I---------------------I-------------------------I
 I       1 -      6    I                2        I
 ------------------------------------------------

1
 Normal records length; Increment = Auto
 ------------------------------------------------
 I        Length       I          Number         I
 I      of record      I        of records       I
 I---------------------I-------------------------I
 I      13 -     18    I                1        I
 I      19 -     24    I                1        I
 ------------------------------------------------

1
 Spanned records length; Increment = Auto
 ------------------------------------------------
```

```
 I        Length       I         Number           I
 I      of record      I       of records         I
 I---------------------I--------------------------I
 I    7827 -   7852    I                1         I
 ------------------------------------------------

1

1
 ************************************************************************
 ************************************************************************
 ************************************************************************
 ****                                                                ****
 ****             F I L E      8 ( LOB-FILE1        )               ****
 ****                                                                ****
 ****                       2012-05-11 12:37:52                      ****
 ************************************************************************
 ************************************************************************
 ************************************************************************

 Consolidated input parameters:
 ------------------------------
  File number = 8
            ERRLIMIT      = 1
            LOB-GROUP     = Yes
            EXP-GROUP     = No
      ASSO  Report type   = SUMMARY + DETAIL
            SAMPLING      = 100    (10 of every 10 records read)
            INCRUSAGE     = 5
            INCRNUMREC    = Auto
            FROM/TO RABN  = 2441         - 2441
      DATA  Report type   = SUMMARY + DETAIL
            SAMPLING      = 100    (10 of every 10 records read)
            INCRUSAGE     = 5
            INCRNUMREC    = Auto
            INCRRECLEN    = Auto
            FROM/TO RABN  = 7002         - 7067

1

 File options (from FCB):
 ------------------------
 Date / Time Loaded ..................... 2010-07-30 10:08:01.13
 Date / Time of last update ............. 2010-08-03 07:04:22.85
 Date / Time of last physical restructure 2010-07-30 10:08:01.13

 ISN Reusage ............................ No
 DATA Storage Reusage ................... No
 MIXDSDEV ............................... No
 Spanned Records Support ................ No
 Index compressed ....................... Yes
 2-Byte MU/PE Index ..................... No
```

```
 Highest Index Level ..................... 3

 LOB fields ............................. No
 LOB file ............................... Yes. Base file = 9
 Expanded ............................... No

1
 I//////////////////////////////////////////////////////////////////I
 II                                                                  II
 II     ASSOCIATOR SUMMARY REPORT OF NORMAL INDEX FOR FILE 8         II
 II                                                                  II
 I//////////////////////////////////////////////////////////////////I

 Physical extents list
 ---------------------------------------------------------------------
 I Extent I Type I          From           To  I % unused I Device I
 I        I      I          RABN          RABN I  blocks  I  type  I
 I--------I------I----------------------------I----------I--------I
 I     1  I  NI  I          2441          2441 I      0   I  3390  I
 I        I      I                             I          I        I
 ---------------------------------------------------------------------

 Range of RABNs being processed according
 to consolidated input parameters
 (unused RABNs at the end of extents excluded)
 -----------------------------------------------
 I Extent I Type I          From           To  I
 I        I      I          RABN          RABN I
 I--------I------I----------------------------I
 I     1  I  NI  I          2441          2441 I
 I        I      I                             I
 -----------------------------------------------

 File ......................... 8
 Device Type .................. 3390
 Block Size ................... 2544
 ASSO Padding factor .......... 50%   (1272 bytes)


 Number of allocated blocks ..............................  1
 Number of used blocks ...................................  1
 Number of blocks read (SAMPLING = 100%) .................  1
 Number of unused blocks at the end of extents ...........  0
 Number of unused blocks within the used extents .........  0
 Percent of blocks with free space less than padding ......  0
 Average percent of block filling (unused blocks excluded)   29
 Number of elements in blocks read .......................  66
 Average number of elements per block ....................  66
 Average element length ..................................  11
 Minimum element length ..................................  9
 Maximum element length ..................................  20
 Number of elements with minimum length ..................  39
```

```
 Number of elements with maximum length ..................  1

1

 I================================================I
 II                                              II
 II    DISTRIBUTION REPORTS ASSO NORMAL INDEX    II
 II                   File = 8                   II
 II             Device type = 3390               II
 I================================================I

 Block filling level; Increment = 5%
 ------------------------------------------------
 I      Range of       I    Number of blocks      I
 I   percent filled    I  filled in this range    I
 I--------------------I-------------------------I
 I      26 -     30    I                1         I
 ------------------------------------------------

 ------------------------------------------------
 I   Percent ranges    I  Number of blocks with   I
 I   of free space     I free space in this range I
 I--------------------I-------------------------I
 I      71 -     75    I                1         I
 ------------------------------------------------

1
 Number of elements per block; Increment = Auto
 ------------------------------------------------
 I  Number of elements I          Number          I
 I     per block       I        of blocks         I
 I---------------------I-------------------------I
 I      64 -     66    I                1         I
 ------------------------------------------------

1
 I////////////////////////////////////////////////////////////////I
 II                                                              II
 II          DATA STORAGE SUMMARY REPORT FOR FILE 8              II
 II                                                              II
 I////////////////////////////////////////////////////////////////I

 Physical extents list
 ------------------------------------------------------------------------
 I Extent I Type I          From           To  I % unused I Device I
 I        I      I          RABN          RABN I  blocks  I  type  I
 I--------I------I-----------------------------I----------I--------I
 I     1  I  DS  I          7002          8501 I     96   I  3390  I
 I        I      I                             I          I        I
 ------------------------------------------------------------------------

 Range of RABNs being processed according
```

```
 to consolidated input parameters
 (unused RABNs at the end of extents excluded)
 ---------------------------------------------
 I Extent I Type I          From          To  I
 I        I      I          RABN         RABN I
 I--------I------I-----------------------------I
 I     1  I  DS  I          7002         7067 I
 I        I      I                             I
 ---------------------------------------------

 File ......................... 8
 Device Type .................. 3390
 Block Size.................... 5064
 DATA Storage Padding factor .. 10%   (506 bytes)


 Number of allocated blocks ..............................  1500
 Number of used blocks ...................................  66
 Number of blocks read (SAMPLING = 100%) .................  66
 Number of unused blocks at the end of extents ............  1434
 Number of unused blocks within the used extents ..........  0
 Percent of blocks with free space less than padding ......  59
 Average percent of block filling (unused blocks excluded)   75
 Number of records in blocks read ........................  66
    Number of Normal records .............................  66
    Number of Spanned records ............................  n/a
 Number of blocks with Spanned records ...................  n/a
 Average number of records per block .....................  1

 Normal records:
    Average record length ................................  3798
    Minimum record length ................................  1926
    Maximum record length ................................  5046
    Number of records with minimum length ................  3
    Number of records with maximum length ................  39

1

 I================================================I
 II                                              II
 II       DISTRIBUTION REPORTS DATA STORAGE      II
 II                   File = 8                   II
 II             Device type = 3390               II
 I================================================I

 Block filling level; Increment = 5%
 -----------------------------------------------
 I      Range of       I    Number of blocks      I
 I   percent filled    I  filled in this range    I
 I--------------------I--------------------------I
 I      36 -     40    I               27         I
 I      96 -    100    I               39         I
```

```
 ------------------------------------------------

 ------------------------------------------------
 I   Percent ranges    I  Number of blocks with   I
 I   of free space     I free space in this range I
 I---------------------I--------------------------I
 I       1 -      5    I               39         I
 I      56 -     60    I               24         I
 I      61 -     65    I                3         I
 ------------------------------------------------

1
 Number of records per block; Increment = Auto
 ------------------------------------------------
 I  Number of records  I          Number          I
 I     per block       I        of blocks         I
 I---------------------I--------------------------I
 I       1 -      6    I               66         I
 ------------------------------------------------

1
 Normal records length; Increment = Auto
 ------------------------------------------------
 I        Length       I          Number          I
 I      of record      I        of records        I
 I---------------------I--------------------------I
 I    1921 -   1926    I                3         I
 I    2005 -   2010    I               24         I
 I    5041 -   5046    I               39         I
 ------------------------------------------------

1

1
 ************************************************************************
 ************************************************************************
 ************************************************************************
 ****                                                                ****
 ****             F I L E      9 ( BASE-FILE1       )               ****
 ****                                                                ****
 ****                       2012-05-11 12:37:52                      ****
 ************************************************************************
 ************************************************************************
 ************************************************************************

 Consolidated input parameters:
 ------------------------------
  File number = 9
            ERRLIMIT      = 1
            LOB-GROUP     = Yes
            EXP-GROUP     = No
      ASSO  Report type   = SUMMARY + DETAIL
```

```
            SAMPLING      = 100    (10 of every 10 records read)
            INCRUSAGE     = 5
            INCRNUMREC    = Auto
            FROM/TO RABN  = 2436         - 2436
            FROM/TO RABN  = 2449         - 2454
            FROM/TO RABN  = 2461         - 2466
      DATA  Report type   = SUMMARY + DETAIL
            SAMPLING      = 100    (10 of every 10 records read)
            INCRUSAGE     = 5
            INCRNUMREC    = Auto
            INCRRECLEN    = Auto
            FROM/TO RABN  = 5502         - 5507

1

 File options (from FCB):
 ------------------------
 Date / Time Loaded ..................... 2010-07-30 10:07:59.03
 Date / Time of last update ............. 2010-08-03 07:04:22.85
 Date / Time of last physical restructure 2010-07-30 10:07:59.03

 ISN Reusage ............................ No
 DATA Storage Reusage ................... No
 MIXDSDEV ............................... No
 Spanned Records Support ................ Yes
 Index compressed ....................... Yes
 2-Byte MU/PE Index ..................... No
 Highest Index Level .................... 3

 LOB fields ............................. Yes. Base file. LOB file = 8
 LOB file ............................... No
 Expanded ............................... No

1
 I/////////////////////////////////////////////////////////////I
 II                                                           II
 II     ASSOCIATOR SUMMARY REPORT OF NORMAL INDEX FOR FILE 9      II
 II                                                           II
 I/////////////////////////////////////////////////////////////I

 Physical extents list
 ---------------------------------------------------------------------
 I Extent I Type I          From           To  I % unused I Device I
 I        I      I          RABN          RABN I  blocks  I  type  I
 I--------I------I-----------------------------I----------I--------I
 I     1  I  NI  I          2436          2436 I      0   I  3390  I
 I     2  I  NI  I          2449          2454 I      0   I  3390  I
 I     3  I  NI  I          2461          2466 I      0   I  3390  I
 I        I      I                             I          I        I
 ---------------------------------------------------------------------

 Range of RABNs being processed according
```

```
to consolidated input parameters
(unused RABNs at the end of extents excluded)
-----------------------------------------------
I Extent I Type I          From           To  I
I        I      I          RABN          RABN I
I--------I------I-----------------------------I
I     1  I  NI  I          2436          2436 I
I     2  I  NI  I          2449          2454 I
I     3  I  NI  I          2461          2466 I
I        I      I                            I
-----------------------------------------------

File ......................... 9
Device Type .................. 3390
Block Size ................... 2544
ASSO Padding factor .......... 50%   (1272 bytes)


Number of allocated blocks ..............................  13
Number of used blocks ...................................  13
Number of blocks read (SAMPLING = 100%) .................  13
Number of unused blocks at the end of extents ............  0
Number of unused blocks within the used extents ..........  0
Percent of blocks with free space less than padding ......  31
Average percent of block filling (unused blocks excluded)   26
Number of elements in blocks read .......................  425
Average number of elements per block ....................  33
Average element length ..................................  20
Minimum element length ..................................  13
Maximum element length ..................................  256
Number of elements with minimum length ..................  396
Number of elements with maximum length ..................  8

1

I================================================I
II                                              II
II    DISTRIBUTION REPORTS ASSO NORMAL INDEX    II
II                   File = 9                   II
II             Device type = 3390               II
I================================================I

Block filling level; Increment = 5%
-------------------------------------------------
I      Range of       I    Number of blocks      I
I   percent filled    I  filled in this range    I
I--------------------I--------------------------I
I       1 -      5    I                1         I
I      11 -     15    I                4         I
I      16 -     20    I                4         I
I      46 -     50    I                2         I
I      51 -     55    I                2         I
```

```
 ------------------------------------------------


 ------------------------------------------------
 I   Percent ranges    I  Number of blocks with   I
 I   of free space     I free space in this range I
 I---------------------I--------------------------I
 I      46 -     50    I                4         I
 I      81 -     85    I                4         I
 I      86 -     90    I                4         I
 I      96 -    100    I                1         I
 ------------------------------------------------


1
 Number of elements per block; Increment = Auto
 ------------------------------------------------
 I  Number of elements I          Number          I
 I     per block       I        of blocks         I
 I---------------------I--------------------------I
 I       1 -      3    I                9         I
 I      97 -     99    I                2         I
 I     100 -    102    I                2         I
 ------------------------------------------------


1
 I/////////////////////////////////////////////////////////////////I
 II                                                               II
 II          DATA STORAGE SUMMARY REPORT FOR FILE 9               II
 II                                                               II
 I/////////////////////////////////////////////////////////////////I

 Physical extents list
 ------------------------------------------------------------------------
 I Extent I Type I          From           To  I % unused I Device I
 I        I      I          RABN          RABN I  blocks  I  type  I
 I--------I------I----------------------------I----------I--------I
 I     1  I  DS  I          5502          7001 I     99   I  3390  I
 I        I      I                            I          I        I
 ------------------------------------------------------------------------

 Range of RABNs being processed according
 to consolidated input parameters
 (unused RABNs at the end of extents excluded)
 ------------------------------------------------
 I Extent I Type I          From           To  I
 I        I      I          RABN          RABN I
 I--------I------I----------------------------I
 I     1  I  DS  I          5502          5507 I
 I        I      I                            I
 ------------------------------------------------

 File ......................... 9
 Device Type .................. 3390
```

```
Block Size..................... 5064
DATA Storage Padding factor .. 0%    (0 bytes)


Number of allocated blocks ..............................  1500
Number of used blocks ...................................  6
Number of blocks read (SAMPLING = 100%) .................  6
Number of unused blocks at the end of extents ............  1494
Number of unused blocks within the used extents ..........  0
Percent of blocks with free space less than padding ......  0
Average percent of block filling (unused blocks excluded)   78
Number of records in blocks read ........................  3
   Number of Normal records .............................  0
   Number of Spanned records ............................  3
Number of blocks with Spanned records ...................  6
Average number of records per block .....................  1

Normal records:
   Average record length ................................  0
   Minimum record length ................................  32767
   Maximum record length ................................  0
   Number of records with minimum length ................  0
   Number of records with maximum length ................  0

Spanned Record:
   Average record length ................................  7850
   Minimum record length ................................  7850
   Maximum record length ................................  7850
   Number of records with minimum length ................  3
   Number of records with maximum length ................  3
   Average numbers of segments ..........................  2
   Minimum numbers of segments ..........................  2
   Maximum numbers of segments ..........................  2
   Number of Primary Spanned Records ....................  3
   Number of Secondary Spanned Records...................  3
   Number of AC2 blocks read ............................  1
   Data Storage blocks with secondary segments read ......  3

1

 I================================================I
 II                                              II
 II       DISTRIBUTION REPORTS DATA STORAGE      II
 II                   File = 9                   II
 II             Device type = 3390               II
 I================================================I

 Block filling level; Increment = 5%
 ------------------------------------------------
 I      Range of       I    Number of blocks      I
 I   percent filled    I  filled in this range    I
 I--------------------I-------------------------I
```

```
 I      71 -     75    I                3         I
 I      81 -     85    I                3         I
 ------------------------------------------------


 ------------------------------------------------
 I   Percent ranges    I  Number of blocks with   I
 I   of free space     I free space in this range I
 I---------------------I--------------------------I
 I      16 -     20    I                3         I
 I      26 -     30    I                3         I
 ------------------------------------------------

1
 Number of records per block; Increment = Auto
 ------------------------------------------------
 I  Number of records  I          Number          I
 I     per block       I        of blocks         I
 I---------------------I--------------------------I
 I       1 -      6    I                6         I
 ------------------------------------------------

 DRU163W, WARNING: No data for "Normal records length" report


1
 Spanned records length; Increment = Auto
 ------------------------------------------------
 I        Length       I         Number           I
 I      of record      I       of records         I
 I---------------------I--------------------------I
 I    7827 -   7852    I                3         I
 ------------------------------------------------




1
 ************************************************************************
 ************************************************************************
 ************************************************************************
 ****                                                                ****
 ****       F i l e s   s t a t i s t i c a l   r e p o r t        ****
 ****                                                                ****
 ************************************************************************
 ************************************************************************
 ************************************************************************

 ---------------------------------------------------------------------------
 I F I  Duration I     ASSOCIATOR (NI)    I      DATA STORAGE      I ParmI Run I
 I i IHH:MM:SS.thI------------------------I------------------------I-----I-----I
 I l I           I  Number  I E I Extents I  Number  I E I Extents I E W I E W I
 I e I           I of RABNs I r I---------I of RABNs I r I---------I r n I r n I
 I   I           I   read   I r I FCB Run I   read   I r I FCB Run I r g I r g I
```

```
 I---I----------I---------I---I----I----I---------I---I----I----I--I--I--I--I
    6 00:00:00.05         11   0    3    3          2   0    1    1  0  0  0  0
    8 00:00:00.05          1   0    1    1         66   0    1    1  0  0  0  0
    9 00:00:00.06         13   0    3    3          6   0    1    1  0  0  0  1



1

 DRU205I, ******* SPACEUSAGE-3 ended *******     CC = 04

DDDETAIL SPACEUSAGE-3

1File options (from FCB):
 ------------------------
 Date / Time Loaded ..................... 2010-07-01 11:13:43.11
 Date / Time of last update ............. 2010-07-06 07:05:02.60
 Date / Time of last physical restructure 2010-07-01 11:13:43.11

 ISN Reusage ............................ No
 DATA Storage Reusage ................... No
 MIXDSDEV ............................... No
 Spanned Records Support ................ Yes
 Index compressed ....................... Yes
 2-Byte MU/PE Index ..................... No
 Highest Index Level .................... 3

 LOB fields ............................. Yes. Base file. LOB file = 7
 LOB file ............................... No
 Expanded ............................... No


 I//////////////////////////////////////////////////////////////////I
 II                                                                  II
 II    ASSOCIATOR DETAIL REPORT OF NORMAL INDEX FOR FILE 6           II
 II                                                                  II
 I//////////////////////////////////////////////////////////////////I

 Device Type ...........  3390
 Block Size ............. 2544
 ASSO Padding Factor .... 50% (1272 bytes)

1File 6                                                    Page 1
 ---------------------------------------------------------------------------
 I   RABN   I   %     I Block  I Free Space I Num  I Element lengths I
 I          I  Used   I length I------------I elem I-----------------I
 I          I         I        I Gap I Pad  I      I Min I Max I Avg I
 I----------I---------I--------I-----I------I------I-----I-----I-----I
        2100     10       259    1013   1272      1   256   256   256
        2113     10       259    1013   1272      1   256   256   256
        2114     10       259    1013   1272      1   256   256   256
        2115     10       259    1013   1272      1   256   256   256
```

```
        2116     10       259    1013   1272      1   256   256   256
        2117     10       259    1013   1272      1   256   256   256
        2118     10       259    1013   1272      1   256   256   256
        2125     10       259    1013   1272      1   256   256   256
        2126      1        11    1261   1272      1     8     8     8
        2127     55      1405       0   1139    200     7     9     7
        2128     55      1405       0   1139    200     7     9     7



 I//////////////////////////////////////////////////////////////////I
 II                                                                II
 II            DATA STORAGE DETAIL REPORT FOR FILE 6               II
 II                                                                II
 I//////////////////////////////////////////////////////////////////I

 Device Type ...........  3390
 Block Size ............. 5064
 DATA Padding Factor .... 0% (0 bytes)

1File 6                                                        Page 1
 ------------------------------------------------------------------------------
 I   RABN   I   %    I Block  I Free space I Num I  Spanned  I Record lengths  I
 I          I  Used  I Length I------------I Rec I-----------I-----------------I
 I          I        I        I Gap I Pad  I     I Num I Flg I Min I Max I Avg I
 I----------I--------I--------I-----I------I-----I-----I-----I-----I-----I-----I
        2382    82       4129    935     0      1    1   P     4125  4125  4125
        2383    74       3767   1297     0      3    1   L       15  3725  1254


1File options (from FCB):
 ------------------------
 Date / Time Loaded ..................... 2010-07-30 10:08:01.13
 Date / Time of last update ............. 2010-08-03 07:04:22.85
 Date / Time of last physical restructure 2010-07-30 10:08:01.13

 ISN Reusage ............................ No
 DATA Storage Reusage ................... No
 MIXDSDEV ............................... No
 Spanned Records Support ................ No
 Index compressed ....................... Yes
 2-Byte MU/PE Index ..................... No
 Highest Index Level .................... 3

 LOB fields ............................. No
 LOB file ............................... Yes. Base file = 9
 Expanded ............................... No


 I//////////////////////////////////////////////////////////////////I
 II                                                                II
 II    ASSOCIATOR DETAIL REPORT OF NORMAL INDEX FOR FILE 8         II
```

```
 II                                                                  II
 I//////////////////////////////////////////////////////////////////I

 Device Type ...........  3390
 Block Size ............. 2544
 ASSO Padding Factor .... 50% (1272 bytes)

1File 8                                                    Page 1
 ------------------------------------------------------------------------
 I   RABN   I   %     I Block  I Free Space I Num  I Element lengths I
 I          I  Used   I length I------------I elem I-----------------I
 I          I         I        I Gap I Pad  I      I Min I Max I Avg I
 I----------I---------I--------I-----I------I------I-----I-----I-----I
        2441     29       749     523   1272     66     9    20    11



 I//////////////////////////////////////////////////////////////////I
 II                                                                  II
 II            DATA STORAGE DETAIL REPORT FOR FILE 8                 II
 II                                                                  II
 I//////////////////////////////////////////////////////////////////I

 Device Type ...........  3390
 Block Size ............. 5064
 DATA Padding Factor .... 10% (506 bytes)

1File 8                                                          Page 1
 ------------------------------------------------------------------------------
 I   RABN   I   %    I Block  I Free space I Num I  Spanned  I Record lengths  I
 I          I  Used  I Length I------------I Rec I-----------I-----------------I
 I          I        I        I Gap I Pad  I     I Num I Flg I Min I Max I Avg I
 I----------I--------I--------I-----I------I-----I-----I-----I-----I-----I-----I
        7002    99       5050      0    14      1    0          5046  5046  5046
        7003    40       2010   2548   506      1    0          2006  2006  2006
        7004    99       5050      0    14      1    0          5046  5046  5046
        7005    40       2010   2548   506      1    0          2006  2006  2006
        7006    99       5050      0    14      1    0          5046  5046  5046
        7007    99       5050      0    14      1    0          5046  5046  5046
        7008    99       5050      0    14      1    0          5046  5046  5046
        7009    99       5050      0    14      1    0          5046  5046  5046
        7010    99       5050      0    14      1    0          5046  5046  5046
        7011    38       1930   2628   506      1    0          1926  1926  1926
        7012    99       5050      0    14      1    0          5046  5046  5046
        7013    40       2010   2548   506      1    0          2006  2006  2006
        7014    99       5050      0    14      1    0          5046  5046  5046
        7015    40       2010   2548   506      1    0          2006  2006  2006
        7016    99       5050      0    14      1    0          5046  5046  5046
        7017    40       2010   2548   506      1    0          2006  2006  2006
        7018    99       5050      0    14      1    0          5046  5046  5046
        7019    40       2010   2548   506      1    0          2006  2006  2006
        7020    99       5050      0    14      1    0          5046  5046  5046
```

```
        7021    40        2010   2548   506       1    0          2006  2006  2006
        7022    99        5050      0    14       1    0          5046  5046  5046
        7023    40        2010   2548   506       1    0          2006  2006  2006
        7024    99        5050      0    14       1    0          5046  5046  5046
        7025    40        2010   2548   506       1    0          2006  2006  2006
        7026    99        5050      0    14       1    0          5046  5046  5046
        7027    40        2010   2548   506       1    0          2006  2006  2006
        7028    99        5050      0    14       1    0          5046  5046  5046
        7029    99        5050      0    14       1    0          5046  5046  5046
        7030    99        5050      0    14       1    0          5046  5046  5046
        7031    99        5050      0    14       1    0          5046  5046  5046
        7032    99        5050      0    14       1    0          5046  5046  5046
        7033    38        1930   2628   506       1    0          1926  1926  1926
        7034    99        5050      0    14       1    0          5046  5046  5046
        7035    40        2010   2548   506       1    0          2006  2006  2006
        7036    99        5050      0    14       1    0          5046  5046  5046
        7037    40        2010   2548   506       1    0          2006  2006  2006
        7038    99        5050      0    14       1    0          5046  5046  5046
        7039    40        2010   2548   506       1    0          2006  2006  2006
        7040    99        5050      0    14       1    0          5046  5046  5046
        7041    40        2010   2548   506       1    0          2006  2006  2006
        7042    99        5050      0    14       1    0          5046  5046  5046
        7043    40        2010   2548   506       1    0          2006  2006  2006
        7044    99        5050      0    14       1    0          5046  5046  5046
        7045    40        2010   2548   506       1    0          2006  2006  2006
        7046    99        5050      0    14       1    0          5046  5046  5046
        7047    40        2010   2548   506       1    0          2006  2006  2006
        7048    99        5050      0    14       1    0          5046  5046  5046
        7049    40        2010   2548   506       1    0          2006  2006  2006
        7050    99        5050      0    14       1    0          5046  5046  5046
        7051    99        5050      0    14       1    0          5046  5046  5046
        7052    99        5050      0    14       1    0          5046  5046  5046
        7053    99        5050      0    14       1    0          5046  5046  5046
        7054    99        5050      0    14       1    0          5046  5046  5046
        7055    38        1930   2628   506       1    0          1926  1926  1926
        7056    99        5050      0    14       1    0          5046  5046  5046
        7057    40        2010   2548   506       1    0          2006  2006  2006
        7058    99        5050      0    14       1    0          5046  5046  5046
        7059    40        2010   2548   506       1    0          2006  2006  2006
        7060    99        5050      0    14       1    0          5046  5046  5046
        7061    40        2010   2548   506       1    0          2006  2006  2006
        7062    99        5050      0    14       1    0          5046  5046  5046
        7063    40        2010   2548   506       1    0          2006  2006  2006
        7064    99        5050      0    14       1    0          5046  5046  5046
        7065    40        2010   2548   506       1    0          2006  2006  2006


1File 8                                                                  Page 2
 ------------------------------------------------------------------------------
 I   RABN   I   %    I Block  I Free space I Num I  Spanned  I Record lengths  I
 I          I  Used  I Length I------------I Rec I-----------I-----------------I
 I          I        I        I Gap I Pad  I     I Num I Flg I Min I Max I Avg I
```

```
 I----------I--------I--------I-----I------I-----I-----I-----I-----I-----I-----I
        7066    99       5050      0    14      1    0         5046  5046  5046
        7067    40       2010   2548   506      1    0         2006  2006  2006


1File options (from FCB):
 ------------------------
 Date / Time Loaded ..................... 2010-07-30 10:07:59.03
 Date / Time of last update ............. 2010-08-03 07:04:22.85
 Date / Time of last physical restructure 2010-07-30 10:07:59.03

 ISN Reusage ............................ No
 DATA Storage Reusage ................... No
 MIXDSDEV ............................... No
 Spanned Records Support ................ Yes
 Index compressed ....................... Yes
 2-Byte MU/PE Index ..................... No
 Highest Index Level .................... 3

 LOB fields ............................. Yes. Base file. LOB file = 8
 LOB file ............................... No
 Expanded ............................... No


 I//////////////////////////////////////////////////////////////I
 II                                                            II
 II    ASSOCIATOR DETAIL REPORT OF NORMAL INDEX FOR FILE 9     II
 II                                                            II
 I//////////////////////////////////////////////////////////////I

 Device Type ...........  3390
 Block Size ............. 2544
 ASSO Padding Factor .... 50% (1272 bytes)

1File 9                                                  Page 1
 ------------------------------------------------------------------------
 I   RABN   I   %     I Block  I Free Space I Num  I Element lengths I
 I          I  Used   I length I------------I elem I-----------------I
 I          I         I        I Gap I Pad  I      I Min I Max I Avg I
 I----------I---------I--------I-----I------I------I-----I-----I-----I
        2436     14       365     907   1272      3    53   256   120
        2449     18       465     807   1272      3   103   256   154
        2450     14       365     907   1272      3    53   256   120
        2451     18       465     807   1272      3   103   256   154
        2452     14       365     907   1272      3    53   256   120
        2453     18       465     807   1272      3   103   256   154
        2454     14       365     907   1272      3    53   256   120
        2461     18       465     807   1272      3   103   256   154
        2462      1        17    1255   1272      1    14    14    14
        2463     50      1279       0   1265     98    13    15    13
        2464     50      1279       0   1265     98    13    15    13
        2465     52      1331       0   1213    102    13    15    13
```

```
        2466     52      1331       0   1213    102    13    15    13



 I//////////////////////////////////////////////////////////////////I
 II                                                                II
 II            DATA STORAGE DETAIL REPORT FOR FILE 9               II
 II                                                                II
 I//////////////////////////////////////////////////////////////////I

 Device Type ...........  3390
 Block Size ............. 5064
 DATA Padding Factor .... 0% (0 bytes)

1File 9                                                        Page 1
 ------------------------------------------------------------------------------
 I   RABN   I   %    I Block  I Free space I Num I  Spanned  I Record lengths  I
 I          I  Used  I Length I------------I Rec I-----------I-----------------I
 I          I        I        I Gap I Pad  I     I Num I Flg I Min I Max I Avg I
 I----------I--------I--------I-----I------I-----I-----I-----I-----I-----I-----I
        5502    82       4129    935     0      1    1   P     4125  4125  4125
        5503    74       3729   1335     0      1    1   L     3725  3725  3725
        5504    82       4129    935     0      1    1   P     4125  4125  4125
        5505    74       3729   1335     0      1    1   L     3725  3725  3725
        5506    82       4129    935     0      1    1   P     4125  4125  4125
        5507    74       3729   1335     0      1    1   L     3725  3725  3725
```

Sample 3: SAMPLING Parameter Use

This sample shows processing of SAMPLING parameter by the ADADRU SPACEUSAGE utility function.

```
 ADADRU SPACEUSAGE FILE=1,PFILESPACE=SUMMARY,SAMPLING=30
 ADADRU            FILE=1,PASSOSPACE=DETAIL,SAMPLING=40
 ADADRU            FILE=1,PDATASPACE=DETAIL,SAMPLING=50









1

 DRU203I, ******* SPACEUSAGE-4 started *******     2012-05-11 12:37:52



 File group number 1.  FILE=1
 -----------------
```

```
 File group number 2.  FILE=1
 -----------------

 File group number 3.  FILE=1
 -----------------

 DRU200I, The input parameters accepted.


1

1
 ************************************************************************
 ************************************************************************
 ************************************************************************
 ****                                                                ****
 ****             F I L E      1 ( EMPLOYEES        )               ****
 ****                                                                ****
 ****                        2012-05-11 12:37:52                     ****
 ************************************************************************
 ************************************************************************
 ************************************************************************


 Consolidated input parameters:
 ------------------------------
  File number = 1
            ERRLIMIT      = 1
            LOB-GROUP     = No
            EXP-GROUP     = No
      ASSO  Report type   = SUMMARY + DETAIL
            SAMPLING      = 40     (4  of every 10 records read)
            INCRUSAGE     = 5
            INCRNUMREC    = Auto
            FROM/TO RABN  = 986          - 1047
      DATA  Report type   = SUMMARY + DETAIL
            SAMPLING      = 50     (5  of every 10 records read)
            INCRUSAGE     = 5
            INCRNUMREC    = Auto
            INCRRECLEN    = Auto
            FROM/TO RABN  = 6            - 52

1

 File options (from FCB):
 ------------------------
 Date / Time Loaded ..................... 2009-10-05 13:36:16.33
 Date / Time of last update ............. 2011-02-12 10:09:37.80
 Date / Time of last physical restructure 2009-10-05 13:36:16.33

 ISN Reusage ........................... Yes
```

```
 DATA Storage Reusage ................... Yes
 MIXDSDEV ............................... Yes
 Spanned Records Support ................ Yes
 Index compressed ....................... No
 2-Byte MU/PE Index ..................... No
 Highest Index Level .................... 3

 LOB fields ............................. No
 LOB file ............................... No
 Expanded ............................... No

1
 I/////////////////////////////////////////////////////////////////I
 II                                                               II
 II     ASSOCIATOR SUMMARY REPORT OF NORMAL INDEX FOR FILE 1      II
 II                                                               II
 I/////////////////////////////////////////////////////////////////I

 Physical extents list
 ---------------------------------------------------------------------
 I Extent I Type I          From           To  I % unused I Device I
 I        I      I          RABN          RABN I  blocks  I  type  I
 I--------I------I-----------------------------I----------I--------I
 I     1  I  NI  I           986          1255 I     77   I  3390  I
 I        I      I                             I          I        I
 ---------------------------------------------------------------------

 Range of RABNs being processed according
 to consolidated input parameters
 (unused RABNs at the end of extents excluded)
 -----------------------------------------------
 I Extent I Type I          From           To  I
 I        I      I          RABN          RABN I
 I--------I------I-----------------------------I
 I     1  I  NI  I           986          1047 I
 I        I      I                             I
 -----------------------------------------------

 File ......................... 1
 Device Type .................. 3390
 Block Size ................... 2544
 ASSO Padding factor .......... 10%   (254 bytes)


 Number of allocated blocks ..............................  270
 Number of used blocks ...................................  62
 Number of blocks read (SAMPLING = 40% ) .................  26
 Number of unused blocks at the end of extents ............  208
 Number of unused blocks within the used extents ..........  0
 Percent of blocks with free space less than padding ......  0
 Average percent of block filling (unused blocks excluded)   80
 Number of elements in blocks read .......................  2668
```

```
 Average number of elements per block ....................  103
 Average element length ..................................  19
 Minimum element length ..................................  7
 Maximum element length ..................................  631
 Number of elements with minimum length ..................  1
 Number of elements with maximum length ..................  1

1

 I================================================I
 II                                              II
 II    DISTRIBUTION REPORTS ASSO NORMAL INDEX    II
 II                   File = 1                   II
 II             Device type = 3390               II
 I================================================I

 Block filling level; Increment = 5%
 ------------------------------------------------
 I      Range of       I    Number of blocks      I
 I   percent filled    I  filled in this range    I
 I--------------------I--------------------------I
 I       1 -      5    I                2         I
 I      41 -     45    I                1         I
 I      66 -     70    I                1         I
 I      86 -     90    I               22         I
 ------------------------------------------------

 ------------------------------------------------
 I   Percent ranges    I  Number of blocks with   I
 I   of free space     I free space in this range I
 I--------------------I--------------------------I
 I       6 -     10    I               17         I
 I      11 -     15    I                5         I
 I      31 -     35    I                1         I
 I      56 -     60    I                1         I
 I      96 -    100    I                2         I
 ------------------------------------------------

1
 Number of elements per block; Increment = Auto
 ------------------------------------------------
 I  Number of elements I          Number          I
 I     per block       I        of blocks         I
 I--------------------I--------------------------I
 I       1 -      3    I                1         I
 I       4 -      6    I                2         I
 I       7 -      9    I                1         I
 I      61 -     63    I                1         I
 I      70 -     72    I                2         I
 I      79 -     81    I                2         I
 I      85 -     87    I                1         I
 I      94 -     96    I                1         I
```

```
 I      97 -     99    I                1         I
 I     103 -    105    I                1         I
 I     106 -    108    I                1         I
 I     115 -    117    I                1         I
 I     118 -    120    I                1         I
 I     124 -    126    I                2         I
 I     127 -    129    I                2         I
 I     175 -    177    I                6         I
 ------------------------------------------------


1
 I/////////////////////////////////////////////////////////////////I
 II                                                               II
 II          DATA STORAGE SUMMARY REPORT FOR FILE 1               II
 II                                                               II
 I/////////////////////////////////////////////////////////////////I

 Physical extents list
 ---------------------------------------------------------------------
 I Extent I Type I          From           To  I % unused I Device I
 I        I      I          RABN          RABN I  blocks  I  type  I
 I--------I------I-----------------------------I----------I--------I
 I     1  I  DS  I             6           155 I     69   I  3390  I
 I        I      I                             I          I        I
 ---------------------------------------------------------------------

 Range of RABNs being processed according
 to consolidated input parameters
 (unused RABNs at the end of extents excluded)
 ------------------------------------------------
 I Extent I Type I          From           To  I
 I        I      I          RABN          RABN I
 I--------I------I-----------------------------I
 I     1  I  DS  I             6            52 I
 I        I      I                             I
 ------------------------------------------------

 File ......................... 1
 Device Type .................. 3390
 Block Size.................... 5064
 DATA Storage Padding factor .. 10%   (506 bytes)


 Number of allocated blocks ..............................  150
 Number of used blocks ...................................  47
 Number of blocks read (SAMPLING = 50% ) .................  27
 Number of unused blocks at the end of extents ............  103
 Number of unused blocks within the used extents ..........  16
 Percent of blocks with free space less than padding ......  11
 Average percent of block filling (unused blocks excluded)   33
 Number of records in blocks read ........................  249
    Number of Normal records .............................  249
```

```
   Number of Spanned records ..........................  0
Number of blocks with Spanned records .................  0
Average number of records per block ...................  9

Normal records:
   Average record length ..............................  182
   Minimum record length ..............................  31
   Maximum record length ..............................  253
   Number of records with minimum length ..............  1
   Number of records with maximum length ..............  1

Spanned Record:
   Average record length ..............................  0
   Minimum record length ..............................  0
   Maximum record length ..............................  0
   Number of records with minimum length ..............  0
   Number of records with maximum length ..............  0
   Average numbers of segments ........................  0
   Minimum numbers of segments ........................  0
   Maximum numbers of segments ........................  0
   Number of Primary Spanned Records ..................  0
   Number of Secondary Spanned Records.................  0
   Number of AC2 blocks read ..........................  0
   Data Storage blocks with secondary segments read ......  0

1

 I================================================I
 II                                              II
 II       DISTRIBUTION REPORTS DATA STORAGE      II
 II                    File = 1                  II
 II             Device type = 3390               II
 I================================================I

 Block filling level; Increment = 5%
 ------------------------------------------------
 I      Range of       I    Number of blocks      I
 I   percent filled    I  filled in this range    I
 I--------------------I--------------------------I
 I       1 -      5    I                1         I
 I      86 -     90    I                7         I
 I      91 -     95    I                3         I
 ------------------------------------------------

 ------------------------------------------------
 I   Percent ranges    I  Number of blocks with   I
 I   of free space     I free space in this range I
 I--------------------I--------------------------I
 I       6 -     10    I                4         I
 I      11 -     15    I                6         I
 I      96 -    100    I                1         I
 ------------------------------------------------
```

```
1
 Number of records per block; Increment = Auto
 ------------------------------------------------
 I  Number of records  I          Number          I
 I     per block       I        of blocks         I
 I---------------------I--------------------------I
 I       1 -      6    I                1         I
 I      19 -     24    I                6         I
 I      25 -     30    I                4         I
 ------------------------------------------------

1
 Normal records length; Increment = Auto
 ------------------------------------------------
 I        Length       I          Number          I
 I      of record      I        of records        I
 I---------------------I--------------------------I
 I       1 -     96    I                1         I
 I     133 -    138    I                5         I
 I     139 -    144    I                4         I
 I     145 -    150    I                6         I
 I     151 -    156    I               15         I
 I     157 -    162    I               24         I
 I     163 -    168    I               22         I
 I     169 -    174    I               18         I
 I     175 -    180    I               18         I
 I     181 -    186    I               24         I
 I     187 -    192    I               27         I
 I     193 -    198    I               26         I
 I     199 -    204    I               20         I
 I     205 -    210    I               11         I
 I     211 -    216    I               10         I
 I     217 -    222    I                8         I
 I     223 -    228    I                3         I
 I     229 -    234    I                5         I
 I     235 -    252    I                1         I
 I     253 -    348    I                1         I
 ------------------------------------------------

 DRU163W, WARNING: No data for "Spanned records length" report




1
 ************************************************************************
 ************************************************************************
 ************************************************************************
 ****                                                                ****
 ****       F i l e s   s t a t i s t i c a l   r e p o r t        ****
 ****                                                                ****
```

```
 ************************************************************************
 ************************************************************************
 ************************************************************************

 ------------------------------------------------------------------------------
 I F I  Duration I     ASSOCIATOR (NI)    I      DATA STORAGE      I ParmI Run I
 I i IHH:MM:SS.thI------------------------I------------------------I-----I-----I
 I l I           I  Number  I E I Extents I  Number  I E I Extents I E W I E W I
 I e I           I of RABNs I r I---------I of RABNs I r I---------I r n I r n I
 I   I           I   read   I r I FCB Run I   read   I r I FCB Run I r g I r g I
 I---I-----------I----------I---I----I----I----------I---I----I----I--I--I--I--I
    1 00:00:00.12         26   0    1    1         27   0    1    1  0  0  0  1



1

 DRU205I, ******* SPACEUSAGE-4 ended *******     CC = 04
 DDDETAIL SPACEUSAGE-4

1File options (from FCB):
 ------------------------
 Date / Time Loaded ..................... 2009-10-05 13:36:16.33
 Date / Time of last update ............. 2011-02-12 10:09:37.80
 Date / Time of last physical restructure 2009-10-05 13:36:16.33

 ISN Reusage ............................ Yes
 DATA Storage Reusage ................... Yes
 MIXDSDEV ............................... Yes
 Spanned Records Support ................ Yes
 Index compressed ....................... No
 2-Byte MU/PE Index ..................... No
 Highest Index Level .................... 3

 LOB fields ............................. No
 LOB file ............................... No
 Expanded ............................... No


 I//////////////////////////////////////////////////////////////////I
 II                                                                  II
 II    ASSOCIATOR DETAIL REPORT OF NORMAL INDEX FOR FILE 1           II
 II                                                                  II
 I//////////////////////////////////////////////////////////////////I

 Device Type ...........  3390
 Block Size ............. 2544
 ASSO Padding Factor .... 10% (254 bytes)

1File 1                                                  Page 1
 ---------------------------------------------------------------------
 I   RABN   I   %     I Block  I Free Space I Num  I Element lengths I
```

```
 I          I  Used   I length I-----------I elem I-----------------I
 I          I         I        I Gap I Pad  I      I Min I Max I Avg I
 I----------I---------I--------I-----I------I------I-----I-----I-----I
         986     90      2278      12    254    175    13    13    13
         987     90      2278      12    254    175    13    13    13
         988     90      2278      12    254    175    13    13    13
         989     90      2278      12    254    175    13    13    13
         990     90      2278      12    254    175    13    13    13
         991     90      2278      12    254    175    13    13    13
        1004     90      2283       7    254    115    10    85    19
        1005      1        31    2259    254      2     9    19    14
        1006     90      2284       6    254     62    11   176    36
        1007     90      2284       6    254     98    11   305    23
        1008      3        81    2209    254      6    11    23    13
        1009     90      2285       5    254     81     7    93    28
        1022     90      2286       4    254      8    30   631   285
        1023     44      1114    1176    254      5    18   556   222
        1024     90      2289       1    254    127    14    23    18
        1025     89      2275      15    254    127    14    28    17
        1026     90      2283       7    254    126    14    24    18
        1027     90      2283       7    254    126    14    25    18
        1040     90      2277      13    254    104    18    45    21
        1041     89      2265      25    254    108    18    39    20
        1042     89      2274      16    254    119    18    30    19
        1043     90      2277      13    254     95    18    42    23
        1044     89      2262      28    254     71    18    60    31
        1045     89      2268      22    254     71    18    90    31
        1046     90      2289       1    254     87    18    72    26
        1047     69      1752     538    254     80    18    42    21



 I//////////////////////////////////////////////////////////////////I
 II                                                                II
 II            DATA STORAGE DETAIL REPORT FOR FILE 1               II
 II                                                                II
 I//////////////////////////////////////////////////////////////////I

 Device Type ...........  3390
 Block Size ............. 5064
 DATA Padding Factor .... 10% (506 bytes)

1File 1                                                      Page 1
 ------------------------------------------------------------------------------
 I   RABN   I   %    I Block  I Free space I Num I  Spanned  I Record lengths  I
 I          I  Used  I Length I-----------I Rec I-----------I-----------------I
 I          I        I        I Gap I Pad  I     I Num I Flg I Min I Max I Avg I
 I----------I--------I--------I-----I------I-----I-----I-----I-----I-----I-----I
          6     1         35   4523   506      1    0           31    31    31
         22    91       4613      0   451     28    0          136   196   164
         23    88       4475     83   506     26    0          143   204   171
         24    87       4382    176   506     25    0          153   205   175
```

```
          25    89       4528     30   506     27    0          143   193   167
          46    89       4506     52   506     24    0          154   252   187
          47    92       4642      0   422     24    0          135   233   193
          48    90       4537     21   506     24    0          136   223   188
          49    89       4513     45   506     24    0          133   253   187
          50    88       4469     89   506     23    0          166   234   194
          51    93       4713      0   351     23    0          157   233   204
```

Sample 4: FILE Parameter Processing

This sample shows FILE parameter processing by ADADRU SPACEUSAGE. All repeated files listed in the FILE parameter are processed only once. ADADRU processesfrom parameter FILE will be processed just once. ADADRU processes files only after all parameters are consolidated.

```
 ADADRU SPACEUSAGE FILE=10-15,9-12,11,11,15



1

 DRU203I, ******* SPACEUSAGE-5 started *******     2012-05-11 12:37:53



 File group number 1.  FILE=10-15,9-12,11,11,15
 -----------------

 DRU200I, The input parameters accepted.


1

1
 ************************************************************************
 ************************************************************************
 ************************************************************************
 ****                                                                ****
 ****             F I L E      9 ( BASE-FILE1       )               ****
 ****                                                                ****
 ****                        2012-05-11 12:37:53                     ****
 ************************************************************************
 ************************************************************************
 ************************************************************************


 Consolidated input parameters:
 ------------------------------
  File number = 9
            ERRLIMIT      = 1
            LOB-GROUP     = No
            EXP-GROUP     = No
```

```
      ASSO  Report type   = SUMMARY
            SAMPLING      = 100    (10 of every 10 records read)
            INCRUSAGE     = 5
            INCRNUMREC    = Auto
            FROM/TO RABN  = 2436         - 2436
            FROM/TO RABN  = 2449         - 2454
            FROM/TO RABN  = 2461         - 2466
      DATA  Report type   = SUMMARY
            SAMPLING      = 100    (10 of every 10 records read)
            INCRUSAGE     = 5
            INCRNUMREC    = Auto
            INCRRECLEN    = Auto
            FROM/TO RABN  = 5502         - 5507

1

 File options (from FCB):
 ------------------------
 Date / Time Loaded ..................... 2010-07-30 10:07:59.03
 Date / Time of last update ............. 2010-08-03 07:04:22.85
 Date / Time of last physical restructure 2010-07-30 10:07:59.03

 ISN Reusage ............................ No
 DATA Storage Reusage ................... No
 MIXDSDEV ............................... No
 Spanned Records Support ................ Yes
 Index compressed ....................... Yes
 2-Byte MU/PE Index ..................... No
 Highest Index Level .................... 3

 LOB fields ............................. Yes. Base file. LOB file = 8
 LOB file ............................... No
 Expanded ............................... No

1
 I/////////////////////////////////////////////////////////////////I
 II                                                               II
 II     ASSOCIATOR SUMMARY REPORT OF NORMAL INDEX FOR FILE 9      II
 II                                                               II
 I/////////////////////////////////////////////////////////////////I

 Physical extents list
 ---------------------------------------------------------------------
 I Extent I Type I          From           To  I % unused I Device I
 I        I      I          RABN          RABN I  blocks  I  type  I
 I--------I------I-----------------------------I----------I--------I
 I     1  I  NI  I          2436          2436 I      0   I  3390  I
 I     2  I  NI  I          2449          2454 I      0   I  3390  I
 I     3  I  NI  I          2461          2466 I      0   I  3390  I
 I        I      I                             I          I        I
 ---------------------------------------------------------------------
```

```
Range of RABNs being processed according
to consolidated input parameters
(unused RABNs at the end of extents excluded)
-----------------------------------------------
I Extent I Type I          From           To  I
I        I      I          RABN          RABN I
I--------I------I-----------------------------I
I     1  I  NI  I          2436          2436 I
I     2  I  NI  I          2449          2454 I
I     3  I  NI  I          2461          2466 I
I        I      I                            I
-----------------------------------------------

File ......................... 9
Device Type .................. 3390
Block Size ................... 2544
ASSO Padding factor .......... 50%   (1272 bytes)


Number of allocated blocks ..............................  13
Number of used blocks ...................................  13
Number of blocks read (SAMPLING = 100%) .................  13
Number of unused blocks at the end of extents ............  0
Number of unused blocks within the used extents ..........  0
Percent of blocks with free space less than padding ......  31
Average percent of block filling (unused blocks excluded)   26
Number of elements in blocks read .......................  425
Average number of elements per block ....................  33
Average element length ..................................  20
Minimum element length ..................................  13
Maximum element length ..................................  256
Number of elements with minimum length ..................  396
Number of elements with maximum length ..................  8

1

I================================================I
II                                              II
II    DISTRIBUTION REPORTS ASSO NORMAL INDEX    II
II                   File = 9                   II
II             Device type = 3390               II
I================================================I

Block filling level; Increment = 5%
-------------------------------------------------
I      Range of       I    Number of blocks      I
I   percent filled    I  filled in this range    I
I--------------------I--------------------------I
I       1 -      5    I                1         I
I      11 -     15    I                4         I
I      16 -     20    I                4         I
I      46 -     50    I                2         I
```

```
I      51 -     55    I                2         I
-------------------------------------------------


-------------------------------------------------
I   Percent ranges    I  Number of blocks with   I
I   of free space     I free space in this range I
I---------------------I--------------------------I
I      46 -     50    I                4         I
I      81 -     85    I                4         I
I      86 -     90    I                4         I
I      96 -    100    I                1         I
-------------------------------------------------

1
 Number of elements per block; Increment = Auto
-------------------------------------------------
I  Number of elements I          Number          I
I     per block       I        of blocks         I
I---------------------I--------------------------I
I       1 -      3    I                9         I
I      97 -     99    I                2         I
I     100 -    102    I                2         I
-------------------------------------------------

1
I/////////////////////////////////////////////////////////////////I
II                                                               II
II          DATA STORAGE SUMMARY REPORT FOR FILE 9               II
II                                                               II
I/////////////////////////////////////////////////////////////////I

 Physical extents list
-----------------------------------------------------------------------
I Extent I Type I          From           To  I % unused I Device I
I        I      I          RABN          RABN I  blocks  I  type  I
I--------I------I-----------------------------I----------I--------I
I     1  I  DS  I          5502          7001 I     99   I  3390  I
I        I      I                             I          I        I
-----------------------------------------------------------------------

 Range of RABNs being processed according
 to consolidated input parameters
 (unused RABNs at the end of extents excluded)
-------------------------------------------------
I Extent I Type I          From           To  I
I        I      I          RABN          RABN I
I--------I------I-----------------------------I
I     1  I  DS  I          5502          5507 I
I        I      I                             I
-------------------------------------------------

 File ......................... 9
```

```
Device Type ................. 3390
Block Size................... 5064
DATA Storage Padding factor .. 0%    (0 bytes)


Number of allocated blocks ..............................  1500
Number of used blocks ...................................  6
Number of blocks read (SAMPLING = 100%) .................  6
Number of unused blocks at the end of extents ...........  1494
Number of unused blocks within the used extents .........  0
Percent of blocks with free space less than padding ......  0
Average percent of block filling (unused blocks excluded)   78
Number of records in blocks read ........................  3
   Number of Normal records .............................  0
   Number of Spanned records ............................  3
Number of blocks with Spanned records ...................  6
Average number of records per block .....................  1

Normal records:
   Average record length ................................  0
   Minimum record length ................................  32767
   Maximum record length ................................  0
   Number of records with minimum length ................  0
   Number of records with maximum length ................  0

Spanned Record:
   Average record length ................................  7850
   Minimum record length ................................  7850
   Maximum record length ................................  7850
   Number of records with minimum length ................  3
   Number of records with maximum length ................  3
   Average numbers of segments ..........................  2
   Minimum numbers of segments ..........................  2
   Maximum numbers of segments ..........................  2
   Number of Primary Spanned Records ....................  3
   Number of Secondary Spanned Records...................  3
   Number of AC2 blocks read ............................  0
   Data Storage blocks with secondary segments read ......  3

1

I================================================I
II                                              II
II       DISTRIBUTION REPORTS DATA STORAGE      II
II                   File = 9                   II
II             Device type = 3390               II
I================================================I

Block filling level; Increment = 5%
-------------------------------------------------
I      Range of       I    Number of blocks      I
I   percent filled    I  filled in this range    I
```

```
 I---------------------I--------------------------I
 I      71 -     75    I                3         I
 I      81 -     85    I                3         I
 ------------------------------------------------


 ------------------------------------------------
 I   Percent ranges    I  Number of blocks with   I
 I   of free space     I free space in this range I
 I---------------------I--------------------------I
 I      16 -     20    I                3         I
 I      26 -     30    I                3         I
 ------------------------------------------------

1
 Number of records per block; Increment = Auto
 ------------------------------------------------
 I  Number of records  I          Number          I
 I     per block       I        of blocks         I
 I---------------------I--------------------------I
 I       1 -      6    I                6         I
 ------------------------------------------------

 DRU163W, WARNING: No data for "Normal records length" report


1
 Spanned records length; Increment = Auto
 ------------------------------------------------
 I        Length       I         Number           I
 I      of record      I       of records         I
 I---------------------I--------------------------I
 I    7827 -   7852    I                3         I
 ------------------------------------------------

1

1
 ************************************************************************
 ************************************************************************
 ************************************************************************
 ****                                                                ****
 ****             F I L E     10 ( EMPLOYEES-EXP1   )                ****
 ****                                                                ****
 ****                       2012-05-11 12:37:53                      ****
 ************************************************************************
 ************************************************************************
 ************************************************************************


 Consolidated input parameters:
 ------------------------------
  File number = 10
```

```
            ERRLIMIT      = 1
            LOB-GROUP     = No
            EXP-GROUP     = No
      ASSO  Report type   = SUMMARY
            SAMPLING      = 100    (10 of every 10 records read)
            INCRUSAGE     = 5
            INCRNUMREC    = Auto
            FROM/TO RABN  = 2164         - 2175
      DATA  Report type   = SUMMARY
            SAMPLING      = 100    (10 of every 10 records read)
            INCRUSAGE     = 5
            INCRNUMREC    = Auto
            INCRRECLEN    = Auto
            FROM/TO RABN  = 5392         - 5395

1

 File options (from FCB):
 ------------------------
 Date / Time Loaded ..................... 2010-07-01 13:39:15.65
 Date / Time of last update ............. n/a
 Date / Time of last physical restructure 2010-07-01 13:39:15.65

 ISN Reusage ............................ No
 DATA Storage Reusage ................... Yes
 MIXDSDEV ............................... No
 Spanned Records Support ................ No
 Index compressed ....................... No
 2-Byte MU/PE Index ..... ..... ..... ... No
 Highest Index Level .................... 3

 LOB fields ............................. No
 LOB file ............................... No
 Expanded ............................... Yes. Seq.number = 2. ANCHOR = 20

1
 I/////////////////////////////////////////////////////////////////I
 II                                                                 II
 II     ASSOCIATOR SUMMARY REPORT OF NORMAL INDEX FOR FILE 10       II
 II                                                                 II
 I/////////////////////////////////////////////////////////////////I

 Physical extents list
 ---------------------------------------------------------------------
 I Extent I Type I          From           To  I % unused I Device I
 I        I      I          RABN          RABN I  blocks  I  type  I
 I--------I------I----------------------------I----------I--------I
 I     1  I  NI  I          2164          2175 I      0   I  3390  I
 I        I      I                            I          I        I
 ---------------------------------------------------------------------

 Range of RABNs being processed according
```

```
 to consolidated input parameters
 (unused RABNs at the end of extents excluded)
 ----------------------------------------------
 I Extent I Type I          From           To  I
 I        I      I          RABN          RABN I
 I--------I------I-----------------------------I
 I     1  I  NI  I          2164          2175 I
 I        I      I                             I
 ----------------------------------------------

 File ......................... 10
 Device Type .................. 3390
 Block Size ................... 2544
 ASSO Padding factor .......... 10%   (254 bytes)


 Number of allocated blocks ..............................  12
 Number of used blocks ...................................  12
 Number of blocks read (SAMPLING = 100%) .................  12
 Number of unused blocks at the end of extents ...........  0
 Number of unused blocks within the used extents .........  0
 Percent of blocks with free space less than padding ......  0
 Average percent of block filling (unused blocks excluded)  37
 Number of elements in blocks read .......................  739
 Average number of elements per block ....................  62
 Average element length ..................................  15
 Minimum element length ..................................  8
 Maximum element length ..................................  302
 Number of elements with minimum length ..................  84
 Number of elements with maximum length ..................  1

1

 I================================================I
 II                                              II
 II    DISTRIBUTION REPORTS ASSO NORMAL INDEX    II
 II                   File = 10                  II
 II             Device type = 3390               II
 I================================================I

 Block filling level; Increment = 5%
 ------------------------------------------------
 I      Range of       I    Number of blocks      I
 I   percent filled    I  filled in this range    I
 I--------------------I--------------------------I
 I      11 -     15    I                2         I
 I      16 -     20    I                1         I
 I      21 -     25    I                1         I
 I      26 -     30    I                1         I
 I      31 -     35    I                1         I
 I      36 -     40    I                1         I
 I      41 -     45    I                2         I
```

```
 I      51 -     55    I                1         I
 I      61 -     65    I                1         I
 I      66 -     70    I                1         I
 ------------------------------------------------


 ------------------------------------------------
 I   Percent ranges    I  Number of blocks with   I
 I   of free space     I free space in this range I
 I---------------------I--------------------------I
 I      26 -     30    I                1         I
 I      36 -     40    I                1         I
 I      46 -     50    I                1         I
 I      51 -     55    I                1         I
 I      56 -     60    I                1         I
 I      61 -     65    I                2         I
 I      66 -     70    I                1         I
 I      71 -     75    I                1         I
 I      81 -     85    I                1         I
 I      86 -     90    I                2         I
 ------------------------------------------------

1
 Number of elements per block; Increment = Auto
 ------------------------------------------------
 I  Number of elements I          Number          I
 I     per block       I        of blocks         I
 I---------------------I--------------------------I
 I       1 -      3    I                1         I
 I       4 -      6    I                1         I
 I       7 -      9    I                1         I
 I      16 -     18    I                1         I
 I      55 -     57    I                1         I
 I      76 -     78    I                1         I
 I      85 -     87    I                1         I
 I      88 -     90    I                1         I
 I      94 -     96    I                1         I
 I     100 -    102    I                3         I
 ------------------------------------------------

1
 I/////////////////////////////////////////////////////////////////I
 II                                                               II
 II          DATA STORAGE SUMMARY REPORT FOR FILE 10              II
 II                                                               II
 I/////////////////////////////////////////////////////////////////I

 Physical extents list
 ---------------------------------------------------------------------
 I Extent I Type I          From           To  I % unused I Device I
 I        I      I          RABN          RABN I  blocks  I  type  I
 I--------I------I-----------------------------I----------I--------I
 I     1  I  DS  I          5392          5401 I     60   I  3390  I
```

```
I        I      I                                I          I        I
-------------------------------------------------------------------

Range of RABNs being processed according
to consolidated input parameters
(unused RABNs at the end of extents excluded)
-----------------------------------------------
I Extent I Type I          From           To  I
I        I      I          RABN          RABN I
I--------I------I-----------------------------I
I     1  I  DS  I          5392          5395 I
I        I      I                               I
-----------------------------------------------

File ......................... 10
Device Type .................. 3390
Block Size.................... 5064
DATA Storage Padding factor .. 10%   (506 bytes)


Number of allocated blocks ..............................  10
Number of used blocks ...................................  4
Number of blocks read (SAMPLING = 100%) .................  4
Number of unused blocks at the end of extents ............  6
Number of unused blocks within the used extents ..........  0
Percent of blocks with free space less than padding ......  25
Average percent of block filling (unused blocks excluded)   77
Number of records in blocks read ........................  100
   Number of Normal records .............................  100
   Number of Spanned records ............................  n/a
Number of blocks with Spanned records ...................  n/a
Average number of records per block .....................  25

Normal records:
   Average record length ................................  155
   Minimum record length ................................  138
   Maximum record length ................................  191
   Number of records with minimum length ................  1
   Number of records with maximum length ................  1

1

I================================================I
II                                              II
II       DISTRIBUTION REPORTS DATA STORAGE      II
II                   File = 10                  II
II             Device type = 3390               II
I================================================I

Block filling level; Increment = 5%
-------------------------------------------------
I      Range of       I    Number of blocks      I
```

```
 I   percent filled    I  filled in this range    I
 I---------------------I--------------------------I
 I      36 -     40    I                1         I
 I      86 -     90    I                2         I
 I      91 -     95    I                1         I
 ------------------------------------------------


 ------------------------------------------------
 I   Percent ranges    I  Number of blocks with   I
 I   of free space     I free space in this range I
 I---------------------I--------------------------I
 I       6 -     10    I                1         I
 I      11 -     15    I                2         I
 I      61 -     65    I                1         I
 ------------------------------------------------

1
 Number of records per block; Increment = Auto
 ------------------------------------------------
 I  Number of records  I          Number          I
 I     per block       I        of blocks         I
 I---------------------I--------------------------I
 I       7 -     12    I                1         I
 I      25 -     30    I                3         I
 ------------------------------------------------

1
 Normal records length; Increment = Auto
 ------------------------------------------------
 I        Length       I          Number          I
 I      of record      I        of records        I
 I---------------------I--------------------------I
 I     133 -    138    I                1         I
 I     139 -    144    I                6         I
 I     145 -    150    I               28         I
 I     151 -    156    I               29         I
 I     157 -    162    I               16         I
 I     163 -    168    I               15         I
 I     169 -    174    I                3         I
 I     175 -    180    I                1         I
 I     187 -    192    I                1         I
 ------------------------------------------------

1

1
 ************************************************************************
 ************************************************************************
 ************************************************************************
 ****                                                                ****
 ****             F I L E     11 ( EMPTY-FILE       )              ****
 ****                                                                ****
```

```
****                        2012-05-11 12:37:53                         ****
************************************************************************
************************************************************************
************************************************************************


 Consolidated input parameters:
 ------------------------------
  File number = 11
            ERRLIMIT      = 1
            LOB-GROUP     = No
            EXP-GROUP     = No
       ASSO  Report type   = SUMMARY
            SAMPLING      = 100    (10 of every 10 records read)
            INCRUSAGE     = 5
            INCRNUMREC    = Auto
            FROM/TO RABN  = Unused area of file extents
       DATA  Report type   = SUMMARY
            SAMPLING      = 100    (10 of every 10 records read)
            INCRUSAGE     = 5
            INCRNUMREC    = Auto
            INCRRECLEN    = Auto
            FROM/TO RABN  = Unused area of file extents

1

 File options (from FCB):
 ------------------------
 Date / Time Loaded ..................... 2010-08-11 12:12:18.04
 Date / Time of last update ............. n/a
 Date / Time of last physical restructure 2010-08-11 12:12:18.04

 ISN Reusage ............................ No
 DATA Storage Reusage ................... Yes
 MIXDSDEV ............................... No
 Spanned Records Support ................ No
 Index compressed ....................... No
 2-Byte MU/PE Index ..................... No
 Highest Index Level .................... 3

 LOB fields ............................. No
 LOB file ............................... No
 Expanded ............................... No

1
 I/////////////////////////////////////////////////////////////////I
 II                                                                 II
 II     ASSOCIATOR SUMMARY REPORT OF NORMAL INDEX FOR FILE 11       II
 II                                                                 II
 I/////////////////////////////////////////////////////////////////I

 Physical extents list
```

```
 ------------------------------------------------------------------------
 I Extent I Type I          From           To  I % unused I Device I
 I        I      I          RABN          RABN I  blocks  I  type  I
 I--------I------I-----------------------------I----------I--------I
 I     1  I  NI  I          2474          2486 I    100   I  3390  I
 I        I      I                               I          I        I
 ------------------------------------------------------------------------

 Range of RABNs being processed according
 to consolidated input parameters
 (unused RABNs at the end of extents excluded)
 ------------------------------------------------
 I Extent I Type I          From           To  I
 I        I      I          RABN          RABN I
 I--------I------I-----------------------------I

 File ......................... 11
 Device Type .................. 3390
 Block Size ................... 2544
 ASSO Padding factor .......... 10%   (254 bytes)


 Number of allocated blocks ..............................  13
 Number of used blocks ...................................  0
 Number of blocks read (SAMPLING = 100%) .................  0
 Number of unused blocks at the end of extents ...........  13
 Number of unused blocks within the used extents .........  0
 Percent of blocks with free space less than padding ......  0
 Average percent of block filling (unused blocks excluded)   0
 Number of elements in blocks read .......................  0
 Average number of elements per block ....................  0
 Average element length ..................................  0
 Minimum element length ..................................  0
 Maximum element length ..................................  0
 Number of elements with minimum length ..................  0
 Number of elements with maximum length ..................  0

1

 I================================================I
 II                                              II
 II    DISTRIBUTION REPORTS ASSO NORMAL INDEX    II
 II                   File = 11                  II
 II             Device type = 3390               II
 I================================================I

 DRU163W, WARNING: No data for "Block filling level" report (Part. 1)


 DRU163W, WARNING: No data for "Block filling level" report (Part. 2)
```

```
 DRU163W, WARNING: No data for "Number of elements per block" report


1
 I////////////////////////////////////////////////////////////////I
 II                                                              II
 II          DATA STORAGE SUMMARY REPORT FOR FILE 11             II
 II                                                              II
 I////////////////////////////////////////////////////////////////I

 Physical extents list
 ------------------------------------------------------------------
 I Extent I Type I          From           To  I % unused I Device I
 I        I      I          RABN          RABN I  blocks  I  type  I
 I--------I------I-----------------------------I----------I--------I
 I     1  I  DS  I          8502          8511 I    100   I  3390  I
 I        I      I                               I          I        I
 ------------------------------------------------------------------

 Range of RABNs being processed according
 to consolidated input parameters
 (unused RABNs at the end of extents excluded)
 ----------------------------------------------
 I Extent I Type I          From           To  I
 I        I      I          RABN          RABN I
 I--------I------I-----------------------------I

 File ......................... 11
 Device Type .................. 3390
 Block Size.................... 5064
 DATA Storage Padding factor .. 10%   (506 bytes)


 Number of allocated blocks ..............................  10
 Number of used blocks ...................................  0
 Number of blocks read (SAMPLING = 100%) .................  0
 Number of unused blocks at the end of extents ...........  10
 Number of unused blocks within the used extents .........  0
 Percent of blocks with free space less than padding ......  0
 Average percent of block filling (unused blocks excluded)   0
 Number of records in blocks read ........................  0
    Number of Normal records .............................  0
    Number of Spanned records ............................  n/a
 Number of blocks with Spanned records ...................  n/a
 Average number of records per block .....................  0

 Normal records:
    Average record length ................................  0
    Minimum record length ................................  0
    Maximum record length ................................  0
    Number of records with minimum length ................  0
    Number of records with maximum length ................  0
```

```
1

 I================================================I
 II                                              II
 II       DISTRIBUTION REPORTS DATA STORAGE      II
 II                   File = 11                  II
 II             Device type = 3390               II
 I================================================I

 DRU163W, WARNING: No data for "Block filling level" report (Part. 1)


 DRU163W, WARNING: No data for "Block filling level" report (Part. 2)


 DRU163W, WARNING: No data for "Number of records per block" report


 DRU163W, WARNING: No data for "Normal records length" report


1

1
 ************************************************************************
 ************************************************************************
 ************************************************************************
 ****                                                                ****
 ****             F I L E     12 ( SECURITY         )                ****
 ****                                                                ****
 ****                        2012-05-11 12:37:53                     ****
 ************************************************************************
 ************************************************************************
 ************************************************************************


 Consolidated input parameters:
 ------------------------------
  File number = 12
            ERRLIMIT      = 1
            LOB-GROUP     = No
            EXP-GROUP     = No
      ASSO  Report type   = SUMMARY
            SAMPLING      = 100    (10 of every 10 records read)
            INCRUSAGE     = 5
            INCRNUMREC    = Auto
            FROM/TO RABN  = Unused area of file extents
      DATA  Report type   = SUMMARY
            SAMPLING      = 100    (10 of every 10 records read)
            INCRUSAGE     = 5
            INCRNUMREC    = Auto
```

```
            INCRRECLEN    = Auto
            FROM/TO RABN  = Unused area of file extents

1

 File options (from FCB):
 ------------------------
 Date / Time Loaded ..................... 2010-11-10 10:04:39.30
 Date / Time of last update ............. n/a
 Date / Time of last physical restructure 2010-11-10 10:04:39.30

 ISN Reusage ............................ No
 DATA Storage Reusage ................... Yes
 MIXDSDEV ............................... No
 Spanned Records Support ................ No
 Index compressed ....................... No
 2-Byte MU/PE Index ..................... No
 Highest Index Level .................... 3

 LOB fields ............................. No
 LOB file ............................... No
 Expanded ............................... No

1
 I/////////////////////////////////////////////////////////////////I
 II                                                               II
 II     ASSOCIATOR SUMMARY REPORT OF NORMAL INDEX FOR FILE 12     II
 II                                                               II
 I/////////////////////////////////////////////////////////////////I

 Physical extents list
 ---------------------------------------------------------------------
 I Extent I Type I          From           To  I % unused I Device I
 I        I      I          RABN          RABN I  blocks  I  type  I
 I--------I------I----------------------------I----------I--------I
 I     1  I  NI  I          2493          2512 I    100   I  3390  I
 I        I      I                            I          I        I
 ---------------------------------------------------------------------

 Range of RABNs being processed according
 to consolidated input parameters
 (unused RABNs at the end of extents excluded)
 ------------------------------------------------
 I Extent I Type I          From           To  I
 I        I      I          RABN          RABN I
 I--------I------I----------------------------I

 File ......................... 12
 Device Type .................. 3390
 Block Size ................... 2544
 ASSO Padding factor .......... 10%   (254 bytes)
```

```
 Number of allocated blocks ..............................  20
 Number of used blocks ...................................  0
 Number of blocks read (SAMPLING = 100%) .................  0
 Number of unused blocks at the end of extents ...........  20
 Number of unused blocks within the used extents .........  0
 Percent of blocks with free space less than padding ......  0
 Average percent of block filling (unused blocks excluded)  0
 Number of elements in blocks read .......................  0
 Average number of elements per block ....................  0
 Average element length ..................................  0
 Minimum element length ..................................  0
 Maximum element length ..................................  0
 Number of elements with minimum length ..................  0
 Number of elements with maximum length ..................  0

1

 I================================================I
 II                                              II
 II    DISTRIBUTION REPORTS ASSO NORMAL INDEX    II
 II                   File = 12                  II
 II             Device type = 3390               II
 I================================================I

 DRU163W, WARNING: No data for "Block filling level" report (Part. 1)


 DRU163W, WARNING: No data for "Block filling level" report (Part. 2)


 DRU163W, WARNING: No data for "Number of elements per block" report


1
 I//////////////////////////////////////////////////////////////I
 II                                                            II
 II          DATA STORAGE SUMMARY REPORT FOR FILE 12           II
 II                                                            II
 I//////////////////////////////////////////////////////////////I

 Physical extents list
 ------------------------------------------------------------------
 I Extent I Type I          From           To  I % unused I Device I
 I        I      I          RABN          RABN I  blocks  I  type  I
 I--------I------I-----------------------------I----------I--------I
 I     1  I  DS  I          8512          8531 I    100   I  3390  I
 I        I      I                               I          I        I
 ------------------------------------------------------------------

 Range of RABNs being processed according
 to consolidated input parameters
```

```
 (unused RABNs at the end of extents excluded)
 ---------------------------------------------
 I Extent I Type I          From           To  I
 I        I      I          RABN          RABN I
 I--------I------I-----------------------------I

 File ......................... 12
 Device Type .................. 3390
 Block Size.................... 5064
 DATA Storage Padding factor .. 10%   (506 bytes)


 Number of allocated blocks ..............................  20
 Number of used blocks ...................................  0
 Number of blocks read (SAMPLING = 100%) .................  0
 Number of unused blocks at the end of extents ...........  20
 Number of unused blocks within the used extents .........  0
 Percent of blocks with free space less than padding ......  0
 Average percent of block filling (unused blocks excluded)   0
 Number of records in blocks read ........................  0
    Number of Normal records .............................  0
    Number of Spanned records ............................  n/a
 Number of blocks with Spanned records ...................  n/a
 Average number of records per block .....................  0

 Normal records:
    Average record length ................................  0
    Minimum record length ................................  0
    Maximum record length ................................  0
    Number of records with minimum length ................  0
    Number of records with maximum length ................  0

1

 I================================================I
 II                                              II
 II       DISTRIBUTION REPORTS DATA STORAGE      II
 II                   File = 12                  II
 II             Device type = 3390               II
 I================================================I

 DRU163W, WARNING: No data for "Block filling level" report (Part. 1)


 DRU163W, WARNING: No data for "Block filling level" report (Part. 2)


 DRU163W, WARNING: No data for "Number of records per block" report


 DRU163W, WARNING: No data for "Normal records length" report
```

```
1

1
 ************************************************************************
 ************************************************************************
 ************************************************************************
 ****                                                                ****
 ****             F I L E     13 ( TESTFILE         )                ****
 ****                                                                ****
 ****                        2012-05-11 12:37:53                     ****
 ************************************************************************
 ************************************************************************
 ************************************************************************


 Consolidated input parameters:
 ------------------------------
  File number = 13
            ERRLIMIT      = 1
            LOB-GROUP     = No
            EXP-GROUP     = No
      ASSO  Report type   = SUMMARY
            SAMPLING      = 100    (10 of every 10 records read)
            INCRUSAGE     = 5
            INCRNUMREC    = Auto
            FROM/TO RABN  = Unused area of file extents
      DATA  Report type   = SUMMARY
            SAMPLING      = 100    (10 of every 10 records read)
            INCRUSAGE     = 5
            INCRNUMREC    = Auto
            INCRRECLEN    = Auto
            FROM/TO RABN  = Unused area of file extents

1

 File options (from FCB):
 ------------------------
 Date / Time Loaded ..................... 2011-02-21 17:30:26.92
 Date / Time of last update ............. n/a
 Date / Time of last physical restructure 2011-02-21 17:30:26.92

 ISN Reusage ............................ No
 DATA Storage Reusage ................... Yes
 MIXDSDEV ............................... No
 Spanned Records Support ................ Yes
 Index compressed ....................... Yes
 2-Byte MU/PE Index ..................... No
 Highest Index Level .................... 3

 LOB fields ............................. No
 LOB file ............................... No
```

```
 Expanded ............................... No

1
 I//////////////////////////////////////////////////////////////I
 II                                                            II
 II     ASSOCIATOR SUMMARY REPORT OF NORMAL INDEX FOR FILE 13     II
 II                                                            II
 I//////////////////////////////////////////////////////////////I

 Physical extents list
 ------------------------------------------------------------------
 I Extent I Type I          From           To  I % unused I Device I
 I        I      I          RABN          RABN I  blocks  I  type  I
 I--------I------I-----------------------------I----------I--------I
 I     1  I  NI  I          3178          3197 I    100   I  3390  I
 I        I      I                               I          I        I
 ------------------------------------------------------------------

 Range of RABNs being processed according
 to consolidated input parameters
 (unused RABNs at the end of extents excluded)
 ------------------------------------------------
 I Extent I Type I          From           To  I
 I        I      I          RABN          RABN I
 I--------I------I-----------------------------I

 File ......................... 13
 Device Type .................. 3390
 Block Size ................... 2544
 ASSO Padding factor .......... 10%   (254 bytes)


 Number of allocated blocks ..............................  20
 Number of used blocks ...................................  0
 Number of blocks read (SAMPLING = 100%) .................  0
 Number of unused blocks at the end of extents ............  20
 Number of unused blocks within the used extents ..........  0
 Percent of blocks with free space less than padding ......  0
 Average percent of block filling (unused blocks excluded)   0
 Number of elements in blocks read .......................  0
 Average number of elements per block ....................  0
 Average element length ..................................  0
 Minimum element length ..................................  0
 Maximum element length ..................................  0
 Number of elements with minimum length ..................  0
 Number of elements with maximum length ..................  0

1

 I================================================I
 II                                              II
 II    DISTRIBUTION REPORTS ASSO NORMAL INDEX    II
```

```
 II                   File = 13                  II
 II             Device type = 3390               II
 I================================================I

 DRU163W, WARNING: No data for "Block filling level" report (Part. 1)


 DRU163W, WARNING: No data for "Block filling level" report (Part. 2)


 DRU163W, WARNING: No data for "Number of elements per block" report


1
 I//////////////////////////////////////////////////////////////I
 II                                                            II
 II          DATA STORAGE SUMMARY REPORT FOR FILE 13           II
 II                                                            II
 I//////////////////////////////////////////////////////////////I

 Physical extents list
 ------------------------------------------------------------------
 I Extent I Type I          From           To  I % unused I Device I
 I        I      I          RABN          RABN I  blocks  I  type  I
 I--------I------I----------------------------I----------I--------I
 I     1  I  DS  I          8592          8611 I    100   I  3390  I
 I        I      I                            I          I        I
 ------------------------------------------------------------------

 Range of RABNs being processed according
 to consolidated input parameters
 (unused RABNs at the end of extents excluded)
 -------------------------------------------------
 I Extent I Type I          From           To  I
 I        I      I          RABN          RABN I
 I--------I------I----------------------------I

 File ......................... 13
 Device Type .................. 3390
 Block Size.................... 5064
 DATA Storage Padding factor .. 0%    (0 bytes)


 Number of allocated blocks ..............................  20
 Number of used blocks ...................................  0
 Number of blocks read (SAMPLING = 100%) .................  0
 Number of unused blocks at the end of extents ............  20
 Number of unused blocks within the used extents ..........  0
 Percent of blocks with free space less than padding ......  0
 Average percent of block filling (unused blocks excluded)   0
 Number of records in blocks read ........................  0
    Number of Normal records .............................  0
```

```
    Number of Spanned records ............................  0
 Number of blocks with Spanned records ...................  0
 Average number of records per block .....................  0

 Normal records:
    Average record length ................................  0
    Minimum record length ................................  0
    Maximum record length ................................  0
    Number of records with minimum length ................  0
    Number of records with maximum length ................  0

 Spanned Record:
    Average record length ................................  0
    Minimum record length ................................  0
    Maximum record length ................................  0
    Number of records with minimum length ................  0
    Number of records with maximum length ................  0
    Average numbers of segments ..........................  0
    Minimum numbers of segments ..........................  0
    Maximum numbers of segments ..........................  0
    Number of Primary Spanned Records ....................  0
    Number of Secondary Spanned Records...................  0
    Number of AC2 blocks read ............................  0
    Data Storage blocks with secondary segments read ......  0

1

 I=============================================I
 II                                           II
 II       DISTRIBUTION REPORTS DATA STORAGE      II
 II                    File = 13                II
 II             Device type = 3390              II
 I=============================================I

 DRU163W, WARNING: No data for "Block filling level" report (Part. 1)


 DRU163W, WARNING: No data for "Block filling level" report (Part. 2)


 DRU163W, WARNING: No data for "Number of records per block" report


 DRU163W, WARNING: No data for "Normal records length" report


 DRU163W, WARNING: No data for "Spanned records length" report


1

1
```

```
************************************************************************
************************************************************************
************************************************************************
****                                                                ****
****             F I L E     14 ( PEGROUP          )                ****
****                                                                ****
****                        2012-05-11 12:37:53                     ****
************************************************************************
************************************************************************
************************************************************************


Consolidated input parameters:
------------------------------
 File number = 14
           ERRLIMIT      = 1
           LOB-GROUP     = No
           EXP-GROUP     = No
     ASSO  Report type   = SUMMARY
           SAMPLING      = 100    (10 of every 10 records read)
           INCRUSAGE     = 5
           INCRNUMREC    = Auto
           FROM/TO RABN  = Unused area of file extents
     DATA  Report type   = SUMMARY
           SAMPLING      = 100    (10 of every 10 records read)
           INCRUSAGE     = 5
           INCRNUMREC    = Auto
           INCRRECLEN    = Auto
           FROM/TO RABN  = Unused area of file extents

1

 File options (from FCB):
 ------------------------
 Date / Time Loaded ..................... 2011-03-10 16:19:48.75
 Date / Time of last update ............. n/a
 Date / Time of last physical restructure 2011-03-10 16:19:48.75

 ISN Reusage ............................ No
 DATA Storage Reusage ................... Yes
 MIXDSDEV ............................... No
 Spanned Records Support ................ No
 Index compressed ....................... No
 2-Byte MU/PE Index ..................... No
 Highest Index Level .................... 3

 LOB fields ............................. No
 LOB file ............................... No
 Expanded ............................... No

1
 I///////////////////////////////////////////////////////////////I
```

```
II                                                                  II
II     ASSOCIATOR SUMMARY REPORT OF NORMAL INDEX FOR FILE 14        II
II                                                                  II
I//////////////////////////////////////////////////////////////////I

Physical extents list
-------------------------------------------------------------------
I Extent I Type I          From           To  I % unused I Device I
I        I      I          RABN          RABN I  blocks  I  type  I
I--------I------I----------------------------I----------I--------I
I     1  I  NI  I          3206          3225 I    100   I  3390  I
I        I      I                             I          I        I
-------------------------------------------------------------------

Range of RABNs being processed according
to consolidated input parameters
(unused RABNs at the end of extents excluded)
-----------------------------------------------
I Extent I Type I          From           To  I
I        I      I          RABN          RABN I
I--------I------I----------------------------I

File ......................... 14
Device Type .................. 3390
Block Size ................... 2544
ASSO Padding factor .......... 10%   (254 bytes)


Number of allocated blocks ..............................  20
Number of used blocks ...................................  0
Number of blocks read (SAMPLING = 100%) .................  0
Number of unused blocks at the end of extents ...........  20
Number of unused blocks within the used extents .........  0
Percent of blocks with free space less than padding ......  0
Average percent of block filling (unused blocks excluded)   0
Number of elements in blocks read .......................  0
Average number of elements per block ....................  0
Average element length ..................................  0
Minimum element length ..................................  0
Maximum element length ..................................  0
Number of elements with minimum length ..................  0
Number of elements with maximum length ..................  0

1

I================================================I
II                                              II
II    DISTRIBUTION REPORTS ASSO NORMAL INDEX    II
II                   File = 14                  II
II             Device type = 3390               II
I================================================I
```

```
 DRU163W, WARNING: No data for "Block filling level" report (Part. 1)


 DRU163W, WARNING: No data for "Block filling level" report (Part. 2)


 DRU163W, WARNING: No data for "Number of elements per block" report


1
 I//////////////////////////////////////////////////////////////////I
 II                                                                  II
 II          DATA STORAGE SUMMARY REPORT FOR FILE 14                 II
 II                                                                  II
 I//////////////////////////////////////////////////////////////////I

 Physical extents list
 ---------------------------------------------------------------------
 I Extent I Type I          From           To  I % unused I Device I
 I        I      I          RABN          RABN I  blocks  I  type  I
 I--------I------I-----------------------------I----------I--------I
 I     1  I  DS  I          8612          8631 I    100   I  3390  I
 I        I      I                               I          I        I
 ---------------------------------------------------------------------

 Range of RABNs being processed according
 to consolidated input parameters
 (unused RABNs at the end of extents excluded)
 -----------------------------------------------
 I Extent I Type I          From           To  I
 I        I      I          RABN          RABN I
 I--------I------I-----------------------------I

 File ......................... 14
 Device Type .................. 3390
 Block Size.................... 5064
 DATA Storage Padding factor .. 10%   (506 bytes)


 Number of allocated blocks ..............................  20
 Number of used blocks ...................................  0
 Number of blocks read (SAMPLING = 100%) .................  0
 Number of unused blocks at the end of extents ...........  20
 Number of unused blocks within the used extents .........  0
 Percent of blocks with free space less than padding ......  0
 Average percent of block filling (unused blocks excluded)   0
 Number of records in blocks read ........................  0
    Number of Normal records .............................  0
    Number of Spanned records ............................  n/a
 Number of blocks with Spanned records ...................  n/a
 Average number of records per block .....................  0
```

```
 Normal records:
    Average record length ..............................  0
    Minimum record length ..............................  0
    Maximum record length ..............................  0
    Number of records with minimum length ..............  0
    Number of records with maximum length ..............  0

1

 I================================================I
 II                                              II
 II       DISTRIBUTION REPORTS DATA STORAGE      II
 II                   File = 14                  II
 II             Device type = 3390               II
 I================================================I

 DRU163W, WARNING: No data for "Block filling level" report (Part. 1)


 DRU163W, WARNING: No data for "Block filling level" report (Part. 2)


 DRU163W, WARNING: No data for "Number of records per block" report


 DRU163W, WARNING: No data for "Normal records length" report


1

1
 ************************************************************************
 ************************************************************************
 ************************************************************************
 ****                                                                ****
 ****             F I L E     15 ( TESTFILE         )                ****
 ****                                                                ****
 ****                       2012-05-11 12:37:53                      ****
 ************************************************************************
 ************************************************************************
 ************************************************************************


 Consolidated input parameters:
 ------------------------------
  File number = 15
            ERRLIMIT      = 1
            LOB-GROUP     = No
            EXP-GROUP     = No
      ASSO  Report type   = SUMMARY
            SAMPLING      = 100    (10 of every 10 records read)
            INCRUSAGE     = 5
```

```
            INCRNUMREC    = Auto
            FROM/TO RABN  = Unused area of file extents
      DATA  Report type   = SUMMARY
            SAMPLING      = 100    (10 of every 10 records read)
            INCRUSAGE     = 5
            INCRNUMREC    = Auto
            INCRRECLEN    = Auto
            FROM/TO RABN  = 8572         - 8573

1

 File options (from FCB):
 ------------------------
 Date / Time Loaded ..................... 2011-01-12 11:53:26.64
 Date / Time of last update ............. n/a
 Date / Time of last physical restructure 2011-01-12 11:53:26.64

 ISN Reusage ............................ No
 DATA Storage Reusage ................... Yes
 MIXDSDEV ............................... No
 Spanned Records Support ................ Yes
 Index compressed ....................... No
 2-Byte MU/PE Index ..................... No
 Highest Index Level .................... 0

 LOB fields ............................. No
 LOB file ............................... No
 Expanded ............................... No

1
 I/////////////////////////////////////////////////////////////////I
 II                                                                 II
 II     ASSOCIATOR SUMMARY REPORT OF NORMAL INDEX FOR FILE 15       II
 II                                                                 II
 I/////////////////////////////////////////////////////////////////I

 Physical extents list
 ---------------------------------------------------------------------
 I Extent I Type I          From           To  I % unused I Device I
 I        I      I          RABN          RABN I  blocks  I  type  I
 I--------I------I----------------------------I----------I--------I
 I     1  I  NI  I          3150          3169 I    100   I  3390  I
 I        I      I                            I          I        I
 ---------------------------------------------------------------------

 Range of RABNs being processed according
 to consolidated input parameters
 (unused RABNs at the end of extents excluded)
 -----------------------------------------------
 I Extent I Type I          From           To  I
 I        I      I          RABN          RABN I
 I--------I------I----------------------------I
```

```
 File ......................... 15
 Device Type .................. 3390
 Block Size ................... 2544
 ASSO Padding factor .......... 10%   (254 bytes)


 Number of allocated blocks ..............................  20
 Number of used blocks ...................................  0
 Number of blocks read (SAMPLING = 100%) .................  0
 Number of unused blocks at the end of extents ...........  20
 Number of unused blocks within the used extents .........  0
 Percent of blocks with free space less than padding ......  0
 Average percent of block filling (unused blocks excluded)   0
 Number of elements in blocks read .......................  0
 Average number of elements per block ....................  0
 Average element length ..................................  0
 Minimum element length ..................................  0
 Maximum element length ..................................  0
 Number of elements with minimum length ..................  0
 Number of elements with maximum length ..................  0

1

 I================================================I
 II                                              II
 II    DISTRIBUTION REPORTS ASSO NORMAL INDEX    II
 II                    File = 15                 II
 II             Device type = 3390               II
 I================================================I

 DRU163W, WARNING: No data for "Block filling level" report (Part. 1)


 DRU163W, WARNING: No data for "Block filling level" report (Part. 2)


 DRU163W, WARNING: No data for "Number of elements per block" report


1
 I//////////////////////////////////////////////////////////////I
 II                                                            II
 II          DATA STORAGE SUMMARY REPORT FOR FILE 15           II
 II                                                            II
 I//////////////////////////////////////////////////////////////I

 Physical extents list
 ------------------------------------------------------------------
 I Extent I Type I          From           To  I % unused I Device I
 I        I      I          RABN          RABN I  blocks  I  type  I
 I--------I------I----------------------------I----------I--------I
```

```
I     1  I  DS  I          8572          8591 I     90   I  3390  I
I        I      I                             I          I        I
-------------------------------------------------------------------

Range of RABNs being processed according
to consolidated input parameters
(unused RABNs at the end of extents excluded)
-----------------------------------------------
I Extent I Type I          From           To  I
I        I      I          RABN          RABN I
I--------I------I-----------------------------I
I     1  I  DS  I          8572          8573 I
I        I      I                             I
-----------------------------------------------

File ......................... 15
Device Type .................. 3390
Block Size.................... 5064
DATA Storage Padding factor .. 0%    (0 bytes)


Number of allocated blocks ..............................  20
Number of used blocks ...................................  2
Number of blocks read (SAMPLING = 100%) .................  2
Number of unused blocks at the end of extents ............  18
Number of unused blocks within the used extents ..........  0
Percent of blocks with free space less than padding ......  0
Average percent of block filling (unused blocks excluded)   77
Number of records in blocks read ........................  1
   Number of Normal records .............................  0
   Number of Spanned records ............................  1
Number of blocks with Spanned records ...................  2
Average number of records per block .....................  1

Normal records:
   Average record length ................................  0
   Minimum record length ................................  32767
   Maximum record length ................................  0
   Number of records with minimum length ................  0
   Number of records with maximum length ................  0

Spanned Record:
   Average record length ................................  7818
   Minimum record length ................................  7818
   Maximum record length ................................  7818
   Number of records with minimum length ................  1
   Number of records with maximum length ................  1
   Average numbers of segments ..........................  2
   Minimum numbers of segments ..........................  2
   Maximum numbers of segments ..........................  2
   Number of Primary Spanned Records ....................  1
   Number of Secondary Spanned Records...................  1
```

```
    Number of AC2 blocks read ..............................  1
    Data Storage blocks with secondary segments read .......  1

1

 I================================================I
 II                                              II
 II       DISTRIBUTION REPORTS DATA STORAGE      II
 II                    File = 15                 II
 II             Device type = 3390               II
 I================================================I

 Block filling level; Increment = 5%
 ------------------------------------------------
 I      Range of       I    Number of blocks     I
 I   percent filled    I  filled in this range   I
 I--------------------I-------------------------I
 I      51 -     55    I                1        I
 I      96 -    100    I                1        I
 ------------------------------------------------

 ------------------------------------------------
 I   Percent ranges    I  Number of blocks with  I
 I   of free space     I free space in this range I
 I--------------------I-------------------------I
 I       1 -      5    I                1        I
 I      41 -     45    I                1        I
 ------------------------------------------------

1
 Number of records per block; Increment = Auto
 ------------------------------------------------
 I  Number of records  I          Number         I
 I     per block       I        of blocks        I
 I--------------------I-------------------------I
 I       1 -      6    I                2        I
 ------------------------------------------------

 DRU163W, WARNING: No data for "Normal records length" report


1
 Spanned records length; Increment = Auto
 ------------------------------------------------
 I        Length       I         Number          I
 I      of record      I       of records        I
 I--------------------I-------------------------I
 I    7801 -   7826    I                1        I
 ------------------------------------------------
```

```
1
 ************************************************************************
 ************************************************************************
 ************************************************************************
 ****                                                                ****
 ****       F i l e s   s t a t i s t i c a l   r e p o r t          ****
 ****                                                                ****
 ************************************************************************
 ************************************************************************
 ************************************************************************

 ------------------------------------------------------------------------------
 I F I  Duration I     ASSOCIATOR (NI)    I      DATA STORAGE      I ParmI Run I
 I i IHH:MM:SS.thI-----------------------I-----------------------I-----I-----I
 I l I           I  Number  I E I Extents I  Number  I E I Extents I E W I E W I
 I e I           I of RABNs I r I---------I of RABNs I r I---------I r n I r n I
 I   I           I   read   I r I FCB Run I   read   I r I FCB Run I r g I r g I
 I---I-----------I----------I---I----I----I----------I---I----I----I--I--I--I--I
    9 00:00:00.13         13   0    3    3          6   0    1    1  0  0  0  1
   10 00:00:00.02         12   0    1    1          4   0    1    1  0  0  0  0
   11 00:00:00.00          0   0    0    0          0   0    0    0  0  0  0  7
   12 00:00:00.00          0   0    0    0          0   0    0    0  0  0  0  7
   13 00:00:00.00          0   0    0    0          0   0    0    0  0  0  0  8
   14 00:00:00.00          0   0    0    0          0   0    0    0  0  0  0  7
   15 00:00:00.02          0   0    0    0          2   0    1    1  0  0  0  4



1

 DRU205I, ******* SPACEUSAGE-5 ended *******     CC = 04
```

Sample 5: EXPGROUP Parameter Processing

This sample shows ADADRU SPACEUSAGE processing of the EXPGROUP parameter. The expanded files in this example have the following chain: file 22 (the anchor file), file 20, and file 13.

In the first FILE parameter, the EXPGROUP parameter is omitted. Consequently, the whole chain of expanded files is not processed.

In the second FILE parameter, the EXPGROUP parameter is included. Consequently, ADADRU SPACEUSAGE processes the whole chain of expanded files in logical order (file 22 first, followed by files 20 and 13). After all the file reports are produced, ADADRU produces a summary report of the expanded files.

ADADRU processes files only after all parameters are consolidated.

```
 ADADRU SPACEUSAGE FILE=20,PFILESPACE=DETAIL
 ADADRU            FILE=13,PFILESPACE=DETAIL,EXPGROUP

1

 DRU203I, ******* SPACEUSAGE-1 started *******     2012-05-11 12:48:31



 File group number 1.  FILE=20
 -----------------

 File group number 2.  FILE=13
 -----------------

 DRU200I, The input parameters accepted.


 EXPANDED GROUPS:
 ---------------------------
 I    File    I  Sequence  I
 I            I   number   I
 I------------I------------I
 I        22  I    Anchor  I
 I        20  I         2  I
 I        13  I         3  I
 I            I            I
 ---------------------------

 DRU161W, WARNING: File 13, part 3 of expanded file
          will be processed in chain of all parts of expanded file.
          Anchor File 22.


 DRU161W, WARNING: File 20, part 2 of expanded file
          will be processed in chain of all parts of expanded file.
          Anchor File 22.

1

1
 ************************************************************************
 ************************************************************************
 ************************************************************************
 ****                                                                ****
 ****             F I L E     22 ( TESTFILE         )                ****
 ****                                                                ****
 ****                       2012-05-11 12:48:31                      ****
 ************************************************************************
 ************************************************************************
 ************************************************************************
```

```
 Consolidated input parameters:
 ------------------------------
  File number = 22
            ERRLIMIT      = 1
            LOB-GROUP     = No
            EXP-GROUP     = Yes
       ASSO  Report type   = SUMMARY + DETAIL
            SAMPLING      = 100    (10 of every 10 records read)
            INCRUSAGE     = 5
            INCRNUMREC    = Auto
            FROM/TO RABN  = 2151         - 2164
       DATA  Report type   = SUMMARY + DETAIL
            SAMPLING      = 100    (10 of every 10 records read)
            INCRUSAGE     = 5
            INCRNUMREC    = Auto
            INCRRECLEN    = Auto
            FROM/TO RABN  = 5382         - 5385

1

 File options (from FCB):
 ------------------------
 Date / Time Loaded ..................... 2011-09-27 11:20:10.24
 Date / Time of last update ............. 2011-10-21 12:17:55.72
 Date / Time of last physical restructure 2011-09-27 11:20:10.24

 ISN Reusage ............................ No
 DATA Storage Reusage ................... Yes
 MIXDSDEV ............................... No
 Spanned Records Support ................ No
 Index compressed ....................... No
 2-Byte MU/PE Index ..................... No
 Highest Index Level .................... 3

 LOB fields ............................. No
 LOB file ............................... No
 Expanded ............................... Yes. Seq.number = 1. ANCHOR = 22

1
 I//////////////////////////////////////////////////////////////I
 II                                                              II
 II     ASSOCIATOR SUMMARY REPORT OF NORMAL INDEX FOR FILE 22    II
 II                                                              II
 I//////////////////////////////////////////////////////////////I

 Physical extents list
 ---------------------------------------------------------------
 I Extent I Type I          From           To  I % unused I Device I
 I        I      I          RABN          RABN I  blocks  I  type  I
 I--------I------I----------------------------I----------I--------I
 I     1  I  NI  I          2151          2170 I     30   I  3390  I
```

```
I        I      I                                I          I        I
-------------------------------------------------------------------

Range of RABNs being processed according
to consolidated input parameters
(unused RABNs at the end of extents excluded)
-----------------------------------------------
I Extent I Type I          From           To  I
I        I      I          RABN          RABN I
I--------I------I-----------------------------I
I     1  I  NI  I          2151          2164 I
I        I      I                               I
-----------------------------------------------

File ......................... 22
Device Type .................. 3390
Block Size ................... 2544
ASSO Padding factor .......... 10%   (254 bytes)


Number of allocated blocks ..............................  20
Number of used blocks ...................................  14
Number of blocks read (SAMPLING = 100%) .................  14
Number of unused blocks at the end of extents ............  6
Number of unused blocks within the used extents ..........  0
Percent of blocks with free space less than padding ......  0
Average percent of block filling (unused blocks excluded)   37
Number of elements in blocks read .......................  767
Average number of elements per block ....................  55
Average element length ..................................  17
Minimum element length ..................................  8
Maximum element length ..................................  293
Number of elements with minimum length ..................  29
Number of elements with maximum length ..................  1

1

I================================================I
II                                              II
II    DISTRIBUTION REPORTS ASSO NORMAL INDEX    II
II                   File = 22                  II
II             Device type = 3390               II
I================================================I

Block filling level; Increment = 5%
-----------------------------------------------
I      Range of       I    Number of blocks      I
I   percent filled    I  filled in this range    I
I--------------------I--------------------------I
I      11 -     15    I                2         I
I      16 -     20    I                3         I
I      21 -     25    I                2         I
```

```
 I      26 -     30    I                1         I
 I      31 -     35    I                1         I
 I      41 -     45    I                1         I
 I      46 -     50    I                1         I
 I      66 -     70    I                1         I
 I      86 -     90    I                2         I
 ------------------------------------------------

 ------------------------------------------------
 I   Percent ranges    I  Number of blocks with   I
 I   of free space     I free space in this range I
 I---------------------I--------------------------I
 I       6 -     10    I                1         I
 I      11 -     15    I                1         I
 I      31 -     35    I                1         I
 I      46 -     50    I                1         I
 I      56 -     60    I                1         I
 I      66 -     70    I                1         I
 I      71 -     75    I                1         I
 I      76 -     80    I                2         I
 I      81 -     85    I                3         I
 I      86 -     90    I                2         I
 ------------------------------------------------

1
 Number of elements per block; Increment = Auto
 ------------------------------------------------
 I  Number of elements I          Number          I
 I     per block       I        of blocks         I
 I---------------------I--------------------------I
 I       4 -      6    I                2         I
 I       7 -      9    I                1         I
 I      22 -     24    I                2         I
 I      40 -     42    I                2         I
 I      46 -     48    I                2         I
 I      91 -     93    I                1         I
 I      94 -     96    I                1         I
 I      97 -     99    I                1         I
 I     115 -    117    I                1         I
 I     121 -    123    I                1         I
 ------------------------------------------------

1
 I////////////////////////////////////////////////////////////////I
 II                                                              II
 II          DATA STORAGE SUMMARY REPORT FOR FILE 22             II
 II                                                              II
 I////////////////////////////////////////////////////////////////I

 Physical extents list
 ------------------------------------------------------------------
 I Extent I Type I          From           To  I % unused I Device I
```

```
I        I      I          RABN          RABN I  blocks  I  type  I
I--------I------I-----------------------------I----------I--------I
I     1  I  DS  I          5382          5401 I     80   I  3390  I
I        I      I                               I          I        I
-------------------------------------------------------------------

Range of RABNs being processed according
to consolidated input parameters
(unused RABNs at the end of extents excluded)
-----------------------------------------------
I Extent I Type I          From           To  I
I        I      I          RABN          RABN I
I--------I------I-----------------------------I
I     1  I  DS  I          5382          5385 I
I        I      I                               I
-----------------------------------------------

File ......................... 22
Device Type .................. 3390
Block Size.................... 5064
DATA Storage Padding factor .. 10%   (506 bytes)


Number of allocated blocks ..............................  20
Number of used blocks ...................................  4
Number of blocks read (SAMPLING = 100%) .................  4
Number of unused blocks at the end of extents ...........  16
Number of unused blocks within the used extents .........  0
Percent of blocks with free space less than padding ......  25
Average percent of block filling (unused blocks excluded)   82
Number of records in blocks read ........................  98
   Number of Normal records .............................  98
   Number of Spanned records ............................  n/a
Number of blocks with Spanned records ...................  n/a
Average number of records per block .....................  25

Normal records:
   Average record length ................................  168
   Minimum record length ................................  126
   Maximum record length ................................  208
   Number of records with minimum length ................  1
   Number of records with maximum length ................  1

1

I================================================I
II                                              II
II       DISTRIBUTION REPORTS DATA STORAGE      II
II                    File = 22                 II
II             Device type = 3390               II
I================================================I
```

```
 Block filling level; Increment = 5%
 ------------------------------------------------
 I      Range of       I    Number of blocks      I
 I   percent filled    I  filled in this range    I
 I---------------------I--------------------------I
 I      61 -     65    I                1         I
 I      76 -     80    I                1         I
 I      86 -     90    I                1         I
 I      91 -     95    I                1         I
 ------------------------------------------------


 ------------------------------------------------
 I   Percent ranges    I  Number of blocks with   I
 I   of free space     I free space in this range I
 I---------------------I--------------------------I
 I       1 -      5    I                1         I
 I      11 -     15    I                1         I
 I      16 -     20    I                1         I
 I      36 -     40    I                1         I
 ------------------------------------------------

1
 Number of records per block; Increment = Auto
 ------------------------------------------------
 I  Number of records  I          Number          I
 I     per block       I        of blocks         I
 I---------------------I--------------------------I
 I      13 -     18    I                1         I
 I      19 -     24    I                1         I
 I      25 -     30    I                2         I
 ------------------------------------------------

1
 Normal records length; Increment = Auto
 ------------------------------------------------
 I        Length       I          Number          I
 I      of record      I        of records        I
 I---------------------I--------------------------I
 I     121 -    126    I                1         I
 I     133 -    138    I                2         I
 I     139 -    144    I                2         I
 I     145 -    150    I                6         I
 I     151 -    156    I               14         I
 I     157 -    162    I               16         I
 I     163 -    168    I               13         I
 I     169 -    174    I               10         I
 I     175 -    180    I                9         I
 I     181 -    186    I               11         I
 I     187 -    192    I                4         I
 I     193 -    198    I                4         I
 I     199 -    204    I                3         I
 I     205 -    210    I                3         I
```

```
 ------------------------------------------------

1

1
 ************************************************************************
 ************************************************************************
 ************************************************************************
 ****                                                                ****
 ****             F I L E     20 ( TESTFILE         )                ****
 ****                                                                ****
 ****                        2012-05-11 12:48:31                     ****
 ************************************************************************
 ************************************************************************
 ************************************************************************


 Consolidated input parameters:
 ------------------------------
  File number = 20
            ERRLIMIT      = 1
            LOB-GROUP     = No
            EXP-GROUP     = Yes
      ASSO  Report type   = SUMMARY + DETAIL
            SAMPLING      = 100    (10 of every 10 records read)
            INCRUSAGE     = 5
            INCRNUMREC    = Auto
            FROM/TO RABN  = 2188         - 2201
      DATA  Report type   = SUMMARY + DETAIL
            SAMPLING      = 100    (10 of every 10 records read)
            INCRUSAGE     = 5
            INCRNUMREC    = Auto
            INCRRECLEN    = Auto
            FROM/TO RABN  = 5402         - 5405

1

 File options (from FCB):
 ------------------------
 Date / Time Loaded ..................... 2011-09-27 11:20:20.72
 Date / Time of last update ............. 2011-09-27 11:39:24.72
 Date / Time of last physical restructure 2011-09-27 11:20:20.72

 ISN Reusage ............................ No
 DATA Storage Reusage ................... Yes
 MIXDSDEV ............................... No
 Spanned Records Support ................ No
 Index compressed ....................... No
 2-Byte MU/PE Index ..................... No
 Highest Index Level .................... 3

 LOB fields ............................. No
```

```
 LOB file ............................... No
 Expanded ............................... Yes. Seq.number = 2. ANCHOR = 22

1
 I//////////////////////////////////////////////////////////////////I
 II                                                                  II
 II     ASSOCIATOR SUMMARY REPORT OF NORMAL INDEX FOR FILE 20        II
 II                                                                  II
 I//////////////////////////////////////////////////////////////////I

 Physical extents list
 ---------------------------------------------------------------------
 I Extent I Type I          From           To  I % unused I Device I
 I        I      I          RABN          RABN I  blocks  I  type  I
 I--------I------I----------------------------I----------I--------I
 I     1  I  NI  I          2188          2207 I     30   I  3390  I
 I        I      I                             I          I        I
 ---------------------------------------------------------------------

 Range of RABNs being processed according
 to consolidated input parameters
 (unused RABNs at the end of extents excluded)
 ------------------------------------------------
 I Extent I Type I          From           To  I
 I        I      I          RABN          RABN I
 I--------I------I----------------------------I
 I     1  I  NI  I          2188          2201 I
 I        I      I                             I
 ------------------------------------------------

 File ......................... 20
 Device Type .................. 3390
 Block Size ................... 2544
 ASSO Padding factor .......... 10%   (254 bytes)


 Number of allocated blocks ..............................  20
 Number of used blocks ...................................  14
 Number of blocks read (SAMPLING = 100%) .................  14
 Number of unused blocks at the end of extents ...........  6
 Number of unused blocks within the used extents .........  0
 Percent of blocks with free space less than padding ......  0
 Average percent of block filling (unused blocks excluded)  38
 Number of elements in blocks read .......................  780
 Average number of elements per block ....................  56
 Average element length ..................................  17
 Minimum element length ..................................  8
 Maximum element length ..................................  299
 Number of elements with minimum length ..................  30
 Number of elements with maximum length ..................  1

1
```

```
I================================================I
II                                              II
II    DISTRIBUTION REPORTS ASSO NORMAL INDEX    II
II                   File = 20                  II
II             Device type = 3390               II
I================================================I

Block filling level; Increment = 5%
------------------------------------------------
I      Range of       I    Number of blocks      I
I   percent filled     I  filled in this range    I
I--------------------I--------------------------I
I      11 -     15    I                2         I
I      16 -     20    I                3         I
I      21 -     25    I                2         I
I      26 -     30    I                1         I
I      31 -     35    I                1         I
I      41 -     45    I                1         I
I      51 -     55    I                1         I
I      66 -     70    I                1         I
I      86 -     90    I                2         I
------------------------------------------------

------------------------------------------------
I   Percent ranges    I  Number of blocks with   I
I   of free space     I free space in this range I
I--------------------I--------------------------I
I       6 -     10    I                2         I
I      31 -     35    I                1         I
I      46 -     50    I                1         I
I      56 -     60    I                1         I
I      61 -     65    I                1         I
I      71 -     75    I                1         I
I      76 -     80    I                3         I
I      81 -     85    I                2         I
I      86 -     90    I                2         I
------------------------------------------------

1
 Number of elements per block; Increment = Auto
------------------------------------------------
I  Number of elements I          Number          I
I     per block       I        of blocks         I
I--------------------I--------------------------I
I       4 -      6    I                2         I
I       7 -      9    I                1         I
I      22 -     24    I                2         I
I      40 -     42    I                2         I
I      46 -     48    I                2         I
I      94 -     96    I                1         I
I      97 -     99    I                1         I
```

```
I     100 -    102    I                1         I
I     118 -    120    I                1         I
I     121 -    123    I                1         I
-------------------------------------------------


1
 I////////////////////////////////////////////////////////////////I
 II                                                              II
 II          DATA STORAGE SUMMARY REPORT FOR FILE 20             II
 II                                                              II
 I////////////////////////////////////////////////////////////////I

 Physical extents list
 ---------------------------------------------------------------------
 I Extent I Type I          From           To  I % unused I Device I
 I        I      I          RABN          RABN I  blocks  I  type  I
 I--------I------I-----------------------------I----------I--------I
 I     1  I  DS  I          5402          5421 I     80   I  3390  I
 I        I      I                               I          I        I
 ---------------------------------------------------------------------

 Range of RABNs being processed according
 to consolidated input parameters
 (unused RABNs at the end of extents excluded)
 ------------------------------------------------
 I Extent I Type I          From           To  I
 I        I      I          RABN          RABN I
 I--------I------I-----------------------------I
 I     1  I  DS  I          5402          5405 I
 I        I      I                               I
 ------------------------------------------------

 File ......................... 20
 Device Type .................. 3390
 Block Size.................... 5064
 DATA Storage Padding factor .. 10%   (506 bytes)


 Number of allocated blocks ..............................  20
 Number of used blocks ...................................  4
 Number of blocks read (SAMPLING = 100%) .................  4
 Number of unused blocks at the end of extents ...........  16
 Number of unused blocks within the used extents .........  0
 Percent of blocks with free space less than padding ......  25
 Average percent of block filling (unused blocks excluded)  83
 Number of records in blocks read ........................  100
    Number of Normal records .............................  100
    Number of Spanned records ............................  n/a
 Number of blocks with Spanned records ...................  n/a
 Average number of records per block .....................  25

 Normal records:
```

```
    Average record length ..............................  168
    Minimum record length ..............................  126
    Maximum record length ..............................  208
    Number of records with minimum length ..............  1
    Number of records with maximum length ..............  1

1

 I================================================I
 II                                              II
 II       DISTRIBUTION REPORTS DATA STORAGE      II
 II                  File = 20                   II
 II             Device type = 3390               II
 I================================================I

 Block filling level; Increment = 5%
 ------------------------------------------------
 I      Range of       I    Number of blocks      I
 I   percent filled    I  filled in this range    I
 I--------------------I--------------------------I
 I      61 -     65    I                1         I
 I      86 -     90    I                2         I
 I      91 -     95    I                1         I
 ------------------------------------------------

 ------------------------------------------------
 I   Percent ranges    I  Number of blocks with   I
 I   of free space     I free space in this range I
 I--------------------I--------------------------I
 I       1 -      5    I                1         I
 I      11 -     15    I                2         I
 I      36 -     40    I                1         I
 ------------------------------------------------

1
 Number of records per block; Increment = Auto
 ------------------------------------------------
 I  Number of records  I          Number          I
 I     per block       I        of blocks         I
 I--------------------I--------------------------I
 I      13 -     18    I                1         I
 I      25 -     30    I                3         I
 ------------------------------------------------

1
 Normal records length; Increment = Auto
 ------------------------------------------------
 I        Length       I          Number          I
 I      of record      I        of records        I
 I--------------------I--------------------------I
 I     121 -    126    I                1         I
 I     133 -    138    I                2         I
```

```
I     139 -    144    I                2         I
I     145 -    150    I                6         I
I     151 -    156    I               14         I
I     157 -    162    I               16         I
I     163 -    168    I               13         I
I     169 -    174    I               11         I
I     175 -    180    I                9         I
I     181 -    186    I               11         I
I     187 -    192    I                5         I
I     193 -    198    I                4         I
I     199 -    204    I                3         I
I     205 -    210    I                3         I
---------------------------------------------------

1

1
*************************************************************************
*************************************************************************
*************************************************************************
****                                                                 ****
****             F I L E     13 ( TESTFILE         )                 ****
****                                                                 ****
****                        2012-05-11 12:48:31                      ****
*************************************************************************
*************************************************************************
*************************************************************************


Consolidated input parameters:
------------------------------
 File number = 13
           ERRLIMIT      = 1
           LOB-GROUP     = No
           EXP-GROUP     = Yes
     ASSO  Report type   = SUMMARY + DETAIL
           SAMPLING      = 100    (10 of every 10 records read)
           INCRUSAGE     = 5
           INCRNUMREC    = Auto
           FROM/TO RABN  = 3761         - 3768
     DATA  Report type   = SUMMARY + DETAIL
           SAMPLING      = 100    (10 of every 10 records read)
           INCRUSAGE     = 5
           INCRNUMREC    = Auto
           INCRRECLEN    = Auto
           FROM/TO RABN  = 9523         - 9523

1

File options (from FCB):
------------------------
Date / Time Loaded ..................... 2011-09-27 11:20:36.45
```

```
 Date / Time of last update ..... 2011-10-04 07:05:44.83
 Date / Time of last physical restructure 2011-09-27 11:20:36.45

 ISN Reusage ..................... No
 DATA Storage Reusage ............ Yes
 MIXDSDEV ........................ No
 Spanned Records Support ......... No
 Index compressed ................ No
 2-Byte MU/PE Index .............. No
 Highest Index Level ............. 3

 LOB fields ...................... No
 LOB file ........................ No
 Expanded ........................ Yes. Seq.number = 3. ANCHOR = 22

1
 I/////////////////////////////////////////////////////////////I
 II                                                           II
 II     ASSOCIATOR SUMMARY REPORT OF NORMAL INDEX FOR FILE 13     II
 II                                                           II
 I/////////////////////////////////////////////////////////////I

 Physical extents list
 ------------------------------------------------------------------
 I Extent I Type I          From           To  I % unused I Device I
 I        I      I          RABN          RABN I  blocks  I  type  I
 I--------I------I----------------------------I----------I--------I
 I     1  I  NI  I          3761          3780 I     60   I  3390  I
 I        I      I                            I          I        I
 ------------------------------------------------------------------

 Range of RABNs being processed according
 to consolidated input parameters
 (unused RABNs at the end of extents excluded)
 ------------------------------------------------
 I Extent I Type I          From           To  I
 I        I      I          RABN          RABN I
 I--------I------I----------------------------I
 I     1  I  NI  I          3761          3768 I
 I        I      I                            I
 ------------------------------------------------

 File ......................... 13
 Device Type .................. 3390
 Block Size ................... 2544
 ASSO Padding factor .......... 10%   (254 bytes)


 Number of allocated blocks ..............................  20
 Number of used blocks ...................................  8
 Number of blocks read (SAMPLING = 100%) .................  8
 Number of unused blocks at the end of extents ...........  12
```

```
 Number of unused blocks within the used extents .........  0
 Percent of blocks with free space less than padding ......  0
 Average percent of block filling (unused blocks excluded)   1
 Number of elements in blocks read .......................  29
 Average number of elements per block ....................  4
 Average element length ..................................  10
 Minimum element length ..................................  6
 Maximum element length ..................................  23
 Number of elements with minimum length ..................  1
 Number of elements with maximum length ..................  1

1

 I================================================I
 II                                              II
 II    DISTRIBUTION REPORTS ASSO NORMAL INDEX    II
 II                   File = 13                  II
 II             Device type = 3390               II
 I================================================I

 Block filling level; Increment = 5%
 ------------------------------------------------
 I      Range of       I    Number of blocks      I
 I   percent filled    I  filled in this range    I
 I---------------------I--------------------------I
 I       1 -      5    I                7         I
 I       6 -     10    I                1         I
 ------------------------------------------------

 ------------------------------------------------
 I   Percent ranges    I  Number of blocks with   I
 I   of free space     I free space in this range I
 I---------------------I--------------------------I
 I      91 -     95    I                1         I
 I      96 -    100    I                7         I
 ------------------------------------------------

1
 Number of elements per block; Increment = Auto
 ------------------------------------------------
 I  Number of elements I          Number          I
 I     per block       I        of blocks         I
 I---------------------I--------------------------I
 I       1 -      3    I                6         I
 I       4 -      6    I                1         I
 I      16 -     18    I                1         I
 ------------------------------------------------

1
 I////////////////////////////////////////////////////////////I
 II                                                          II
 II          DATA STORAGE SUMMARY REPORT FOR FILE 13         II
```

```
II                                                                  II
I//////////////////////////////////////////////////////////////////I

Physical extents list
-----------------------------------------------------------------
I Extent I Type I          From           To  I % unused I Device I
I        I      I          RABN          RABN I  blocks  I  type  I
I--------I------I-----------------------------I----------I--------I
I     1  I  DS  I          9523          9542 I     95   I  3390  I
I        I      I                               I          I        I
-----------------------------------------------------------------

Range of RABNs being processed according
to consolidated input parameters
(unused RABNs at the end of extents excluded)
-----------------------------------------------
I Extent I Type I          From           To  I
I        I      I          RABN          RABN I
I--------I------I-----------------------------I
I     1  I  DS  I          9523          9523 I
I        I      I                               I
-----------------------------------------------

File ......................... 13
Device Type .................. 3390
Block Size.................... 5064
DATA Storage Padding factor .. 10%   (506 bytes)


Number of allocated blocks ..............................  20
Number of used blocks ...................................  1
Number of blocks read (SAMPLING = 100%) .................  1
Number of unused blocks at the end of extents ...........  19
Number of unused blocks within the used extents .........  0
Percent of blocks with free space less than padding ......  0
Average percent of block filling (unused blocks excluded)   6
Number of records in blocks read ........................  6
   Number of Normal records .............................  6
   Number of Spanned records ............................  n/a
Number of blocks with Spanned records ...................  n/a
Average number of records per block .....................  6

Normal records:
   Average record length ................................  47
   Minimum record length ................................  31
   Maximum record length ................................  71
   Number of records with minimum length ................  1
   Number of records with maximum length ................  1

1

 I================================================I
```

```
 II                                                II
 II       DISTRIBUTION REPORTS DATA STORAGE        II
 II                    File = 13                   II
 II              Device type = 3390                II
 I================================================I

 Block filling level; Increment = 5%
 ------------------------------------------------
 I      Range of       I    Number of blocks      I
 I   percent filled    I  filled in this range    I
 I--------------------I-------------------------I
 I       6 -     10    I                1         I
 ------------------------------------------------

 ------------------------------------------------
 I   Percent ranges    I  Number of blocks with   I
 I   of free space     I free space in this range I
 I--------------------I-------------------------I
 I      91 -     95    I                1         I
 ------------------------------------------------

1
 Number of records per block; Increment = Auto
 ------------------------------------------------
 I  Number of records  I          Number          I
 I     per block       I        of blocks         I
 I--------------------I-------------------------I
 I       1 -      6    I                1         I
 ------------------------------------------------

1
 Normal records length; Increment = Auto
 ------------------------------------------------
 I        Length       I          Number          I
 I      of record      I        of records        I
 I--------------------I-------------------------I
 I      31 -     36    I                2         I
 I      43 -     48    I                2         I
 I      55 -     60    I                1         I
 I      67 -     72    I                1         I
 ------------------------------------------------

 ****************************************************************
 ****************************************************************
 ****************************************************************
 ****                                                        ****
 ****     Summary reports about all parts of expanded file:  ****
 ****    22   20   13                                        ****
 ****************************************************************
 ****************************************************************
 ****************************************************************
1
```

```
 I//////////////////////////////////////////////////////////////////I
 II                                                                II
 II     ASSOCIATOR SUMMARY REPORT OF NORMAL INDEX                  II
 II           (All parts of expanded file)                         II
 I//////////////////////////////////////////////////////////////////I

 Number of allocated blocks ..............................  60
 Number of used blocks ...................................  36
 Number of blocks read (SAMPLING = 100%) .................  36
 Number of unused blocks at the end of extents ...........  24
 Number of unused blocks within the used extents .........  0
 Percent of blocks with free space less than padding ......  0
 Average percent of block filling (unused blocks excluded)   30
 Number of elements in blocks read .......................  1576
 Average number of elements per block ....................  44
 Average element length ..................................  17
 Minimum element length ..................................  6
 Maximum element length ..................................  299
 Number of elements with minimum length ..................  1
 Number of elements with maximum length ..................  1

1

 I================================================I
 II                                              II
 II    DISTRIBUTION REPORTS ASSO NORMAL INDEX    II
 II          (All parts of expanded file)        II
 II             Device type = 3390               II
 I================================================I

 Block filling level; Increment = 5%
 ------------------------------------------------
 I      Range of       I    Number of blocks      I
 I   percent filled    I  filled in this range    I
 I--------------------I-------------------------I
 I       1 -      5    I                7         I
 I       6 -     10    I                1         I
 I      11 -     15    I                4         I
 I      16 -     20    I                6         I
 I      21 -     25    I                4         I
 I      26 -     30    I                2         I
 I      31 -     35    I                2         I
 I      41 -     45    I                2         I
 I      46 -     50    I                1         I
 I      51 -     55    I                1         I
 I      66 -     70    I                2         I
 I      86 -     90    I                4         I
 ------------------------------------------------

 ------------------------------------------------
 I   Percent ranges    I  Number of blocks with   I
 I   of free space     I free space in this range I
```

```
 I---------------------I--------------------------I
 I       6 -     10    I                3         I
 I      11 -     15    I                1         I
 I      31 -     35    I                2         I
 I      46 -     50    I                2         I
 I      56 -     60    I                2         I
 I      61 -     65    I                1         I
 I      66 -     70    I                1         I
 I      71 -     75    I                2         I
 I      76 -     80    I                5         I
 I      81 -     85    I                5         I
 I      86 -     90    I                4         I
 I      91 -     95    I                1         I
 I      96 -    100    I                7         I
 ------------------------------------------------


1
 Number of elements per block; Increment = Auto
 ------------------------------------------------
 I  Number of elements I          Number          I
 I     per block       I        of blocks         I
 I---------------------I--------------------------I
 I       1 -      3    I                6         I
 I       4 -      6    I                5         I
 I       7 -      9    I                2         I
 I      16 -     18    I                1         I
 I      22 -     24    I                4         I
 I      40 -     42    I                4         I
 I      46 -     48    I                4         I
 I      91 -     93    I                1         I
 I      94 -     96    I                2         I
 I      97 -     99    I                2         I
 I     100 -    102    I                1         I
 I     115 -    117    I                1         I
 I     118 -    120    I                1         I
 I     121 -    123    I                2         I
 ------------------------------------------------


1
 I//////////////////////////////////////////////////////////////I
 II                                                            II
 II          DATA STORAGE SUMMARY REPORT                       II
 II         (All parts of expanded file)                       II
 I//////////////////////////////////////////////////////////////I

 Number of allocated blocks ..............................  60
 Number of used blocks ...................................  9
 Number of blocks read (SAMPLING = 100%) .................  9
 Number of unused blocks at the end of extents ...........  51
 Number of unused blocks within the used extents .........  0
 Percent of blocks with free space less than padding ......  22
 Average percent of block filling (unused blocks excluded)  74
```

```
 Number of records in blocks read .......................  204
    Number of Normal records ............................  204
    Number of Spanned records ...........................  n/a
 Number of blocks with Spanned records ..................  n/a
 Average number of records per block ....................  23

 Normal records:
    Average record length ...............................  165
    Minimum record length ...............................  31
    Maximum record length ...............................  208
    Number of records with minimum length ...............  1
    Number of records with maximum length ...............  2

1

 I================================================I
 II                                              II
 II       DISTRIBUTION REPORTS DATA STORAGE      II
 II          (All parts of expanded file)        II
 II             Device type = 3390               II
 I================================================I

 Block filling level; Increment = 5%
 ------------------------------------------------
 I      Range of       I    Number of blocks      I
 I   percent filled    I  filled in this range    I
 I--------------------I--------------------------I
 I       6 -     10    I                1         I
 I      61 -     65    I                2         I
 I      76 -     80    I                1         I
 I      86 -     90    I                3         I
 I      91 -     95    I                2         I
 ------------------------------------------------

 ------------------------------------------------
 I   Percent ranges    I  Number of blocks with   I
 I   of free space     I free space in this range I
 I--------------------I--------------------------I
 I       1 -      5    I                2         I
 I      11 -     15    I                3         I
 I      16 -     20    I                1         I
 I      36 -     40    I                2         I
 I      91 -     95    I                1         I
 ------------------------------------------------

1
 Number of records per block; Increment = Auto
 ------------------------------------------------
 I  Number of records  I          Number          I
 I     per block       I        of blocks         I
 I--------------------I--------------------------I
 I       1 -      6    I                1         I
```

```
I      13 -     18    I                2         I
I      19 -     24    I                1         I
I      25 -     30    I                5         I
-------------------------------------------------


1
 Normal records length; Increment = Auto
 -------------------------------------------------
 I        Length       I          Number          I
 I      of record      I        of records        I
 I---------------------I--------------------------I
 I      31 -     36    I                2         I
 I      43 -     48    I                2         I
 I      49 -     60    I                1         I
 I      61 -    120    I                1         I
 I     121 -    126    I                2         I
 I     133 -    138    I                4         I
 I     139 -    144    I                4         I
 I     145 -    150    I               12         I
 I     151 -    156    I               28         I
 I     157 -    162    I               32         I
 I     163 -    168    I               26         I
 I     169 -    174    I               21         I
 I     175 -    180    I               18         I
 I     181 -    186    I               22         I
 I     187 -    192    I                9         I
 I     193 -    198    I                8         I
 I     199 -    204    I                6         I
 I     205 -    210    I                6         I
 -------------------------------------------------



1
 ************************************************************************
 ************************************************************************
 ************************************************************************
 ****                                                                ****
 ****       F i l e s   s t a t i s t i c a l   r e p o r t        ****
 ****                                                                ****
 ************************************************************************
 ************************************************************************
 ************************************************************************

 ---------------------------------------------------------------------------
 I F I  Duration I     ASSOCIATOR (NI)    I      DATA STORAGE      I ParmI Run I
 I i IHH:MM:SS.thI------------------------I------------------------I-----I-----I
 I l I           I  Number  I E I Extents I  Number  I E I Extents I E W I E W I
 I e I           I of RABNs I r I---------I of RABNs I r I---------I r n I r n I
 I   I           I   read   I r I FCB Run I   read   I r I FCB Run I r g I r g I
 I---I-----------I----------I---I----I----I----------I---I----I----I--I--I--I--I
   13 00:00:00.05          8   0    1    1          1   0    1    1  0  0  0  1
```

```
   20 00:00:00.02         14   0    1    1          4   0    1    1  0  0  0  1
   22 00:00:00.05         14   0    1    1          4   0    1    1  0  0  0  0



1

 DRU205I, ******* SPACEUSAGE-1 ended *******     CC = 04



DDDETAIL


1File options (from FCB):
 ------------------------
 Date / Time Loaded ..................... 2011-09-27 11:20:10.24
 Date / Time of last update ............. 2011-10-21 12:17:55.72
 Date / Time of last physical restructure 2011-09-27 11:20:10.24

 ISN Reusage ............................ No
 DATA Storage Reusage ................... Yes
 MIXDSDEV ............................... No
 Spanned Records Support ................ No
 Index compressed ....................... No
 2-Byte MU/PE Index ..................... No
 Highest Index Level .................... 3

 LOB fields ............................. No
 LOB file ............................... No
 Expanded ............................... Yes. Seq.number = 1. ANCHOR = 22


 I/////////////////////////////////////////////////////////////I
 II                                                           II
 II    ASSOCIATOR DETAIL REPORT OF NORMAL INDEX FOR FILE 22   II
 II                                                           II
 I/////////////////////////////////////////////////////////////I

 Device Type ...........  3390
 Block Size ............. 2544
 ASSO Padding Factor .... 10% (254 bytes)

1File 22                                                Page 1
 ------------------------------------------------------------------------
 I   RABN   I   %     I Block  I Free Space I Num  I Element lengths I
 I          I  Used   I length I------------I elem I-----------------I
 I          I         I        I Gap I Pad  I      I Min I Max I Avg I
 I----------I---------I--------I-----I------I------I-----I-----I-----I
        2151     50      1277    1013    254     98    13    13    13
        2152     28       719    1571    254     47     8    35    15
```

```
        2153     34       873    1417    254     96     9    12     9
        2154     14       353    1937    254      6    10   128    58
        2155     19       481    1809    254     23    11    35    20
        2156     42      1074    1216    254     42    11    57    25
        2157     19       484    1806    254      5    11   293    96
        2158     21       543    1747    254     41     9    33    13
        2159     21       532    1758    254     47     8    29    11
        2160     14       351    1939    254      9     9    78    38
        2161     67      1703     587    254     93    14    25    18
        2162     90      2277      13    254    121    18    33    18
        2163     87      2205      85    254    115    18    30    19
        2164     17       435    1855    254     24    18    18    18



 I//////////////////////////////////////////////////////////////////I
 II                                                                II
 II            DATA STORAGE DETAIL REPORT FOR FILE 22              II
 II                                                                II
 I//////////////////////////////////////////////////////////////////I

 Device Type ...........  3390
 Block Size ............. 5064
 DATA Padding Factor .... 10% (506 bytes)

1File 22                                                      Page 1
 ------------------------------------------------------------------------------
 I   RABN   I   %    I Block  I Free space I Num I  Spanned  I Record lengths  I
 I          I  Used  I Length I------------I Rec I-----------I-----------------I
 I          I        I        I Gap I Pad  I     I Num I Flg I Min I Max I Avg I
 I----------I--------I--------I-----I------I-----I-----I-----I-----I-----I-----I
        5382    80       4057    501   506     24    0          126   208   168
        5383    95       4821      0   243     30    0          136   189   160
        5384    88       4469     89   506     26    0          143   201   171
        5385    63       3207   1351   506     18    0          153   205   177


1File options (from FCB):
 ------------------------
 Date / Time Loaded ..................... 2011-09-27 11:20:20.72
 Date / Time of last update ............. 2011-09-27 11:39:24.72
 Date / Time of last physical restructure 2011-09-27 11:20:20.72

 ISN Reusage ............................ No
 DATA Storage Reusage ................... Yes
 MIXDSDEV ............................... No
 Spanned Records Support ................ No
 Index compressed ....................... No
 2-Byte MU/PE Index ..................... No
 Highest Index Level .................... 3

 LOB fields ............................. No
```

```
 LOB file ............................... No
 Expanded ............................... Yes. Seq.number = 2. ANCHOR = 22


 I/////////////////////////////////////////////////////////////////I
 II                                                               II
 II    ASSOCIATOR DETAIL REPORT OF NORMAL INDEX FOR FILE 20       II
 II                                                               II
 I/////////////////////////////////////////////////////////////////I

 Device Type ...........  3390
 Block Size ............. 2544
 ASSO Padding Factor .... 10% (254 bytes)

1File 20                                                  Page 1
 ------------------------------------------------------------------------
 I   RABN   I   %     I Block  I Free Space I Num  I Element lengths I
 I          I  Used   I length I------------I elem I-----------------I
 I          I         I        I Gap I Pad  I      I Min I Max I Avg I
 I----------I---------I--------I-----I------I------I-----I-----I-----I
        2188     51      1303     987    254    100    13    13    13
        2189     29       739    1551    254     48     8    35    15
        2190     35       891    1399    254     98     9    12     9
        2191     14       359    1931    254      6    10   128    59
        2192     19       487    1803    254     23    11    35    21
        2193     42      1080    1210    254     42    11    57    25
        2194     20       505    1785    254      5    11   299   100
        2195     22       555    1735    254     42     9    33    13
        2196     21       543    1747    254     48     8    29    11
        2197     14       357    1933    254      9     9    78    39
        2198     68      1742     548    254     95    14    25    18
        2199     90      2277      13    254    121    18    33    18
        2200     90      2283       7    254    119    18    30    19
        2201     17       435    1855    254     24    18    18    18



 I/////////////////////////////////////////////////////////////////I
 II                                                               II
 II            DATA STORAGE DETAIL REPORT FOR FILE 20             II
 II                                                               II
 I/////////////////////////////////////////////////////////////////I

 Device Type ...........  3390
 Block Size ............. 5064
 DATA Padding Factor .... 10% (506 bytes)

1File 20                                                      Page 1
 ------------------------------------------------------------------------------
 I   RABN   I   %    I Block  I Free space I Num I  Spanned  I Record lengths  I
 I          I  Used  I Length I------------I Rec I-----------I-----------------I
 I          I        I        I Gap I Pad  I     I Num I Flg I Min I Max I Avg I
```

```
 I----------I--------I--------I-----I------I-----I-----I-----I-----I-----I-----I
        5402    87       4415    143   506     26    0          126   208   169
        5403    95       4821      0   243     30    0          136   189   160
        5404    88       4469     89   506     26    0          143   201   171
        5405    63       3207   1351   506     18    0          153   205   177


1File options (from FCB):
 ------------------------
 Date / Time Loaded ..................... 2011-09-27 11:20:36.45
 Date / Time of last update ............. 2011-10-04 07:05:44.83
 Date / Time of last physical restructure 2011-09-27 11:20:36.45

 ISN Reusage ............................ No
 DATA Storage Reusage ................... Yes
 MIXDSDEV ............................... No
 Spanned Records Support ................ No
 Index compressed ....................... No
 2-Byte MU/PE Index ..................... No
 Highest Index Level .................... 3

 LOB fields ............................. No
 LOB file ............................... No
 Expanded ............................... Yes. Seq.number = 3. ANCHOR = 22


 I//////////////////////////////////////////////////////////////////I
 II                                                                  II
 II    ASSOCIATOR DETAIL REPORT OF NORMAL INDEX FOR FILE 13          II
 II                                                                  II
 I//////////////////////////////////////////////////////////////////I

 Device Type ...........  3390
 Block Size ............. 2544
 ASSO Padding Factor .... 10% (254 bytes)

1File 13                                                  Page 1
 ------------------------------------------------------------------------
 I   RABN   I   %     I Block  I Free Space I Num  I Element lengths I
 I          I  Used   I length I------------I elem I-----------------I
 I          I         I        I Gap I Pad  I      I Min I Max I Avg I
 I----------I---------I--------I-----I------I------I-----I-----I-----I
        3761      3        74    2216    254      6     6    13    11
        3762      1        24    2266    254      1    21    21    21
        3763      1        26    2264    254      1    23    23    23
        3764      1        18    2272    254      1    15    15    15
        3765      1        18    2272    254      1    15    15    15
        3766      6       142    2148    254     17     8    11     8
        3767      1        12    2278    254      1     9     9     9
        3768      1        18    2272    254      1    15    15    15
```

```
 I//////////////////////////////////////////////////////////////////I
 II                                                                II
 II            DATA STORAGE DETAIL REPORT FOR FILE 13              II
 II                                                                II
 I//////////////////////////////////////////////////////////////////I

 Device Type ...........  3390
 Block Size ............. 5064
 DATA Padding Factor .... 10% (506 bytes)

1File 13                                                     Page 1
 ------------------------------------------------------------------------------
 I   RABN   I   %    I Block  I Free space I Num I  Spanned  I Record lengths  I
 I          I  Used  I Length I------------I Rec I-----------I-----------------I
 I          I        I        I Gap I Pad  I     I Num I Flg I Min I Max I Avg I
 I----------I--------I--------I-----I------I-----I-----I-----I-----I-----I-----I
        9523     6        288   4270   506      6    0           31    71    47
```

Sample 6: XML Output for Detail and Summary Report

This sample shows ADADRU SPACEUSAGE processing of the OUTPUT parameter. It shows summary and detail ASSO and DATA space usage on file 10 (EMPLOYEES) in XML format.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="adadru_summary.xsl"?>
<AdadruSummary>
 <RunTime>
  <DBID>192</DBID>
  <VER>8</VER>
  <Release>3</Release>
  <SM>8</SM>
  <Date>2013-04-17</Date>
  <Time>10:30:07</Time>
 </RunTime>
 <SpaceUsage>
  <SpaceUsageNum>1</SpaceUsageNum>
  <File>
   <Number>10</Number>
   <FileName>EMPL-192</FileName>
   <ConsolidatedParameters>
    <ErrLimit>1</ErrLimit>
    <LOBGroup>No</LOBGroup>
    <ExpGroup>No</ExpGroup>
    <XML>XML</XML>
    <ACODE>No</ACODE>
    <AssoParameters>
     <Report>SUMMARY + DETAIL</Report>
     <SAMPLING>100    (10 of every 10 records read)</SAMPLING>
     <INCRUSAGE>5</INCRUSAGE>
```

```
    <INCRNUMREC>Auto</INCRNUMREC>
    <RABN>
     <FROMRABN>3700</FROMRABN>
     <TORABN>3749</TORABN>
    </RABN>
    <RABN>
     <FROMRABN>3780</FROMRABN>
     <TORABN>3786</TORABN>
    </RABN>
   </AssoParameters>
   <DataParameters>
    <Report>SUMMARY + DETAIL</Report>
    <SAMPLING>100    (10 of every 10 records read)</SAMPLING>
    <INCRUSAGE>5</INCRUSAGE>
    <INCRNUMREC>Auto</INCRNUMREC>
    <INCRRECLEN>Auto</INCRRECLEN>
    <RABN>
     <FROMRABN>7079</FROMRABN>
     <TORABN>7122</TORABN>
    </RABN>
   </DataParameters>
  </ConsolidatedParameters>
  <FileOptions>
   <DateLoaded>2013-03-11</DateLoaded>
   <TimeLoaded>12:01:05.21</TimeLoaded>
   <DateUpdated>n/a</DateUpdated>
   <TimeUpdated>n/a</TimeUpdated>
   <DateRestructured>2013-03-11</DateRestructured>
   <TimeRestructured>12:01:05.21</TimeRestructured>
   <ISNReusage>Yes</ISNReusage>
   <DSReusage>Yes</DSReusage>
   <MixDSDev>No</MixDSDev>
   <Spanned>No</Spanned>
   <IndexCompressed>No</IndexCompressed>
   <TwoByteIndex>No</TwoByteIndex>
   <HighestIndexLevel>3</HighestIndexLevel>
   <LOBFields>No</LOBFields>
   <LOBFile>No</LOBFile>
   <Expanded>No</Expanded>
  </FileOptions>
  <AssoSummaryReport>
   <NILayout>
    <PhysicalLayout>
     <Type>NI</Type>
     <FromRABN>3700</FromRABN>
     <ToRABN>3749</ToRABN>
     <UnusedBlocks>0</UnusedBlocks>
     <Device>3390</Device>
    </PhysicalLayout>
    <PhysicalLayout>
     <Type>NI</Type>
     <FromRABN>3780</FromRABN>
```

```
      <ToRABN>3879</ToRABN>
      <UnusedBlocks>93</UnusedBlocks>
      <Device>3390</Device>
     </PhysicalLayout>
     <ProcessingLayout>
      <Type>NI</Type>
      <FromRABN>3700</FromRABN>
      <ToRABN>3749</ToRABN>
     </ProcessingLayout>
     <ProcessingLayout>
      <Type>NI</Type>
      <FromRABN>3780</FromRABN>
      <ToRABN>3786</ToRABN>
     </ProcessingLayout>
    </NILayout>
    <Device>
     <DeviceType>3390</DeviceType>
     <BlockSize>2544</BlockSize>
     <PaddingSpace>127</PaddingSpace>
     <PaddingFactor>5%</PaddingFactor>
     <NumberOfAllocatedBlocks>150</NumberOfAllocatedBlocks>
     <NumberOfUsedblocks>57</NumberOfUsedblocks>
     <BlocksRead>
      <Sampling>100%</Sampling>
      <Number>57</Number>
     </BlocksRead>
     <NumberOfUnusedBlocksAtTheEnd>93</NumberOfUnusedBlocksAtTheEnd>
     <NumberOfUnusedBlocksIntheMiddle>0</NumberOfUnusedBlocksIntheMiddle>
     <FreeSpaceLessThenPadding>0</FreeSpaceLessThenPadding>
     <AveragePercentOfBlockFilling>86</AveragePercentOfBlockFilling>
     <NumberOfElementsRead>7071</NumberOfElementsRead>
     <AverageNumberOfElementsPerBlock>124</AverageNumberOfElementsPerBlock>
     <AverageElementLength>17</AverageElementLength>
     <MinimumElementLength>7</MinimumElementLength>
     <MaximumElementLength>2070</MaximumElementLength>
     <NumberOfMinLengthElements>1</NumberOfMinLengthElements>
     <NumberOfMaxLengthElements>1</NumberOfMaxLengthElements>
     <DistributionReports>
      <BlockFillingLevel>
       <Increment>5%</Increment>
       <FilledSpace>
        <PercentRange>
         <From>1</From>
         <To>5</To>
        </PercentRange>
        <NumberOfBlocks>2</NumberOfBlocks>
       </FilledSpace>
       <FilledSpace>
        <PercentRange>
         <From>26</From>
         <To>30</To>
        </PercentRange>
```

```
        <NumberOfBlocks>1</NumberOfBlocks>
       </FilledSpace>
       <FreeSpace>
        <PercentRange>
         <From>1</From>
         <To>5</To>
        </PercentRange>
        <NumberOfBlocks>41</NumberOfBlocks>
       </FreeSpace>
       <FreeSpace>
        <PercentRange>
         <From>6</From>
         <To>10</To>
        </PercentRange>
        <NumberOfBlocks>6</NumberOfBlocks>
       </FreeSpace>
       <FreeSpace>
        <PercentRange>
         <From>11</From>
         <To>15</To>
        </PercentRange>
        <NumberOfBlocks>1</NumberOfBlocks>
       </FreeSpace>
       <FreeSpace>
        <PercentRange>
         <From>16</From>
         <To>20</To>
        </PercentRange>
        <NumberOfBlocks>2</NumberOfBlocks>
       </FreeSpace>
      </BlockFillingLevel>
      <ElementsPerBlock>
       <Increment>Auto</Increment>
       <Elements>
        <ElementsRange>
         <From>1</From>
         <To>3</To>
        </ElementsRange>
        <NumberOfBlocks>1</NumberOfBlocks>
       </Elements>
       <Elements>
        <ElementsRange>
         <From>4</From>
         <To>6</To>
        </ElementsRange>
        <NumberOfBlocks>3</NumberOfBlocks>
       </Elements>
       <Elements>
        <ElementsRange>
         <From>7</From>
         <To>9</To>
        </ElementsRange>
```

```
        <NumberOfBlocks>3</NumberOfBlocks>
       </Elements>
       <Elements>
        <ElementsRange>
         <From>61</From>
         <To>63</To>
        </ElementsRange>
        <NumberOfBlocks>1</NumberOfBlocks>
       </Elements>
       <Elements>
        <ElementsRange>
         <From>64</From>
         <To>66</To>
        </ElementsRange>
        <NumberOfBlocks>1</NumberOfBlocks>
       </Elements>
       <Elements>
        <ElementsRange>
         <From>70</From>
         <To>72</To>
        </ElementsRange>
        <NumberOfBlocks>2</NumberOfBlocks>
       </Elements>
       <Elements>
        <ElementsRange>
         <From>73</From>
         <To>75</To>
        </ElementsRange>
        <NumberOfBlocks>1</NumberOfBlocks>
       </Elements>
       <Elements>
        <ElementsRange>
         <From>79</From>
         <To>81</To>
        </ElementsRange>
        <NumberOfBlocks>1</NumberOfBlocks>
       </Elements>
       <Elements>
        <ElementsRange>
         <From>85</From>
         <To>87</To>
        </ElementsRange>
        <NumberOfBlocks>1</NumberOfBlocks>
       </Elements>
       <Elements>
        <ElementsRange>
         <From>88</From>
         <To>90</To>
        </ElementsRange>
        <NumberOfBlocks>1</NumberOfBlocks>
       </Elements>
       <Elements>
```

```
        <ElementsRange>
         <From>91</From>
         <To>93</To>
        </ElementsRange>
        <NumberOfBlocks>1</NumberOfBlocks>
       </Elements>
       <Elements>
        <ElementsRange>
         <From>103</From>
         <To>105</To>
        </ElementsRange>
        <NumberOfBlocks>1</NumberOfBlocks>
       </Elements>
       <Elements>
        <ElementsRange>
         <From>106</From>
         <To>108</To>
        </ElementsRange>
        <NumberOfBlocks>2</NumberOfBlocks>
       </Elements>
       <Elements>
        <ElementsRange>
         <From>109</From>
         <To>111</To>
        </ElementsRange>
        <NumberOfBlocks>2</NumberOfBlocks>
       </Elements>
       <Elements>
        <ElementsRange>
         <From>115</From>
         <To>117</To>
        </ElementsRange>
        <NumberOfBlocks>3</NumberOfBlocks>
       </Elements>
       <Elements>
        <ElementsRange>
         <From>118</From>
         <To>120</To>
        </ElementsRange>
        <NumberOfBlocks>2</NumberOfBlocks>
       </Elements>
       <Elements>
        <ElementsRange>
         <From>121</From>
         <To>123</To>
        </ElementsRange>
        <NumberOfBlocks>1</NumberOfBlocks>
       </Elements>
       <Elements>
        <ElementsRange>
         <From>124</From>
         <To>126</To>
```

```
        </ElementsRange>
        <NumberOfBlocks>2</NumberOfBlocks>
       </Elements>
       <Elements>
        <ElementsRange>
         <From>127</From>
         <To>129</To>
        </ElementsRange>
        <NumberOfBlocks>2</NumberOfBlocks>
       </Elements>
       <Elements>
        <ElementsRange>
         <From>130</From>
         <To>132</To>
        </ElementsRange>
        <NumberOfBlocks>2</NumberOfBlocks>
       </Elements>
       <Elements>
        <ElementsRange>
         <From>133</From>
         <To>135</To>
        </ElementsRange>
        <NumberOfBlocks>6</NumberOfBlocks>
       </Elements>
       <Elements>
        <ElementsRange>
         <From>136</From>
         <To>138</To>
        </ElementsRange>
        <NumberOfBlocks>1</NumberOfBlocks>
       </Elements>
       <Elements>
        <ElementsRange>
         <From>139</From>
         <To>141</To>
        </ElementsRange>
        <NumberOfBlocks>1</NumberOfBlocks>
       </Elements>
       <Elements>
        <ElementsRange>
         <From>154</From>
         <To>156</To>
        </ElementsRange>
        <NumberOfBlocks>1</NumberOfBlocks>
       </Elements>
       <Elements>
        <ElementsRange>
         <From>178</From>
         <To>180</To>
        </ElementsRange>
        <NumberOfBlocks>1</NumberOfBlocks>
       </Elements>
```

```
      <Elements>
       <ElementsRange>
        <From>181</From>
        <To>183</To>
       </ElementsRange>
       <NumberOfBlocks>1</NumberOfBlocks>
      </Elements>
      <Elements>
       <ElementsRange>
        <From>184</From>
        <To>186</To>
       </ElementsRange>
       <NumberOfBlocks>6</NumberOfBlocks>
      </Elements>
      <Elements>
       <ElementsRange>
        <From>187</From>
        <To>189</To>
       </ElementsRange>
       <NumberOfBlocks>1</NumberOfBlocks>
      </Elements>
      <Elements>
       <ElementsRange>
        <From>190</From>
        <To>192</To>
       </ElementsRange>
       <NumberOfBlocks>1</NumberOfBlocks>
      </Elements>
      <Elements>
       <ElementsRange>
        <From>232</From>
        <To>234</To>
       </ElementsRange>
       <NumberOfBlocks>2</NumberOfBlocks>
      </Elements>
      <Elements>
       <ElementsRange>
        <From>250</From>
        <To>252</To>
       </ElementsRange>
       <NumberOfBlocks>1</NumberOfBlocks>
      </Elements>
      <Elements>
       <ElementsRange>
        <From>253</From>
        <To>255</To>
       </ElementsRange>
       <NumberOfBlocks>1</NumberOfBlocks>
      </Elements>
      <Elements>
       <ElementsRange>
        <From>259</From>
```

```
        <To>261</To>
       </ElementsRange>
       <NumberOfBlocks>1</NumberOfBlocks>
      </Elements>
     </ElementsPerBlock>
    </DistributionReports>
   </Device>
   <DataSummaryReport>
    <DataLayout>
     <PhysicalLayout>
      <Type>DS</Type>
      <FromRABN>7079</FromRABN>
      <ToRABN>7228</ToRABN>
      <UnusedBlocks>71</UnusedBlocks>
      <Device>3390</Device>
     </PhysicalLayout>
     <ProcessingLayout>
      <Type>DS</Type>
      <FromRABN>7079</FromRABN>
      <ToRABN>7122</ToRABN>
     </ProcessingLayout>
    </DataLayout>
    <Device>
     <DeviceType>3390</DeviceType>
     <BlockSize>5064</BlockSize>
     <PaddingSpace>253</PaddingSpace>
     <PaddingFactor>5%</PaddingFactor>
     <NumberOfAllocatedBlocks>150</NumberOfAllocatedBlocks>
     <NumberOfUsedblocks>44</NumberOfUsedblocks>
     <BlocksRead>
      <Sampling>100%</Sampling>
      <Number>44</Number>
     </BlocksRead>
     <NumberOfUnusedBlocksAtTheEnd>106</NumberOfUnusedBlocksAtTheEnd>
     <NumberOfUnusedBlocksIntheMiddle>0</NumberOfUnusedBlocksIntheMiddle>
     <FreeSpaceLessThenPadding>50</FreeSpaceLessThenPadding>
     <AveragePercentOfBlockFilling>93</AveragePercentOfBlockFilling>
     <Records>
      <TotalNumberOfRecords>1107</TotalNumberOfRecords>
      <NormalRecords>1107</NormalRecords>
      <SpannedRecords>n/a</SpannedRecords>
     </Records>
     <NumberOfBlocksWithSpanned>n/a</NumberOfBlocksWithSpanned>
     <AverageNumberOfRecordsPerBlock>25</AverageNumberOfRecordsPerBlock>
     <NormalRecords>
      <AverageRecordLength>188</AverageRecordLength>
      <MinRecordLength>126</MinRecordLength>
      <MaxRecordLength>253</MaxRecordLength>
      <NumberOfMinLengthRecords>1</NumberOfMinLengthRecords>
      <NumberOfMaxLengthRecords>1</NumberOfMaxLengthRecords>
     </NormalRecords>
     <DistributionReports>
```

```
      <BlockFillingLevel>
       <Increment>5%</Increment>
       <FilledSpace>
        <PercentRange>
         <From>26</From>
         <To>30</To>
        </PercentRange>
        <NumberOfBlocks>1</NumberOfBlocks>
       </FilledSpace>
       <FreeSpace>
        <PercentRange>
         <From>1</From>
         <To>5</To>
        </PercentRange>
        <NumberOfBlocks>27</NumberOfBlocks>
       </FreeSpace>
       <FreeSpace>
        <PercentRange>
         <From>6</From>
         <To>10</To>
        </PercentRange>
        <NumberOfBlocks>16</NumberOfBlocks>
       </FreeSpace>
      </BlockFillingLevel>
      <RecordsPerBlock>
       <Increment>Auto</Increment>
       <Records>
        <RecordsRange>
         <From>7</From>
         <To>12</To>
        </RecordsRange>
        <NumberOfBlocks>1</NumberOfBlocks>
       </Records>
       <Records>
        <RecordsRange>
         <From>19</From>
         <To>24</To>
        </RecordsRange>
        <NumberOfBlocks>17</NumberOfBlocks>
       </Records>
       <Records>
        <RecordsRange>
         <From>25</From>
         <To>30</To>
        </RecordsRange>
        <NumberOfBlocks>22</NumberOfBlocks>
       </Records>
       <Records>
        <RecordsRange>
         <From>31</From>
         <To>36</To>
        </RecordsRange>
```

```
        <NumberOfBlocks>4</NumberOfBlocks>
       </Records>
      </RecordsPerBlock>
      <NormalRecordLength>
       <Increment>Auto</Increment>
       <Lengths>
        <LengthsRange>
         <From>97</From>
         <To>132</To>
        </LengthsRange>
        <NumberOfRecords>5</NumberOfRecords>
       </Lengths>
       <Lengths>
        <LengthsRange>
         <From>133</From>
         <To>138</To>
        </LengthsRange>
        <NumberOfRecords>8</NumberOfRecords>
       </Lengths>
       <Lengths>
        <LengthsRange>
         <From>139</From>
         <To>144</To>
        </LengthsRange>
        <NumberOfRecords>20</NumberOfRecords>
       </Lengths>
       <Lengths>
        <LengthsRange>
         <From>145</From>
         <To>150</To>
        </LengthsRange>
        <NumberOfRecords>60</NumberOfRecords>
       </Lengths>
       <Lengths>
        <LengthsRange>
         <From>151</From>
         <To>156</To>
        </LengthsRange>
        <NumberOfRecords>99</NumberOfRecords>
       </Lengths>
       <Lengths>
        <LengthsRange>
         <From>157</From>
         <To>162</To>
        </LengthsRange>
        <NumberOfRecords>86</NumberOfRecords>
       </Lengths>
       <Lengths>
        <LengthsRange>
         <From>163</From>
         <To>168</To>
        </LengthsRange>
```

```
        <NumberOfRecords>84</NumberOfRecords>
       </Lengths>
       <Lengths>
        <LengthsRange>
         <From>169</From>
         <To>174</To>
        </LengthsRange>
        <NumberOfRecords>60</NumberOfRecords>
       </Lengths>
       <Lengths>
        <LengthsRange>
         <From>175</From>
         <To>180</To>
        </LengthsRange>
        <NumberOfRecords>63</NumberOfRecords>
       </Lengths>
       <Lengths>
        <LengthsRange>
         <From>181</From>
         <To>186</To>
        </LengthsRange>
        <NumberOfRecords>70</NumberOfRecords>
       </Lengths>
       <Lengths>
        <LengthsRange>
         <From>187</From>
         <To>192</To>
        </LengthsRange>
        <NumberOfRecords>61</NumberOfRecords>
       </Lengths>
       <Lengths>
        <LengthsRange>
         <From>193</From>
         <To>198</To>
        </LengthsRange>
        <NumberOfRecords>59</NumberOfRecords>
       </Lengths>
       <Lengths>
        <LengthsRange>
         <From>199</From>
         <To>204</To>
        </LengthsRange>
        <NumberOfRecords>55</NumberOfRecords>
       </Lengths>
       <Lengths>
        <LengthsRange>
         <From>205</From>
         <To>210</To>
        </LengthsRange>
        <NumberOfRecords>63</NumberOfRecords>
       </Lengths>
       <Lengths>
```

```
        <LengthsRange>
         <From>211</From>
         <To>216</To>
        </LengthsRange>
        <NumberOfRecords>89</NumberOfRecords>
       </Lengths>
       <Lengths>
        <LengthsRange>
         <From>217</From>
         <To>222</To>
        </LengthsRange>
        <NumberOfRecords>79</NumberOfRecords>
       </Lengths>
       <Lengths>
        <LengthsRange>
         <From>223</From>
         <To>228</To>
        </LengthsRange>
        <NumberOfRecords>58</NumberOfRecords>
       </Lengths>
       <Lengths>
        <LengthsRange>
         <From>229</From>
         <To>234</To>
        </LengthsRange>
        <NumberOfRecords>41</NumberOfRecords>
       </Lengths>
       <Lengths>
        <LengthsRange>
         <From>235</From>
         <To>240</To>
        </LengthsRange>
        <NumberOfRecords>28</NumberOfRecords>
       </Lengths>
       <Lengths>
        <LengthsRange>
         <From>241</From>
         <To>246</To>
        </LengthsRange>
        <NumberOfRecords>16</NumberOfRecords>
       </Lengths>
       <Lengths>
        <LengthsRange>
         <From>247</From>
         <To>342</To>
        </LengthsRange>
        <NumberOfRecords>3</NumberOfRecords>
       </Lengths>
      </NormalRecordLength>
     </DistributionReports>
    </Device>
   </DataSummaryReport>
```

```
   </File>
   <StatisticalReport>
    <File>
     <FileNumber>10</FileNumber>
     <Duration>00:00:00.05</Duration>
     <Asso>
      <NumberOfBlocksRead>57</NumberOfBlocksRead>
      <ReadErrors>0</ReadErrors>
      <Extents>
       <FCB>2</FCB>
       <Run>2</Run>
      </Extents>
     </Asso>
     <Data>
      <NumberOfBlocksRead>44</NumberOfBlocksRead>
      <ReadErrors>0</ReadErrors>
      <Extents>
       <FCB>1</FCB>
       <Run>1</Run>
      </Extents>
     </Data>
     <Errors>
      <Parm>0</Parm>
      <Run>0</Run>
     </Errors>
     <Warnings>
      <Parm>0</Parm>
      <Run>0</Run>
     </Warnings>
    </File>
   </AdadruSummary>


   Detail:
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="adadru_detail.xsl"?>
<AdadruDetail>
 <RunTime>
  <DBID>192</DBID>
  <VER>8</VER>
  <Release>3</Release>
  <SM>8</SM>
  <Date>2013-04-17</Date>
  <Time>10:30:07</Time>
 </RunTime>
 <SpaceUsage>
  <SpaceUsageNum>1</SpaceUsageNum>
  <File>
   <Number>10</Number>
   <FileName>EMPL-192</FileName>
   <FileOptions>
    <DateLoaded>2013-03-11</DateLoaded>
```

```
    <TimeLoaded>12:01:05.21</TimeLoaded>
    <DateUpdated>n/a</DateUpdated>
    <TimeUpdated>n/a</TimeUpdated>
    <DateRestructured>2013-03-11</DateRestructured>
    <TimeRestructured>12:01:05.21</TimeRestructured>
    <ISNReusage>Yes</ISNReusage>
    <DSReusage>Yes</DSReusage>
    <MixDSDev>No</MixDSDev>
    <Spanned>No</Spanned>
    <IndexCompressed>No</IndexCompressed>
    <TwoByteIndex>No</TwoByteIndex>
    <HighestIndexLevel>3</HighestIndexLevel>
    <LOBFields>No</LOBFields>
    <LOBFile>No</LOBFile>
    <Expanded>No</Expanded>
   </FileOptions>
   <AssoDetailReport>
   </AssoDetailReport>
   <DataDetailReport>
   </DataDetailReport>
  </File>
 </SpaceUsage>
</AdadruDetail>
```

104 JCL Requirements and Examples

- Requirements to Run ADADRU JCL 752

This section describes the job control information required to run ADADRU on the z/OS operating system and shows examples of job streams.

Requirements to Run ADADRU JCL

Data Set	DD Name	Storage	More Information
Associator	DDASSOR*n* or DDASSO*nn*	disk	
Data Storage	DDDATAR*n* or DDDATA*nn*	disk	
ADARUN parameters	DDCARD	reader	*Operations*
ADADRU parameters	DDKARTE	reader	
ADADRU detail output	DDDETAIL	disk	
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADADRU messages	DDDRUCK	printer	*Messages and Codes*
ADADRU XML output summary report	DDXMLR1	tape/disk	Only necessary when the OUTPUT or ACODE parameters are specified. The XML dataset will be created as a sequential file, with record format VB (variable blocked) and a maximum record length of 512.
ADADRU XML output detailed report	DDXMLR2	tape/disk	Only necessary when the OUTPUT or ACODE parameters are specified in the same run as the CPLIST or CPEXLIST parameters. The XML dataset will be created as a sequential file, with record format VB (variable blocked) and a maximum record length of 512.

ADADRU JCL Examples

Refer to members ADADRU and ADADRUU in the JOBS data set for these examples.

General Report (Sample Member ADADRU)

```
//ADADRU    JOB
//*
//*   ADADRU:
//*      Print File Reports
//*
//DRU       EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.Vvrs.LOAD          <=== Adabas Load
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1    <=== ASSO
```

```
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1    <=== DATA
//*
//DDDRUCK   DD   SYSOUT=X   <=== Summary Reports and ADADRU Messages
//DDPRINT   DD   SYSOUT=X   <=== ADARUN Messages
//DDDETAIL  DD   SYSOUT=X   <=== Detail Reports
//DDXMLR1   DD   SYSOUT=X,DCB=(LRECL=512,RECFM=VB)
//DDXMLR2   DD   SYSOUT=X,DCB=(LRECL=512,RECFM=VB)
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADADRU,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
*
*    ADADRU Control Statements
*    Print Detailed Report for File 5
*
ADADRU NOUSERABEND,OUTPUT='XML'
*
ADADRU SPACEUSAGE FILE=5,SAMPLING=10,INCRUSAGE=1,INCRNUMREC=1
ADADRU            INCRRECLEN=6,PFILESPACE=DETAIL
//
```

Report with OUTPUT='XML,UTF8' (Sample Member ADADRUU)

```
//ADADRU    JOB
//*
//*   ADADRU:
//*      Print File Reports, XML in UTF8 Format
//*
//DRU       EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.V vrs .LOAD     <=== ADABAS LOAD
//          DD   DISP=SHR,DSN=SAG.ICS vrs .L0nn      <=== Code pages
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1    <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1    <=== DATA
//*
//DDDRUCK   DD   SYSOUT=X   <=== Summary Reports and ADADRU messages
//DDPRINT   DD   SYSOUT=X   <=== ADARUN Messages
//DDDETAIL  DD   SYSOUT=X   <=== Detail Reports
//DDXMLR1   DD   SYSOUT=X,DCB=(LRECL=512,RECFM=VB)
//DDXMLR2   DD   SYSOUT=X,DCB=(LRECL=512,RECFM=VB)
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADADRU,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
*
*    ADADRU Control Statements
*    Print Detailed Report for File 5
*
ADADRU NOUSERABEND,OUTPUT='XML,UTF8'
*
```

```
ADADRU SPACEUSAGE FILE=5,SAMPLING=10,INCRUSAGE=1,INCRNUMREC=1
ADADRU            INCRRECLEN=6,PFILESPACE=DETAIL
/*
//
```

IX ADAFRM Utility: Format Adabas Database Components

This chapter covers the following topics:

Functional Overview
Formatting Database Components
JCL Requirements and Examples

105 Functional Overview

- Statement Restrictions 758
- Formatting Operation 758

Primary Adabas direct access (DASD) data sets must be formatted using the ADAFRM utility.

These data sets include the Associator, Data Storage, and Work data sets as well as the intermediate storage (Temp, Sort, and Command/Protection/Recovery logging) data sets.

Formatting must be performed before any new data set can be used by the Adabas nucleus or an Adabas utility. After increasing a data set with the ADADBS INCREASE or ADD function, new RABNs must also be formatted.

ADAFRM also provides functions to reset existing Associator, Data Storage, or Work blocks/cylinders to binary zeros (nulls). Resetting fills the specified blocks in an existing Associator, Data Storage, or Work data set with binary zeros.

Finally, you can use this utility to clear multiple PLOG headers from the PLOG, without requiring that you reformat the entire PLOG. For more information, read about using the FROMRABN, NUMBER, and SIZE parameters together, as described in the ADAFRM ***FROMRABN***, ***NUMBER***, and ***SIZE*** parameter documentation, elsewhere in this chapter.

Statement Restrictions

More than one ADAFRM function (ASSOFRM, DATAFRM, RLOGFRM, and so on) can be performed in the same job. However, each function must be specified on separate statements. See the examples at the end of the do for more information.

Formatting Operation

Formatting with ADAFRM comprises two basic operations:

1. creating blocks (called RABNS) on the specified tracks/cylinders;
2. filling the created blocks with binary zeros (nulls).

106 Formatting Database Components

- Formatting Modes 760
- Enhanced Formatting 760
- Syntax 761
- Essential Parameter 763
- Optional Parameters 763
- Examples 765

This chapter describes the syntax, parameters, and processing of the ADAFRM utility.

Formatting Modes

There are several ADAFRM formatting modes:

1. Format a *new* data set (...FRM functions). Only the data set specified by the function name and the NUMBER parameter is accessed and formatted. The FROMRABN parameter cannot be specified when formatting a new data set.
2. Format *part of an existing* data set (ASSOFRM, DATAFRM, WORKFRM, and TEMPFRM functions). When formatting Work and Data Storage (WORKFRM and DATAFRM functions), the ADAFRM job control must also contain the Associator data sets.

   This formatting mode is used in combination with the ADADBS INCREASE function for ASSO and DATA. If a greater WORK is needed, then ADADEF NEWWORK should be used.
3. Reformat *blocks of an existing* data set (...RESET functions). This mode opens all Associator, Data Storage, and Work data sets in the database for access. The FROMRABN parameter is *must* be specified for these functions.
4. You can use this utility to clear multiple PLOG headers from the PLOG, without requiring that you reformat the entire PLOG. For more information, read about using the FROMRABN, NUMBER, and SIZE parameters together, as described in the ADAFRM ***FROMRABN***, ***NUMBER***, and ***SIZE*** parameter documentation, elsewhere in this section.

Enhanced Formatting

Effective with Adabas 8.3, data sets are formatted by the ADAFRM utility using an enhanced, quicker method. If you want to use the original (older) data set formatting technique, you can specify a new NOTENH (not enhanced) parameter.

Enhanced formatting support has changed ADAFRM processing, so that all parameter validation is done before formatting begins. This is true regardless of whether or not the NOTENH parameter is specified.

Once parameter validation is complete, ADAFRM issues the following message:

```
Parameter validation complete. Formatting started.
```

When enhanced formatting is *not* in use (parameter NOTENH specified), data sets are formatted one at a time. When enhanced formatting *is* in use, multiple data sets may be formatted simultaneously. Therefore, in the event of a failure, we recommend that you correct the failure and then rerun the ADAFRM job in its entirety.

With enhanced formatting, you can control the number of concurrent I/Os per volume per data set using the VOLIOCOUNT parameter.

Note: Users should be aware that enhanced formatting can cause significant IO activity if many large multi-volume datasets are formatted in a single ADAFRM run. In situations where this might negatively impact overall system performance while ADAFRM is running, users should consider specifying VOLIOCOUNT=1 and formatting the datasets using multiple ADAFRM job steps.

Syntax

To format the Associator (ASSO..) or Data Storage (DATA..) data set, use this syntax:

```
ADAFRM {ASSOFRM | DATAFRM}  SIZE = size
                            [DEVICE = device-type ]
                            [ { FROMRABN = { starting-rabn | NEXT } |
                                NUMBER = { dataset-number |  1 } } ]
                            [NOTENH]
                            [NOUSERABEND]
                            [VOLIOCOUNT = nnn | 4 ]
```

To format the Work (WORK..) data set, Command log (CLOG..), Protection log (PLOG..), or Sort (SORT..) data set, use this syntax:

Note: If you are using Adabas 8 or later, you can also use this syntax to clear multiple PLOG headers from the PLOG, without requiring that you reformat the entire PLOG. For more information, read about using the FROMRABN, NUMBER, and SIZE parameters together, as described in the ADAFRM ***FROMRABN***, ***NUMBER***, and ***SIZE*** parameter documentation, elsewhere in this section.

```
ADAFRM  { WORKFRM | CLOGFRM | PLOGFRM | SORTFRM }
                    SIZE = size
                    [DEVICE = device-type ]
                    [ { FROMRABN = starting-rabn  |  NUMBER = {dataset-number  |  1 } } ]
                    [NOTENH]
                    [NOUSERABEND]
                    [VOLIOCOUNT = nnn | 4 ]
```

To format the Recovery log (RLOG..) data set, use this syntax:

```
ADAFRM  RLOGFRM  SIZE = size
                 [DEVICE = device-type ]
                 [NOTENH]
                 [NOUSERABEND]
                 [VOLIOCOUNT = nnn | 4 ]
```

To format a Temp (TEMP..) data set, use this syntax:

```
ADAFRM  TEMPFRM  SIZE = size
                 [DEVICE = device-type ]
                 [FROMRABN =  starting-rabn ]
                 [NOTENH]
                 [NOUSERABEND]
                 [VOLIOCOUNT = nnn | 4 ]
```

To reformat blocks of an existing Associator, Data Storage, or Work data set, use this syntax:

```
ADAFRM  { ASSORESET | DATARESET | WORKRESET }
                    SIZE = size
                    FROMRABN = start-rabn
                    [NOTENH]
                    [NOUSERABEND]
                    [VOLIOCOUNT = nnn | 4 ]
```

Essential Parameter

SIZE: Size of Area to be Formatted

SIZE specifies the size of the area to be formatted (or reset). Blocks (a decimal value followed by a "B") or cylinders may be specified. For the RLOGFRM function, the size must be the same as that specified by the RLOGSIZE parameter on the ADARAI utility's PREPARE function. See section **Essential Parameter**.

If you are using Adabas 8 or later and you want to clear multiple PLOG headers from the PLOG without reformatting the entire PLOG, set the value of the **SIZE** parameter to "1" and specify values for both the **FROMRABN** and **NUMBER** parameters.

Optional Parameters

DEVICE: Device Type

DEVICE is the physical device type on which the area to be formatted is contained. If DEVICE is not specified, the device type specified by the ADARUN DEVICE parameter is used.

FROMRABN: Starting RABN

FROMRABN specifies the RABN at which formatting or resetting is to begin. This parameter may only be used for an existing data set; NUMBER cannot be specified in the same ADAFRM job as FROMRABN, except for PLOG data sets.

When FROMRABN is specified with a *xxxx*FRM function, formatting begins at the FROMRABN point and continues up to the *highest complete track* before the RABN computed from FROMRABN + SIZE (assuming a size specified in or converted to blocks). This means that the last track within the specified range (FROMRABN + SIZE) will be formatted *only* if all the track's RABNs are within that range.

If you are using Adabas 8 or later and you want to clear multiple PLOG headers from the PLOG without reformatting the entire PLOG, set the value of the **SIZE** parameter to "1" and specify values for both the **FROMRABN** and **NUMBER** parameters.

FROMRABN should only be used to reformat existing blocks as the last record pointer in the VTOC cannot be modified by function FROMRABN. See the **examples for ADADBS INCREASE** .

This parameter is *required* for the ASSORESET, DATARESET and WORKRESET functions. When specified with the function ASSORESET, the FROMRABN value must be greater than 30.

NOTENH: Use Unenhanced Processing

Starting with Adabas 8.3, new data sets are formatted using an enhanced and quicker method. If you want to use the original (older) data set formatting technique, specify the NOTENH parameter for the ADAFRM run.

To format a data set using the older, unenhanced technique used by releases prior to Adabas 8.3:

```
ADAFRM xxxFRM SIZE=size,NOTENH
```

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

NUMBER: Data Set Number

NUMBER selects the nonsequential command log, nonsequential protection log, Associator, Data Storage and sort data set to be formatted. The default is 1 (first data set). Values allowed for:

- the Associator (ASSO) or Data Storage (DATA) are 1 through 99;
- protection logs (PLOGs) or command logs (CLOGs) are 1 through 8;
- the recovery log (RLOG) is just 1;
- SORT is either 1 or 2; and
- WORK or TEMP is either 1 or the default.

ADAFRM ...FRM function statements cannot specify (and will not default to) a NUMBER value if other ADAFRM statements in the same job specify a FROMRABN value.

NUMBER must match the number suffix of the related data definition (DD) statement. See the tables of allowed statements and the examples in section **JCL Requirements and Examples**.

If you are using Adabas 8 or later and you want to clear multiple PLOG headers from the PLOG without reformatting the entire PLOG, set the value of the **SIZE** parameter to "1" and specify values for both the **FROMRABN** and **NUMBER** parameters.

VOLIOCOUNT: Number of Concurrent I/Os per Volume per Data Set

VOLIOCOUNT specifies the number of concurrent I/Os per volume to process for a data set. The minimum value you can specify is "1"; maximum is "256". The default value is "4". The setting of this parameter can affect the processing speed of the ADAFRM utility.

The VOLIOCOUNT parameter is ignored unless enhanced formatting is in use.

Examples

Example 1:

Format 50 cylinders for the Associator, 200 cylinders for Data Storage, 10 cylinders for Work, and 2 cylinders for the recovery log (RLOG).

```
ADAFRM   ASSOFRM SIZE=50,DEVICE=3390
ADAFRM  DATAFRM SIZE=200,DEVICE=3390
ADAFRM  WORKFRM SIZE=10,DEVICE=3390
ADAFRM  RLOGFRM SIZE=2
```

Example 2:

One cylinder for nonsequential command log data set 1, and 1 cylinder for nonsequential command log data set 2 are to be formatted.

```
ADAFRM   CLOGFRM SIZE=1,DEVICE=3390,NUMBER=1
ADAFRM  CLOGFRM SIZE=1,DEVICE=3390,NUMBER=2
```

Example 3:

The first two blocks of an existing Work data set are to be reset to binary zeros.

```
ADAFRM WORKRESET FROMRABN=1,SIZE=2B
```

Example 4:

Assuming the Data Storage data set is on a 3390 disk (10 blocks/track, 15 tracks/cylinder), 100 cylinders-starting at cylinder position 201 relative to the beginning of the data set will be formatted.

```
ADAFRM DATAFRM SIZE=100,FROMRABN=29991
```

Example 5:

In the following example, 5000 cylinders on the third DATA data set are to be formatted using the older, unenhanced technique used by releases prior to Adabas 8.3.

```
ADAFRM DATAFRM SIZE=5000,NUMBER=3,NOTENH
```

107 JCL Requirements and Examples

- Requirements to Run ADAFRM JCL 768

This section describes the job control information required to run ADAFRM on the z/OS operating system and shows an example job stream.

Note: When running with the optional Recovery Aid (RLOG), all temporary data sets must also be cataloged in the job control.

Requirements to Run ADAFRM JCL

Data Set	DD Name	Storage	More Information
Associator	DDASSORn	disk	data sets to be formatted
Data Storage	DDDATARn		
Work	DDWORKR1 DDWORKR4		
Temp	DDTEMPR1		
Sort	DDSORTRn		
Multiple command logs	DDCLOGRn		
Multiple protection logs	DDPLOGRn		
Recovery log	DDRLOGR1		
ADARUN parameters	DDCARD	reader	*Operations*
ADAFRM parameters	DDKARTE	disk	
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADAFRM messages	DDDRUCK	printer	*Messages and Codes*

ADAFRM JCL Example

Refer to ADAFRM in the JOBS data set for this example.

```
//ADAFRM    JOB
//*
//*     ALLOCATE AND FORMAT THE DATABASE COMPONENTS
//*
//*     MORE THAN ONE DATA SET CAN BE FORMATTED IN A SINGLE RUN
//*
//*
//FRM       EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD                <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=(NEW,CATLG),DSN=EXAMPLE.DByyyyy.ASSOR1, <=== ASSO
//            SPACE=(CYL,(100,100)),UNIT=DISK,VOL=SER=VOL001
//DDDATAR1  DD   DISP=(NEW,CATLG),DSN=EXAMPLE.DByyyyy.DATAR1, <=== DATA
```

```
//            SPACE=(CYL,(200,200)),UNIT=DISK,VOL=SER=VOL002
//DDWORKR1  DD   DISP=(NEW,CATLG),DSN=EXAMPLE.DByyyyy.WORKR1, <=== WORK
//            SPACE=(CYL,(40,40)),UNIT=DISK,VOL=SER=VOL003
//DDSORTR1  DD   DISP=(NEW,CATLG),DSN=EXAMPLE.DByyyyy.SORTR1, <=== SORT
//            SPACE=(CYL,(100,100)),UNIT=DISK,VOL=SER=VOL003
//DDTEMPR1  DD   DISP=(NEW,CATLG),DSN=EXAMPLE.DByyyyy.TEMPR1, <=== TEMP
//            SPACE=(CYL,(100,100)),UNIT=DISK,VOL=SER=VOL003
//DDPLOGR1  DD   DISP=(NEW,CATLG),DSN=EXAMPLE.DByyyyy.PLOGR1, <=== PLOG1
//            SPACE=(CYL,(50)),UNIT=DISK,VOL=SER=VOL003
//DDPLOGR2  DD   DISP=(NEW,CATLG),DSN=EXAMPLE.DByyyyy.PLOGR2, <=== PLOG2
//            SPACE=(CYL,(50)),UNIT=DISK,VOL=SER=VOL003
//DDCLOGR1  DD   DISP=(NEW,CATLG),DSN=EXAMPLE.DByyyyy.CLOGR1, <=== CLOG1
//            SPACE=(CYL,(50)),UNIT=DISK,VOL=SER=VOL003
//DDCLOGR2  DD   DISP=(NEW,CATLG),DSN=EXAMPLE.DByyyyy.CLOGR2, <=== CLOG2
//            SPACE=(CYL,(50)),UNIT=DISK,VOL=SER=VOL003
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADAFRM,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADAFRM ASSOFRM SIZE=100,DEVICE=dddd
ADAFRM DATAFRM SIZE=200,DEVICE=dddd
ADAFRM WORKFRM SIZE=40,DEVICE=dddd
ADAFRM SORTFRM SIZE=100,DEVICE=dddd
ADAFRM TEMPFRM SIZE=100,DEVICE=dddd
ADAFRM PLOGFRM SIZE=50,NUMBER=1,DEVICE=dddd
ADAFRM PLOGFRM SIZE=50,NUMBER=2,DEVICE=dddd
ADAFRM CLOGFRM SIZE=50,NUMBER=1,DEVICE=dddd
ADAFRM CLOGFRM SIZE=50,NUMBER=2,DEVICE=dddd
/*
```

X ADAINV Utility: Inverted List Management

This chapter covers the following topics:

Functional Overview
COUPLE: Define File-Coupling Descriptors
INVERT: Create Descriptors
JCL Requirements and Examples

108 Functional Overview

The INVERT function:

- modifies the field definition table (FDT) to indicate that the specified field is a descriptor; and
- adds all values and corresponding ISN lists for the field to the inverted list.

The newly defined descriptor may then be used in the same manner as any other descriptor. This function may also be used to create a subdescriptor, superdescriptor, phonetic descriptor, or hyperdescriptor.

The COUPLE function adds a common descriptor to two files (updates their inverted lists). Any two files may be coupled provided that a common descriptor with identical format and length definitions is present in both files. A single file may be coupled with up to 18 other files, but only one coupling relationship may exist between any two files at any one time. A file may not be coupled to itself.



Note: Only files with numbers 255 or lower can be coupled.

Changes affecting a coupled file's inverted lists are automatically made to the other file. The DBA should consider the additional overhead required to update the coupling lists when the descriptor used as the basis for coupling is updated, or when records are added to or deleted from either file. If a field that is not defined with the NU option is used as the basis for coupling and the field contains a large number of null values, a considerable amount of additional execution time and required disk space to store the coupling lists may result.

An interrupted ADAINV operation can be restarted without first having to restore the file.

The ADAINV utility requires that the nucleus be active. If the nucleus is canceled (terminated) while the ADAINV utility is running, all the work performed at that point is lost. This is because the FDT and the FCB have not been rewritten to include the new descriptor. The rewrite occurs at the very end of processing and is triggered by a special Adabas call sent by ADAINV.

109 COUPLE: Define File-Coupling Descriptors

- Essential Parameters 776
- Optional Parameters 776
- Example 778
- Temporary Space for File Coupling 778
- Associator Coupling Lists 779
- Space for Coupling Lists 780
- Space Allocation 781
- Enhanced Temp and Sort Processing 781

Use the COUPLE function to define one descriptor for each of two files to be coupled.

```
ADAINV  COUPLE  FILES = file-number1 , file-number2
                DESCRIPTOR = ' fieldname , fieldname '
                [LPB = prefetch-buffer-size ]
                [LWP = { workpool-size | 1048576 } ]
                [NOUSERABEND]
                [PASSWORD = ' password ' ]
                [SORTDEV = device-type ]
                [SORTSIZE = size ]
                [TEMPDEV = device-type ]
                [TEMPSIZE = size ]
                [TEST]
```

Essential Parameters

DESCRIPTOR: Descriptors Used as Basis for Coupling

The DESCRIPTOR parameter defines one descriptor in each file to provide the basis for coupling the files. Subdescriptors or superdescriptors may also be used, or may be defined as or derived from a multiple-value field. The descriptors specified may not be contained within a periodic group, nor be derived from a periodic group. The descriptors can have different names, but must have the same length and format definitions.

FILES: Files to Be Coupled

FILES specifies the two files to be coupled. The number of each file must be 255 or lower. The files specified may not be currently coupled to each other.

Optional Parameters

LPB: Prefetch Buffer Size

LPB specifies the size, in bytes, of the internal prefetch buffer. The maximum value is 32760 bytes. The default depends on the ADARUN LU parameter; ADAINV may also reduce a specified LPB value if the LU value is too small.

LWP: Work Pool Size

LWP specifies the size of the work pool to be used for descriptor value sorting. The value can be specified in bytes or kilobytes followed by a "K". If no value is specified, the default is 1048576 bytes (or 1024K); however, to shorten ADAINV run time for files with very long

descriptors or an unusually large number of descriptors, set LWP to a higher value. To avoid problems with the sort data set, a smaller LWP value should be specified when defining descriptors for relatively small files.

The minimum work pool size depends on the sort data set's device type:

Sort Device	Minimum LWP	Minimum LWP
	Bytes	Kilobytes
3380	139264	136K
3390	159744	156K

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "`utility` TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PASSWORD: File Password

If one or both of the files being coupled is security protected, a valid password for the file (or files) must be specified with this parameter. If both files are password-protected, both must have the same password.

SORTDEV: Sort Device Type

ADAINV uses the sort data set to sort descriptor values. The SORTDEV parameter indicates the device type to be used for the sort data set. See the ***z/OS job control information*** for specific SORTDEV considerations. Refer also to ***Enhancements to TEMP and SORT Processing in ADAINV, ADALOD and ADAULD*** for more details on specifying the SORDEV parameter.

SORTSIZE: Sort Size

SORTSIZE specifies the space available for the sort data set or data sets R1/2. The value can be either cylinders (a numeric value only) or blocks (a numeric value followed by a "B"). If blocks are specified, they should be equivalent to a full number of cylinders.Refer to the *Adabas DBA Reference* documentation for more information on estimating the sort space. Refer also to ***Enhancements to TEMP and SORT Processing in ADAINV, ADALOD and ADAULD*** for more details on specifying the SORTSIZE parameter.

TEMPDEV: Temporary Storage Device Type

ADAINV uses the temp data set to store intermediate data. The TEMPDEV parameter indicates the device type to be used for this data set. Refer to ***Enhancements to TEMP and SORT Processing in ADAINV, ADALOD and ADAULD*** for more details on specifying the TEMPDEV parameter.

TEMPSIZE: Temporary Storage Size

TEMPSIZE defines the space available for the temp data set. The value may be in cylinders (a numeric value only) or blocks (a numeric value followed by a "B"). Refer to ***Enhancements to TEMP and SORT Processing in ADAINV, ADALOD and ADAULD*** for more details on specifying the TEMPSIZE parameter.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

Example

```
ADAINV COUPLE
FILES=3,4,DESCRIPTOR='AA,BB'
```

Files 3 and 4 are to be coupled. Descriptor AA from file 3 and descriptor BB from file 4 are to be used as the basis for the coupling.

Temporary Space for File Coupling

An intermediate data set is generated for *each* of the files being coupled.

An entry is written to the data set for each record contained in the file. Each entry contains the ISN of the record (3 or 4 bytes, depending on the ISNSIZE defined for the files) and the value (in compressed form) of the descriptor being used as the basis for the coupling. If the descriptor is defined with the NU option, no entries are written for records in which the descriptor contains a null value. If the descriptor is a multiple-value field, an entry is written for each different value.

The space required for *each* of the intermediate data sets is a function of the number of records contained in each Adabas file, and the length and the number of the different values present for the coupling descriptor in each record.

Use the following equation to determine the space needed for an intermediate data set:

```
SP = RECS x UV x (ISNSIZE + (AVLEN x 4)
```

where

SP	intermediate data set space required (in bytes).
RECS	number of records contained in the coupled file.
UV	average number of unique values per record for the descriptor. If the descriptor is not defined with the NU option, UV is equal to or less than 1. If the descriptor is defined with the NU option, UV is equal to the average number of values per record minus the percentage of records that contain a null value. For example, if the average number of values per record is 1 and 20 percent of the values are null, UV is equal to 1 - 0.2 = 0.8.
ISNSIZE	length of ISNs in the file (3 or 4 bytes).
AVLEN	average length (after compression) of each value for the descriptor.

Example: Calculating Intermediate Space Requirements for File Coupling

The file being coupled has 3-byte ISNs and contains 50,000 records. The descriptor being used as the basis for coupling contains 1 value per record (with no null values) and has an average value length of 5 bytes.

```
SP = 50,000 x  1 x  (3 + (5 + 1))
SP = 50,000 x  9
SP = 450,000 bytes
```

Associator Coupling Lists

ADAINV matches the two lists, sorts each resulting list, and writes each list to the Associator coupling lists.

The temp data set stores the matched (coupled) ISNs for each file. An entry is written to the temp data set for each match found. The entry contains the ISN of each record containing a matching value.

ADAINV sorts the entries stored on the temp data set using the sort area and writes the sorted entries to the Associator coupling lists for file A. The same process is then repeated for file B.

The temp area size requirement depends on the number of matching values in the two files for the descriptor used to couple the files. Each match requires 6 or 8 bytes, depending on the ISNSIZE defined for the files.

The sort area generally requires twice the amount of space as that needed for the temp area.

File coupling is bidirectional rather than hierarchical in that two coupling lists are created with each list containing the ISNs which are coupled to the other file.

Example: Coupling Lists

Assume that 2 files containing the descriptors AA and BB, respectively, are to be coupled. The values for the first five records of each file are as follows:

File A		File B	
ISN	Field AA value	ISN	Field BB value
1	20	1	18
2	25	2	40
3	27	3	25
4	30	4	20
5	40	5	20

If the two files were coupled using AA and BB as the basis for the coupling, the resulting coupling lists would be:

File A			File B		
ISN in FILE B*	COUNT	COUPLED ISNs	ISN in FILE A*	COUNT	COUPLED ISNs
2	1	5	1	2	4,5
3	1	2	2	1	3
4	1	1	5	1	2
5	1	1			

** Internally, Adabas uses this field like a descriptor to determine the number and the ISNs of the coupled records.*

Space for Coupling Lists

The total space requirement for the coupling lists depends upon the number of common values that exist between the two descriptors used as the basis for the coupling.

The space requirement for *each common value* may be estimated as follows:

```
SP = 4a + 4b + 6ab
```

where

SP	space requirement for one common value (in bytes);
a	number of records in file A containing the common value;
b	number of records in file B containing the common value.

The total coupling list requirement is the sum of the space requirements of each common value.

Using sample files A and B as previously defined, space requirements per common value are

Common Value	Space Requirements
20	SP = 4(1) + 4(2) + 6(1 • 2) = 24 bytes
25	SP = 4(1) + 4(1) + 6(1 • 1) = 14 bytes
40	SP = 4(1) + 4(1) + 6(1 • 1) = 14 bytes

Total space required = 24 + 14 + 14 = 52 bytes

Example: Coupling List Space Requirements

Assume that 2 files are being coupled on the field ID. The values for ID are unique within each file. There are 5,000 common values in the coupled files.

Common Value	Space Requirements
n	SP = 4(1) + 4(1) + 6(1) SP = 14 bytes for one common value

There are 5,000 common values, each of which requires 14 bytes. The total space requirement for the coupling lists is 70,000 bytes.

Space Allocation

The coupling lists constructed by ADAINV are contained within the normal (NI) and upper (UI) index for each file being coupled. If the NI or UI component's logical extents currently allocated to the file are used up during ADAINV execution, ADAINV attempts to allocate an additional extent to the component. The size of the extent allocated is equal to 25 percent of the current total size of all logical extents currently assigned to the component. If insufficient space is available or if the maximum number of allocated extents has been reached for the component, ADAINV terminates with an error message.

Enhanced Temp and Sort Processing

The TEMPSIZE and SORTSIZE parameters are now optional. When omitted, various enhancements to TEMP/SORT processing are available to the user. See *Enhancements to TEMP and SORT Processing in ADAINV, ADALOD and ADAULD* for details.

110 INVERT: Create Descriptors

- Essential Parameters 784
- Optional Parameters and Subparameters 784
- Space Allocation for the INVERT Function 787
- Enhanced Temp and Sort Processing 787
- Examples 788

The INVERT function creates descriptors, subdescriptors, superdescriptors, hyperdescriptors, phonetic descriptors or collation descriptors for existing fields in a file. Several descriptors may be created in a single ADAINV INVERT run, but only for a single file.

```
ADAINV  INVERT  FILES = file-num
                [FIELD = ' field-name [ , option ]... ' ] ...
                [COLDE = ' num , name [ , UQ [, XI ] ] = parent-field ' ]
                [HYPDE = ' num , name , length , format [ , option ]... = parent-field , ...' ]
                [PHONDE = ' name ( field-name )' ]
                [SUBDE = ' name [, UQ [, XI ] ] = parent-field ( begin , end )' ]
                [SUPDE = ' name [, UQ [, XI ] ] = { parent-field ( begin , end ) } , ...' ]
                [CODE = cipher-code ]
                [LPB = prefetch-buffer-size ]
                [LWP = { workpool-size | 1048576 } ]
                [NOUSERABEND]
                [PASSWORD = ' password ' ]
                [SORTDEV = device-type ]
                [SORTSIZE = size ]
                [TEMPDEV = device-type ]
                [TEMPSIZE = size ]
                [TEST]
```

Essential Parameters

FILE: File Number

FILE specifies the file in which the descriptor(s) to be created is contained.

Optional Parameters and Subparameters

CODE: Cipher Code

If the file specified with the FILE parameter is ciphered, an appropriate cipher code must be supplied using the CODE parameter.

FIELD/ COLDE/ HYPDE/ PHONDE/ SUBDE/ SUPDE: Define Descriptor(s)

These parameters may be used to define various types of descriptors. You must specify at least one descriptor definition for the file specified; you may specify more than one descriptor or type of descriptor.

Use the FIELD parameter to define one or more fields as descriptors; use the COLDE parameter for a collation descriptor; HYPDE parameter for a hyperdescriptor; PHONDE for a phonetic descriptor; SUBDE for a subdescriptor; and SUPDE for a superdescriptor.

If provided, a FIELD specification must come before any collation descriptor, hyper-, super-, sub-, or phonetic descriptor specification.

FIELD specifies an existing field (or fields) to be inverted. The field may be an elementary or multiple-value field and may be contained within a periodic group (unless the field is defined with the FI option).

If the descriptor is to be unique, specify "UQ" following the field name. If the uniqueness of the descriptor is to be determined with the index (occurrence number) excluded, specify "XI" as well.



Note: For Adabas expanded files, ADAINV can only detect unique descriptor violations within the specified component file. If an identical value exists for a unique descriptor in one of the other component files, ADAINV cannot detect it. You must therefore ensure that unique descriptor values remain unique throughout an expanded file.

Although multiple fields can be specified for inversion using the FIELD parameter, only one collation descriptor, hyper-, sub-, super-, or phonetic descriptor is defined per instance of its parameter. Multiple instances of the parameters are allowed per execution of ADAINV.

When inverting a sub- or superfield, the respective SUBDE or SUPDE parameter must specify the same parent fields that were specified when the field was created; otherwise, an error occurs. Begin and end values are taken from the original field definitions.

If a parent field with the NU option is specified, no entries are made in the inverted list for those records containing a null value for the field. For superdescriptors, this is true regardless of the presence or absence of values for other descriptor elements. For hyperdescriptors, no input parameter element for a null parent field is provided to the hyperdescriptor exit, but input parameter elements for other parent fields may still be provided to the exit. (See Null Value Option for details).

If a parent field is not initialized and logically falls past the end of the physical record, the inverted list entry for that record is not generated for performance reasons. To generate the inverted list entry in this case, it is necessary to unload short, decompress, and reload the file; or use an application program to initialize the field for each record of the file.

For detailed information about the individual descriptor syntax, subparameter values, and coding, read ***Field Definition Statements*** in the description of the ADACMP utility, elsewhere in this guide.

LPB: Prefetch Buffer Size

LPB specifies the size, in bytes, of the internal prefetch buffer. The maximum value is 32760 bytes. The default depends on the ADARUN LU parameter; ADAINV may also reduce a specified LPB value if the LU value is too small.

LWP: Work Pool Size

LWP specifies the size of the work pool to be used for descriptor value sorting. The value can be specified in bytes or kilobytes followed by a "K". If no value is specified, the default is 1048576 bytes (or 1024K); however, to shorten ADAINV run time for files with very long descriptors or an unusually large number of descriptors, set LWP to a higher value. To avoid problems with the Sort data set, a smaller LWP value should be specified when defining descriptors for relatively small files.

The minimum work pool size depends on the Sort data set's device type:

Sort Device	Minimum LWP	Minimum LWP
	Bytes	Kilobytes
3380	139264	136K
3390	159744	156K

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "`utility` TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PASSWORD: File Password

If the file specified with the FILE parameter is security protected, the file's password must be supplied using this parameter.

SORTDEV: Sort Device Type

ADAINV uses the sort data set to sort descriptor values. The SORTDEV parameter indicates the device type to be used for the sort data set. See the job control information at the end of this section for specific z/OS SORTDEV considerations. Refer also to ***Enhancements to TEMP and SORT Processing in ADAINV, ADALOD and ADAULD*** for more details on specifying the SORTDEV parameter.

SORTSIZE: Sort Size

SORTSIZE specifies the space available for the sort data set or data sets R1/2. The value can be either cylinders (a numeric value only) or blocks (a numeric value followed by a "B"). If blocks are specified, they should be equivalent to a full number of cylinders. Refer to the

Adabas DBA Reference documentation for more information on estimating the sort space. Refer also to ***Enhancements to TEMP and SORT Processing in ADAINV, ADALOD and ADAULD*** for more details on specifying the SORTSIZE parameter.

TEMPDEV: Temporary Storage Device Type

ADAINV uses the temp data set to store intermediate data. The TEMPDEV parameter indicates the device type to be used for this data set. Refer to ***Enhancements to TEMP and SORT Processing in ADAINV, ADALOD and ADAULD*** for more details on specifying the TEMPDEV parameter.

TEMPSIZE: Temporary Storage Size

TEMPSIZE defines the space available for the temp data set. The value may be in cylinders (a numeric value only) or blocks (a numeric value followed by a "B"). Refer to***Enhancements to TEMP and SORT Processing in ADAINV, ADALOD and ADAULD*** for more details on specifying the TEMPSIZE parameter.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

Space Allocation for the INVERT Function

The values for the field being inverted and the ISNs of the records containing the values are written to the inverted list (normal and upper indexes).

If either the normal or upper index logical extent is exhausted during ADAINV execution, ADAINV allocates an additional extent. The size of the extent allocated is equal to 25 percent of the current total size of all the normal index extents currently allocated to the file.

If sufficient space is not available for the new extent or if the maximum number of allocated extents has been reached, ADAINV terminates with an error message.

Enhanced Temp and Sort Processing

The TEMPSIZE and SORTSIZE parameters are now optional. When omitted, various enhancements to TEMP/SORT processing are available to the user. See ***Enhancements to TEMP and SORT Processing in ADAINV, ADALOD and ADAULD*** for details.

Examples

Example 1:

```
ADAINV INVERT
FILE=3,FIELD='AR',TEMPSIZE=10,SORTSIZE=5
```

Field AR in file 3 is to be made a descriptor.

Example 2:

```
ADAINV INVERT FILE=5,SUBDE='SA=AA(1,4)'
ADAINV            TEMPSIZE=6,SORTSIZE=3
```

Subdescriptor SA is to be created using field AA (positions 1-4) in file 5 as the parent field.

Example 3:

```
ADAINV INVERT FILE=6,SUPDE='SB=AA(1,4),AB(1,1)'
ADAINV            TEMPSIZE=5,SORTSIZE=3
```

Superdescriptor SB is to be created using fields AA (positions 1-4) and AB (position 1) in file 6.

Example 4:

```
ADAINV INVERT FILE=1,PHONDE='XX(AA)'
ADAINV            TEMPSIZE=5,SORTSIZE=3
```

A phonetic descriptor XX is created using field AA as the source field.

Example 5:

```
ADAINV INVERT FILE=6,COLDE='1,Y1=AA'
ADAINV            TEMPSIZE=5,SORTSIZE=4
```

Collation descriptor CDX=01 named Y1 is created using AA as the source field.

111 JCL Requirements and Examples

- Collation with User Exit 790
- Requirements to Run ADAINV JCL 790

This section describes the job control information required to run ADAINV on the z/OS operating system and shows examples of job streams.

Collation with User Exit

If a collation user exit is to be used during ADAINV execution, the ADARUN CDXnn parameter must be specified for the utility run.

Used in conjunction with the universal encoding support (UES), the format of the collation descriptor user exit parameter is



where

nn	is the number of the collation descriptor exit, a two-digit decimal integer in the range 01-08 inclusive.
exit-name	is the name of the user routine that gets control at the collation descriptor exit; the name can be up to 8 characters long.

Only one program may be specified for each collation descriptor exit. Up to 8 collation descriptor exits may be specified (in any order). See the *Adabas DBA Reference* documentation for more information.

Requirements to Run ADAINV JCL

Data Set	DD Name	Storage	More Information
Associator	DDASSORn	disk	
Intermediate storage	DDTEMPR1	disk	
Sort area	DDSORTR1	disk	
Sort area	DDSORTR2	disk	Optional additional SORT area (see Note).
Recovery log (RLOG)	DDRLOGR1	disk	Required when using the recovery log option
ADARUN parameters	DDCARD	reader	*Operations*
ADAINV parameters	DDKARTE	reader	
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADAINV messages	DDDRUCK	printer	*Messages and Codes*

Note: A second sort data set can optionally be defined using DD DDSORTR2. If DDSORTR2 is specified, both sort data sets must be on the same device type (SORTDEV parameter). If the SORTSIZE parameter is specified, both sort data must be exactly half the size specified in the SORTSIZE parameter. If the SORTSIZE parameter is not specified, and DDSORTR1 and DDSORTR2 fill up, an attempt to allocate additional space at the end of DDSORTR2 will be made.

*

ADAINV JCL Examples

Couple Files

Refer to ADAINVCO in the JOBS data set for this example.

```
//ADAINVCO  JOB
//*
//*    ADAINV:  COUPLE FILES
//*
//INV       EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD      <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <===== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <===== DATA
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <===== WORK
//DDTEMPR1  DD   DISP=OLD,DSN=EXAMPLE.DByyyyy.TEMPR1 <===== TEMP
//DDSORTR1  DD   DISP=OLD,DSN=EXAMPLE.DByyyyy.SORTR1 <===== SORT
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADAINV,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADAINV COUPLE FILE=2,3,DESCRIPTOR='BB,BB'
ADAINV        TEMPSIZE=100,SORTSIZE=100
/*
```

Invert File

Refer to ADAINV in the JOBS data set for this example.

```
//ADAINVDE  JOB
//*
//*     ADAINV:  INVERT A FIELD TO A DE
//*
//INV       EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD      <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <===== ASSO
//DDTEMPR1  DD   DISP=OLD,DSN=EXAMPLE.DByyyyy.TEMPR1 <===== TEMP
//DDSORTR1  DD   DISP=OLD,DSN=EXAMPLE.DByyyyy.SORTR1 <===== SORT
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADAINV,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADAINV INVERT FILE=1
ADAINV FIELD='AC'
ADAINV SUPDE='S1,UQ=AA(1,3),AD(2,4)'
ADAINV TEMPSIZE=100,SORTSIZE=100
/*
```

Invert File Using Unformatted Temporary Data Sets for TEMP and SORT

Refer to ADAINVTM in the JOBS data set for this example.

```
//ADAINVDE  JOB
//*
//*     ADAINV:  INVERT A FIELD TO A DE
//*
//INV       EXEC PGM=ADARUN
//STEPLIB  DD  DISP=SHR,DSN=ADABAS.ADAvrs.LOAD           <=== LOAD
//*
//DDASSOR1 DD  DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1       <=== ASSO
//DDSORTR1 DD  DISP=NEW,UNIT=SYSDA,SPACE=(CYL,(100,80)),
//             DSN=&&SORTR1                              <=== SORT
//DDTEMPR1 DD  DISP=NEW,UNIT=SYSDA,SPACE=(CYL,(100,80)),
//             DSN=&&TEMPR1                              <=== TEMP
//DDDRUCK  DD  SYSOUT=*
//DDPRINT  DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//DDCARD   DD  *
ADARUN PROG=ADAINV,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADAINV INVERT FILE=1
ADAINV FIELD='AC'
ADAINV SUPDE='S1,UQ=AA(1,3),AD(2,4)'
/*
```

The TEMP and SORT data sets each have an initial allocation of 100 cylinders. If either TEMP or SORT fills up during processing, additional extents of 80 cylinders will be allocated and the job will continue processing.

XI ADALOD Utility: File Loader

This chapter covers the following topics:

Functional Overview
LOAD: Load a File
UPDATE : ADD/Delete Records
Loader Storage Requirements and Use
Temp Data Set Space Usage
ADALOD Space/Statistics Report
JCL Requirements and Examples

112 Functional Overview

The **ADALOD LOAD function** loads a file into the database. Compressed records produced by the ADACMP or ADAULD utility may be used as input.

ADALOD loads each compressed record into Data Storage, builds the address converter for the file, and enters the field definitions for the file into the field definition table (FDT). ADALOD also extracts the values for all descriptors in the file together with the ISNs of all records in which the value is present, to an intermediate data set. This data set is then sorted into value/ISN sequence and then entered into the Associator inverted lists.

The **ADALOD UPDATE function** is used to add or delete a large number of records to/from an Adabas file. The UPDATE function requires considerably less processing time than the repetitive execution of the Adabas adddelete record commands. Records to be added may be the compressed records produced by the ADACMP or ADAULD utility. The ISNs of records to be deleted can be provided either in an input data set or by using control statements.

Records may be added and other records deleted during a single execution of ADALOD.



Note: You can load files from input data sets that were created by ADAULD or ADACMP utilities running under Adabas 5.1 - 5.3, 6.1, 6.2, 7.1, 7.2, or 7.4. In addition, you can load Adabas 8 (or later) files into Adabas databases released after 5.3, as long as the file to be loaded does not use any of the new features that were introduced after the version used to load the file.

113 LOAD: Load a File

- Essential Parameters 802
- Optional Parameters and Subparameters 804
- Examples 820
- LOAD Data and Space Requirements 821
- Loading Expanded Files 825
- Loading Multiclient Files 827
- Enhanced Temp and Sort Processing 828

Use the LOAD function to load a file into a database.

```
ADALOD LOAD  FILE = file-number [ , filetype ]
             DSSIZE = size
             MAXISN = max-number-of-records [MAXISN2 = max-number-of-secondary-spanned-records ]
             [ACRABN = starting-rabn ] [AC2RABN = starting-rabn ]
             [ADAMFILE ADAMDE = { field | ISN } [ADAMOFLOW = size ] [ADAMPARM = { number | 0 } ] ]
             [ALLOCATION = { FORCE | NOFORCE } ]
             [ANCHOR = file-number MINISN = lowest-allocated-isn , NOACEXTENSION ]
             [ASSOPFAC = { padding-factor | 10 } ] [ASSOVOLUME = ' Associator-extent-volume ' ]
             [{BASEFILE | LOBFILE} = file-number ]
             [DATAFRM = { YES | NO }]
             [DATAPFAC = { padding-factor | 10 } ] [DATAVOLUME = ' Data-Storage-extent-volume ' ]
             [DSDEV = device-type ] [DSRABN = start-rabn ] [DSREUSE = { YES | NO }]
             [ETID = owner-id ]
             [IGNFDT]
             [INDEXCOMPRESSION = { YES | NO }]
             [ISNREUSE = { YES | NO ] [ISNSIZE = { 3 | 4 } ]
             [LIP = { isn-pool-size | 2000 } ]
             [LOWNERID = { owner-id-length | 0 } ]
             [LWP = { work-pool-size | 1048576 } ]
             [MAXDS = max-DS-secondary-allocation ]
             [MAXNI = max-NI-secondary-allocation ]
             [MAXRECL = max-compressed-record-length ]
             [MAXUI= max-UI-secondary-allocation ]
             [MINISN = { lowest-allocated-isn | 1 } ]
             [MIXDSDEV]
             [NAME = { name | TESTFILE } ]
             [NIRABN = start-rabn ] [NISIZE = size ]
             [NOACEXTENSION]
             [NOUSERABEND]
             [NUMREC = max-number-of-records-to-load ]
             [PGMREFRESH = { YES | NO } ]
             [READONLY = { YES | NO } ]
             [REPLICATOR]
             [RESTART]
               RPLTARGETID = ' reptor-target-id '
                   [RPLDSBI]
                   [RPLERRORDEACTFILE = { NO | YES} ]
                   [RPLINITERROR = { FAIL | CONTINUE} ]
                   [RPLKEY = ' primary-key-for-replication ' ]
                   [RPLLOAD = { YES | FILE | NO } ]
             [RPLUPDATEONLY = { YES | NO } ]
             [SKIPREC = { number | 0 } ]
             [SLOG]
             [SORTDEV = device-type ]
             [SORTSIZE = size ]
             [SYFMAXUV = nn ]
             [TEMPDEV = device-type ]
             [TEMPSIZE = size ]
             [TEST]
             [UIRABN = start-rabn ] [UISIZE= size ]
             [UQDE = descriptor-list ]
             [USERISN = { YES | NO } ]
             [VERSION = { 4 | 5 | 6 | 7 } ]
```

Essential Parameters

DSSIZE: Extent Size for Data Storage

DSSIZE is the count of blocks or cylinders to be assigned to the file's Data Storage logical extent. This value must be specified. Block values must be followed by a "B" (for example, "5000B").

The number can be taken directly from the Space Requirements report produced by the ADACMP utility. If the specified extent size exceeds the largest free size, ADALOD allocates as many file extents as necessary to satisfy the request.

If a small number of records is being loaded now and a larger number of records is to be added later, the ADACMP report value should be increased in proportion to the total records to be added; otherwise, the space allocation for Data Storage may not be large enough to accommodate the records to be added. The file must then be unloaded and reloaded (or reordered) to increase the Data Storage space allocation. For more information, see the section *LOAD File Space Allocation* in the ***LOAD Data and Space Requirements*** section.

FILE: File Number, File Type

FILE specifies the Adabas file number and file type to be assigned to the file.

The number specified must not be currently assigned to another file in the database, unless that file was first deleted using the KEEPFDT parameter (see ADADBS DELETE function). The number must not be greater than the maximum file number defined for the database; for a checkpoint, security, or trigger, or system file, the number must be 5000 or lower. File numbers may be assigned in any sequence.

The file type is optional and is used to indicate that the file is an Adabas system file or an Adabas LOB file. One of the following keywords may be specified:

CHECKPOINT	Adabas checkpoint system file
LOB	Adabas *LOB file*
SECURITY	Adabas security system file
SYSFILE	Adabas system file
TRIGGER	Adabas trigger system file

Notes:

1. An existing checkpoint system file created using the ADADEF utility cannot be overwritten.
2. The security system file is required if Adabas Security is to be used.
3. In an Adabas Transaction Manager (ATM) database, SYSFILE numbers 5 and 6 are reserved for the ATM nucleus. For Adabas version 7.1, these file numbers cannot be changed. The file numbers are more flexible in subsequent versions of Adabas.

4. Use the following parameters to load the ATM system files on an ATM database (ADARUN DTP=TM): `ADALOD LOAD FILE=5,SYSFILE ,ADALOD LOAD FILE=6,SYSFILE`
5. If CHECKPOINT, SECURITY, or TRIGGER is specified, the contents of DDEBAND are ignored.
6. No DDEBAND data set need be supplied if you are loading an empty LOB file.
7. CHECKPOINT, SECURITY, or SYSFILE files can be deleted only by the ADADBS DELETE function running as the only Adabas user; deleting a system file terminates Adabas when deletion is completed.
8. Adabas allows a maximum of eight (8) system files.
9. If a *LOB file* is being loaded, the parameters ADAMFILE, ANCHOR, LOWNERID, NUMREC, SKIPREC, and UQDE cannot be specified in the ADALOD LOAD run.

MAXISN: Maximum ISN Count

The MAXISN parameter is required. Specify the maximum number of ISN mappings in the address converter (AC). ADALOD determines the number of ISN mappings to allow space for in the AC using the calculation:

```
(MAXISN - MINISN) + 1
```

There is no default value.

The MAXISN and MINISN values you specify are used to calculate the initial number of AC blocks to allocate during the ADALOD execution. Depending on the size of RABNs in the database (which is determined by the ADADEF DEFINE parameter RABNSIZE), each RABN requires 3 or 4 bytes in the AC. In addition, the block size of each AC block depends upon the device type of the Associator. So the number of AC blocks that should be allocated is affected by the number of ISN mappings allowed, the RABN size, and the block size of the Associator device.

To calculate the number of AC blocks that must be allocated, ADALOD uses the following calculation and rounds up to the nearest integer:

```
(#-of-ISN-mappings X RABN-size)/device-blocksize
```

For example, assume the RABN size for the database is set to "3" (the ADADEF DEFINE RABNSIZE parameter) and that the block size of the device on which the Associator resides is 2544 bytes. If MAXISN=1000 and MINISN=1, ADALOD calculates that the actual number of ISNs to be mapped as (1000 -1)+1 (`MAXISN - MINISN +1`), or 1000. It then multiplies 1000 by three (the RABN size), to get 3000 bytes. Finally, it divides 3000 by 2544 (the block size of the device), resulting in a value of roughly 1.18, which it rounds up to two. So ADALOD determines that two AC blocks should be allocated for this ADALOD run. (Note that on the corresponding ADAREP report, the "MAX-ISN Expected" value would not be listed as 1000; instead it is listed as the actual number of ISNs that would fit into two AC blocks – in this case about 1694.)

If more than *(MAXISN - MINISN) + 1* records are to be loaded, and if NOACEXTENSION is *not* specified, ADALOD increases the MAXISN value and allocates an additional AC extent.

MAXISN does not specify the maximum number of records that can be loaded into the file. The maximum number of records that Adabas permits in a file depends on the ISNSIZE parameter, which specifies whether ISNs in the file are 3 bytes or 4 bytes long. (If ISNSIZE=3, Adabas permits up to 16,777,215 records. If ISNSIZE=4, Adabas permits up to 4,294,967,294 records.)

Optional Parameters and Subparameters

ACRABN/ AC2RABN/DSRABN/ NIRABN/ UIRABN: Starting RABN

Causes space allocation for the address converter (ACRABN), secondary address converter (AC2RABN), Data Storage (DSRABN), the normal index (NIRABN), or the upper index (UIRABN) to begin at the specified RABN.

ADAMFILE: File to Be Loaded with ADAM Option

ADAMFILE specifies the file is to be loaded using the ADAM option.

If this parameter is specified, the Data Storage RABN for each input record is calculated using a randomizing algorithm, the result of which is based on the value of the ADAM descriptor in each record. See the ADAMER utility description for additional information about using the ADAM option. If ADAMFILE is specified, ADAMDE must also be specified.



Note: If a *LOB file* is being loaded (read about the **FILE parameter**), the ADAMFILE parameter cannot be specified in the ADALOD LOAD run.

ADAMDE: ADAM Key

ADAMDE specifies the field to be used as the ADAM key.

The ADAM descriptor must be defined in the field definition table (FDT). The descriptor must have been defined with the UQ option, and *cannot*

- be a sub-, super-, hyper-, collation, or phonetic descriptor;
- be a multiple-value field;
- be a field within a periodic group;
- be variable length;
- specify the null suppression (NU) option.

If the ISN of the record is to be used as the ADAM key, ADAMDE=ISN must be specified.

This parameter must be specified when the ADAM option has been selected for the file being loaded with the ADAMFILE parameter.

ADAMOFLOW: Overflow Area Size for ADAM File

ADAMOFLOW is the size of the Data Storage area to be used for ADAM file overflow. The ADAMOFLOW value applies only if the ADAM option has been selected for the file being loaded (see ADAMFILE parameter).

ADALOD will choose a prime number which is less than DSSIZE minus ADAMOFLOW (in blocks). This prime number is used to compute the Data Storage RABN for each record. If a record does not fit into the block with the computed RABN, it is written to the next free RABN in the overflow area.

ADAMPARM: Bit Truncation for ADAM File

ADAMPARM specifies the number of bits to be truncated from the ADAM descriptor value before it is used as input to the ADAM randomizing algorithm. A value in the range 1-255 may be specified. If this parameter is omitted, a value of 0 bits (no truncation) will be used.

This parameter achieves a type of record clustering, with nearly equal ADAM keys. ADAMPARM can be specified only when the ADAMFILE parameter has also been specified.

ALLOCATION: Action to Follow File Extent Allocation Failure

ALLOCATION specifies the action to be taken if file extent allocations cannot be obtained according to the placement parameters ACRABN, DSRABN, NIRABN, or UIRABN.

By default (that is, ALLOCATION=FORCE), the utility terminates with error if any file extent allocation cannot be met according to RABN placement parameters.

If ALLOCATION=NOFORCE is specified and any allocation with placement parameters fails, the utility retries the allocation without the placement parameter.

If insufficient space can be obtained according to the placement parameters DSRABN, NIRABN, or UIRABN, only the first extent will be made there and the rest will be made elsewhere. But if the placement parameter ACRABN is used with ALLOCATION=FORCE, the complete space has to be available there; otherwise, the utility terminates with an error.

ANCHOR: Expanded Component/ Anchor File

ANCHOR defines the base (anchor) file for either an existing or a new expanded file. If the file defined by ANCHOR is the same as that defined by the FILE parameter, the loaded file becomes the physical base (anchor) file for a new expanded logical file. Otherwise, the FILE file becomes a new component of the expanded file defined by ANCHOR.

If ANCHOR specifies a file that is not part of an expanded file, the LOAD operation defines this file and the file specified by the FILE parameter as a new expanded file. It also sets the NOACEXTENSION indicator for the file specified by ANCHOR.

If ANCHOR specifies the anchor file of an already existing expanded file, the LOAD operation adds the file specified by FILE to the expanded file.

Note: When loading a new file to an existing expanded file, you must have exclusive update use of the anchor file as well as the file being added. This can be achieved by locking the anchor file for utility use.

Both the file specified by ANCHOR and the file specified by FILE must have the same field definition table (FDT) structure. The maximum record length (MAXRECL parameter) and any file security definitions must also be the same.

If ANCHOR is specified, the MINISN and NOACEXTENSION parameters must also be specified. Coupled files or multiclient files cannot be part of expanded files.

Note: If a *LOB file* is being loaded (read about the **FILE parameter**), the ANCHOR parameter cannot be specified in the ADALOD LOAD run.

ASSOPFAC: Associator Padding Factor

ASSOPFAC defines the padding factor to be used for each Associator block. If not specified, the default padding factor is 10.

The value specified represents the percentage of each Associator block (padding area) that is not to be used during the loading process. The padding area is reserved for use when additional entries must be added to the block for new descriptor values or new ISNs for existing values, thereby avoiding the overhead caused by relocating overflow entries into another block.

A value in the range 1-90 may be specified. The number of bytes contained in an Associator block, minus the number of bytes reserved for padding, must be larger than the largest descriptor value contained in the file, plus 10 bytes.

A small padding factor (1-10) should be specified if little or no descriptor updating is planned. A larger padding factor (10-50) should be specified if a large amount of updating including addition of new descriptor values (or new ISNs) is planned.

ASSOVOLUME: Associator Extent Volume

Note: The value for ASSOVOLUME must be enclosed in apostrophes.

ASSOVOLUME specifies the volume on which the file's Associator space (that is, the AC, NI, and UI extents) is to be allocated. If the requested number of blocks cannot be found on the specified volume, ADALOD retries the allocation while disregarding the ASSOVOLUME parameter.

Note: If there are five or more blocks of unused ASSO space on the specified volume, ADALOD allocates these blocks; if this is not enough space, it ends with ERROR-060. If there are no free blocks remaining on the specified volume, ADALOD tries to allocate space on another volume.

If ACRABN, UIRABN, or NIRABN is specified, ADALOD ignores the ASSOVOLUME value when allocating the corresponding extent type. If ASSOVOLUME is not specified, the file's Associator space is allocated according to ADALOD's default allocation rules.

BASEFILE: Base File Number

BASEFILE specifies the file number of the *base file* associated with the *LOB file* you are loading. This parameter is only used when loading LOB files.

For more information, read *Large Object (LB) Files and Fields*, in *Adabas DBA Tasks Manual*.

DATAFRM: Overwrite ADAM Data Storage

DATAFRM controls overwriting of an ADAM file's Data Storage during loading. DATAFRM=YES (the default) forces ADALOD to reformat the Data Storage area when the file is loaded; DATAFRM=NO prevents reformatting, and is recommended when loading relatively few records because the load operation may run significantly faster. Specifying NO, however, assumes that the Data Storage area was previously formatted with the ADAFRM utility specifying FROMRABN.



Caution: Specify DATAFRM=NO with care. If the primary Data Storage area was incorrectly formatted, later file processing could cause errors and unpredictable results.

DATAPFAC: Data Storage Padding Factor

DATAPFAC is the padding factor to be used for each Data Storage physical block. A percentage value in the range 1-90 may be specified. If not specified here, the default padding factor is 10.

A small padding factor (1-10) should be specified if little or no record expansion is expected. A larger padding factor (10-50) should be specified if a large amount of updating is planned that will expand the logical records.

The percentage value specified represents the portion of each Data Storage block (padding area) to be reserved during the loading process for later record expansion. The padding area is used when any given logical record within the block requires additional space as the result of record updating, thereby avoiding the overhead that would be needed to relocate the record to another block.

Since records loaded into a file can be different lengths, the padding factor cannot be exactly the percentage specified in each block. Adabas balances the size of the padding area for the different record lengths to the extent that at least 50 bytes remain in a block.

Example:

A block size is 1000 bytes; the padding factor is 10%. The space available for loading records (block size - padding-area) is therefore 900 bytes.

After loading some records, 800 bytes of the block have been used. The next record is 170 bytes long. This record cannot be loaded into the current block because less the 50 bytes would remain in the block after the record was loaded. Therefore, the record is loaded into the next block.

The current block remains filled to 800 bytes. The difference between 800 and 900 bytes (that is, -100 bytes) is used for balancing.

Suppose the next record had been 150 bytes instead of 170 bytes, and assume that the cumulative balancing value at that point in time is a negative number of bytes. The 150-byte record would be loaded because 50 bytes would remain in the block after the record was loaded (1000 - 950).

However, 50 bytes of the padding area would have been used (900 - 950) leaving +50 bytes for balancing.

For files loaded with the ADAM option, a new record is loaded into its calculated Data Storage block if space is available in the block (including the padding area). Records that cannot be stored in their calculated block are stored in another block (in this case, the padding area is not used).

DATAVOLUME: Data Storage Extent Volume



Note: The value for DATAVOLUME must be enclosed in apostrophes.

DATAVOLUME specifies the volume on which the file's Data Storage space (DS extents) is to be allocated. If the number of blocks requested with DSSIZE cannot be found on the specified volume, ADALOD retries the allocation while disregarding the DATAVOLUME value.

If DSRABN is specified, DATAVOLUME is ignored for the related file. If DATAVOLUME is not specified, the Data Storage space is allocated according to ADALOD's default allocation rules.

DSDEV: Data Storage Device Type

DSDEV specifies the device type on which the file's Data Storage is to be loaded. There is no default value; if DSDEV is not specified, an arbitrary device type is used.

DSREUSE: Data Storage Reusage

DSREUSE indicates whether Data Storage space which becomes available is to be reused. The default is YES.

ETID: Multiclient File Owner ID

The ETID parameter assigns a new owner ID to all records being loaded into a multiclient file. It specifies the user ID identifying the owner of the records being loaded. The owner ID assigned to the records is taken from the user profile of the specified user ID.

The ETID parameter must be specified if the file is to be loaded as a multiclient file (see the **LOWNERID parameter discussion**) and the input file contains no owner IDs; that is, the input file was not unloaded from a multiclient source file.

ETID is optional if the input file was unloaded from a multiclient source file. In this case, the loaded records keep their original owner IDs.

The ETID parameter must not be specified when loading a non-multiclient file.



Note: If the ETID parameter is used, the ADALOD utility requires an active nucleus. The nucleus will translate the ETID value into the internal owner ID value.

IGNFDT: Ignore Old FDT

When a file is deleted using the ADADBS DELETE function with the KEEPFDT parameter, the field definition table (FDT) remains in the Associator. When the file is again reloaded and IGNFDT is not specified, ADALOD compares the file's old FDT with the new one (security information is not compared). If both FDTs are identical, ADALOD loads the file and replaces the old FDT with the new FDT. If the FDTs are not identical, the old FDT is kept and the ADALOD operation ends with an error message.

Specifying the IGNFDT parameter causes ADALOD to ignore any existing (old) FDT for the file; no comparison is made. The new FDT replaces the old FDT, and ADALOD loads the file.

INDEXCOMPRESSION: Compress File Index

INDEXCOMPRESSION indicates whether the index of the file is loaded in compressed or uncompressed form. A compressed index usually requires less index space and improves the efficiency of index operations in the Adabas nucleus.

If INDEXCOMPRESSION is not specified, ADALOD obtains the default value from the sequential input file. If the input file was created using

- ADACMP, the default value is NO.
- ADAULD, the value of the file at the time of the unload is taken as the default.

ISNREUSE: ISN Reusage

ISNREUSE indicates whether or not an ISN freed as the result of deleting records may be reassigned to a new record. The default is NO. NO is recommended for files that mostly have inserts for performance reasons.



Note: ISNREUSE can cause excessive fragmentation of the file if the file is very volatile having many deletes and inserts. Adabas uses a performance based algorithm to search for a free ISN when doing an insert. RSP078 may be returned indicating that the file has reached the TOPISN and the file is severely fragmented.

ISNSIZE: 3- or 4-Byte ISN

ISNSIZE indicates whether ISNs in the file are 3 or 4 bytes long. The default is 3 bytes.

LIP: ISN Buffer Pool Size

LIP specifies the size of the ISN pool for containing ISNs and their assigned Data Storage RABNs. The value may be specified in bytes as a numeric value ("2048") or in kilobytes as a value followed by a "K" ("2K"). The default for LIP is 2000 bytes.

LIP can be used to decrease the number of address converter I/Os during loading when the USERISN=YES and the user-supplied ISNs are unsorted. Optimum performance is obtained if LIP specifies a buffer size large enough to hold all ISNs to be processed.

The length of one input record is ISNSIZE + RABNSIZE + 1. Thus the entry length is at least 7 bytes (the ISNSIZE of the file is 3 and the RABNSIZE of the database is 3) and at most 9 bytes (the ISNSIZE is 4 and the RABNSIZE is 4).

LOBFILE: LOB File Number

LOBFILE specifies the file number of the *LOB file* associated with the *base file* you are loading. This parameter is only used when loading base files.

For more information, read *Large Object (LB) Files and Fields*, in *Adabas DBA Tasks Manual*.

LOWNERID: Internal Owner ID Length for Multiclient File

The LOWNERID parameter specifies the length of the internal owner ID values assigned to each record for multiclient files. Valid length values are 0-8. If the LOWNERID parameter is not specified, its default value is the length of the owner IDs in the input file.

The specified or default value of the LOWNERID parameter determine whether a file is to be loaded as a multiclient or a non-multiclient file. If the effective LOWNERID value is zero, the file is loaded as a normal, non-multiclient file; if it is nonzero, the file is loaded as a multiclient file.

In combination with the ETID parameter, the LOWNERID parameter can be used to

- reload a non-multiclient file as a multiclient file;
- increase/decrease the length of the owner ID for the file; or
- remove the owner ID from the records of a file.

The following table shows the possible combinations of the LOWNERID parameter and the owner ID length in the input file.

LOWNERID Parameter Setting	Owner ID Length Value in Input File	
	0	2
0	Keep as a non-multiclient file	Convert to a non-multiclient file
1	Set up multiclient file (ETID)	Decrease owner ID length
2	Set up multiclient file (ETID)	Keep owner ID length
3	Set up multiclient file (ETID)	Increase owner ID length
(not specified)	Keep as a non-multiclient file	Keep as a multiclient file

When loading a multiclient file (the specified or default value of LOWNERID is non-zero), the ETID parameter can be specified to assign a new owner ID to all records being loaded. If the input file already contains owner IDs and ETID is omitted, all records keep their original owner IDs.

Where the table indicates the ETID parameter in the "Owner ID Length...0" column, the ETID parameter is mandatory, as there are no owner IDs given in the input file.

Note: If a *LOB file* is being loaded (read about the **FILE parameter**), the LOWNERID parameter cannot be specified in the ADALOD LOAD run.

LWP: Work Pool Size

LWP specifies the size of the work pool to be used for descriptor value sorting. The value can be specified in bytes or kilobytes followed by a "K". If no value is specified, the default is 1048576 bytes (or 1024K); however, to shorten ADALOD run time for files with very long descriptors or an unusually large number of descriptors, set LWP to a higher value. To avoid problems with the sort data set, a smaller LWP value should be specified when loading relatively small files.

The minimum work pool size depends on the sort data set's device type:

Sort Device	Minimum LWP	Minimum LWP
	Bytes	Kilobytes
3380	139264	136K
3390	159744	156K

MAXDS/ MAXNI/ MAXUI: Maximum Secondary Allocation

Specifies the maximum number of blocks per secondary extent allocation for Data Storage (MAXDS), normal index (MAXNI), or upper index (MAXUI). The value specified may either be in blocks and then be followed by a "B" (for example, MAXDS=8000B) or cylinders. If no limit is specified, no limit is assumed (the default).

MAXISN2: Allocate Secondary Address Converter RABNs to Account for Maximum Secondary ISNs

The MAXISN2 parameter is optional, regardless of whether or not spanned records exist in the ADALOD input file or not. Use this parameter to specify the desired size of the secondary address converter (AC2) in ISNs. The secondary address converter is used to map secondary ISNs of secondary spanned records to the RABNs of the Data Storage blocks where the secondary records are stored.

ADALOD determines the number of ISN mappings for which to allow space in the secondary AC using the following method:

1. If a secondary AC is not yet allocated, one secondary AC block is allocated.
2. If the current MAXISN2 setting is greater than or equal to four times the MAXISN value (the current maximum primary ISNs expected), the same algorithm is used to determine the number of secondary ISNs as is used to allocate additional primary ISNs.
3. If none of the above conditions is met, then secondary AC space is allocated as the smaller of the following two calculations:
   - The product of 10 times the old MAXISN2 setting (10 x oldMAXISN2)
   - The sum of the old MAXISN2 setting and the MAXISN setting (oldMAXISN2 + MAXISN)

There is no default value.

MAXRECL: Maximum Compressed Record Length

MAXRECL specifies the maximum compressed record length permitted for the file. The default is the maximum length supported by the device type being used.



Note: MAXRECL is ignored if specified for files enabled for spanned data storage records.

MINISN: Lowest ISN Count

This parameter specifies the lowest number of ISNs that can be assigned in the file. The default is 1.

The main purpose of MINISN is to assign the low end of the ISN range for a component file of an Adabas expanded file. MINISN is required when ANCHOR is specified for an expanded file.

Use MINISN to avoid wasting Associator space in files where all records are assigned ISNs significantly greater than 1. For example, a savings bank uses account numbers as ISN numbers, and the lowest account number is 1000001. Specifying MINISN = 1000001 stops Adabas from allocating address converter space for ISNs 1-999999, which would be unused. For more information, see the description of the MAXISN parameter.

MIXDSDEV: Data Storage Mixed Device Types

MIXDSDEV allows the allocation of secondary Data Storage extents on different device types, and therefore with different block lengths. If MIXDSDEV is not specified (the default), Data Storage extents for the specified file must all be on the same device type.



Note: MIXDSDEV is ignored if specified for files enabled for spanned data storage records.

NAME: File Name

NAME is the name to be assigned to the file. This name appears, along with data pertaining to this file, on the Database Status Report produced by the ADAREP utility. The maximum number of characters permitted is 16. The default name assigned is TESTFILE.

If the file name contains special characters or embedded blanks, the name must be enclosed within apostrophes ('...'), which themselves must be doubled if one is included in the name; for example, 'JAN''S FILE'.

NISIZE: Normal Index Size

NISIZE specifies the number of blocks or cylinders to be assigned to the normal index. A block value must be followed by a "B" (for example, "5500B").

If the specified extent size exceeds the largest free size, ADALOD allocates as many file extents as necessary to satisfy the request.

If the NISIZE parameter is omitted:

- ADALOD determines the space allocation for the normal index based on a sampling of records taken from the input data set. Since this calculation requires additional CPU time and I/O operations, we recommend setting this parameter if the size is known so that no estimation is performed.
- and INDEXCOMPRESSION=YES is set, the index size estimation made by ADALOD does not consider the index compression as it has no knowledge of the rate of compression to be expected. ADALOD may thus allocate a larger index than necessary.

If a small number of records is being loaded and a larger number of records is to be added later, the NISIZE parameter should be set to increase the Normal Index to accommodate the total record amount. For more information, see the section *LOAD File Space Allocation* in the ***LOAD Data and Space Requirements*** section.

NOACEXTENSION: Limit Address Converter Extents

If NOACEXTENSION is specified, the MAXISN defined for this file cannot be increased in the future. No additional address converter (AC) extents will be created. NOACEXTENSION applies mainly to component files comprising Adabas expanded files; if ANCHOR is specified, NOACEXTENSION must also be specified.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "`utility` TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

NUMREC: Limit Number of Records to Be Loaded

NUMREC specifies the limit on the number of records to be loaded. If NUMREC is specified, ADALOD stops after processing the specified number of records (unless an end-of-file condition on the input data set ends ADALOD operation before that time). This option is most often used to create a subset of a file for test purposes. If this parameter is omitted, all input records are processed.

If the input data set contains more records than specified by NUMREC, ADALOD processes the number of records specified by NUMREC and then ends with condition code 4.

Note: If a *LOB file* is being loaded (read about the **FILE parameter**), the NUMREC parameter cannot be specified in the ADALOD LOAD run.

PGMREFRESH: Program-Generated File Refresh

PGMREFRESH specifies whether a user program is allowed to perform a refresh operation on the file being loaded. If PGMREFRESH is specified, a refresh can be made using an E1 command, or an equivalent call to the nucleus.

READONLY: Read-only Status Indicator

READONLY indicates whether the read-only status is on or off for the file. Valid values for this parameter are "YES" (read-only status is on) and "NO" (read-only status is off).

If READONLY is not specified, the default is "NO."

REPLICATOR: Load an Adabas Replicator System File

The REPLICATOR parameter is an Event Replicator for Adabas parameter for an Event Replicator Server. It is a file type parameter. Use this parameter to load the Replicator system file into the Event Replicator Server.

The Replicator system file stores the Event Replicator initialization parameters. When it is loaded into the Event Replicator Server, it can be read during Event Replicator Server startup. You can modify the initialization parameters in the Replicator system file using the Adabas Event Replicator Subsystem. If the Replicator system file cannot be found at startup, the Event Replicator initialization parameters are read from the DDKARTE statements of the Event Replicator Server startup job.

The REPLICATOR parameter may not be specified when loading a file on a database that is not an Event Replicator Server. The contents of DDEBAND are ignored when loading a Replicator system file. For more information about Adabas system files, read about the FILE parameter of the ADALOD LOAD function, elsewhere in this section.

The REPLICATOR parameter may not be specified in the same ADALOD LOAD as the RPLTARGETID, or any of its associated parameters.

RESTART: Restart Interrupted ADALOD Execution

RESTART forces an interrupted ADALOD run to be restarted, beginning with the last *restart point* reached before the interruption. The *restart point* is the latest point of execution that can be restored from the Temp data set.

If ADALOD is interrupted by a defined error condition, ADALOD issues a message indicating whether or not a restart is possible.

When restarting the ADALOD operation, the following parameters may be changed:

- TEMPSIZE can be increased to make the temp data set larger. Note, however, that the temp data set content contains information necessary for the restart operation, and therefore *must not be changed* ;
- The SORTSIZE and SORTDEV parameters and the sort data set can be changed.

No other parameters can be changed. The DDEBAND/EBAND and DDFILEA/FILEA data sets must remain the same.

RPLDSBI: Before Image for Data Storage

The RPLDSBI parameter is an Event Replicator for Adabas parameter for an Adabas database that turns on the collection of before images of data storage during an update command to the file. Parameter RPLDSBI may only be specified if the RPLTARGETID parameter is also specified. Specify RPLDSBI to turn on the collection of data storage before images during an update. For

more information about how this setting is used in Adabas database processing during replication, read *Adabas Nucleus Replication Setup* and *Detailed Adabas Nucleus Processing* in the *Event Replicator for Adabas Concepts Manual* documentation.

The `RPLDSBI` parameter may not be specified in the same ADALOD LOAD run as the `REPLICATOR` parameter.

RPLERRORDEACTFILE: Deactivate Replication for File on Error

Use the RPLERRORDEACTFILE parameter to deactivate replication for the file for which replication errors occurred during ADALOD LOAD processing. RPLERRODEACTFILE controls ADALOD LOAD behavior for replication initialization errors as well as other replication errors (in contrast with the RPLINITERROR parameter which affects ADALOD LOAD behavior only for replication initialization errors).



Note: The RPLERRORDEACTFILE parameter can only be specified if the RPLLOAD parameter is set to "YES" or "FILE".

Valid settings for RPLERRORDEACTFILE are "YES" and "NO":

- Specifying "YES" indicates that replication for the file should be deactivated when a replication error occurs.
- Specifying "NO" (the default) indicates that replication for the file should not be deactivated.



Caution: If you elect to have ADALOD LOAD continue its processing after a replication error, you are also responsible for recovering your environment appropriately.

Once replication is deactivated for a file it can only be reactivated using Adabas Online System (AOS) or the ADADBS utility.

RPLINITERROR: Replication Control on Error

Use the RPLINITERROR parameter to indicate whether ADALOD processing should continue if replication initialization errors occur.



Note: The RPLINITERROR parameter can only be specified if the RPLLOAD parameter is set to "YES" or "FILE".

Valid settings for RPLINITERROR are "FAIL" and "CONTINUE":

- Specifying "FAIL" (the default) indicates that ADALOD processing should fail if a replication initialization error occurs. This is the how ADALOD LOAD works now, before the enhancement described in this enhancement preview is applied.
- Specifying "CONTINUE" indicates that ADALOD LOAD processing should continue if a replication initialization error occurs.



Caution: If you elect to have ADALOD LOAD continue its processing after a replication error, you are also responsible for recovering your environment appropriately.

RPLKEY: Primary Key for Replication

The RPLKEY parameter is an Event Replicator for Adabas parameter for an Adabas database that specifies the primary key for replication. This parameter may only be specified if the RPLTARGETID parameter is also specified. For more information about how this primary key is used in Adabas database processing during replication, read *Adabas Nucleus Replication Setup* and *Detailed Adabas Nucleus Processing* in the *Event Replicator for Adabas Concepts Manual* documentation.

The RPLKEY parameter may not be specified in the same ADALOD LOAD run as the REPLICATOR parameter.

RPLLOAD: Replicate Load Data

The ADALOD LOAD RPLLOAD parameter is an Event Replicator for Adabas parameter that can be specified for the Adabas database files when using replication. It is only allowed for ADALOD LOAD if replication is already turned on for a database.



Note: The version of Event Replicator specified in an ADALOD LOAD utility job that specifies RPLLOAD=YES must match the version of Event Replicator used by the Event Replicator Server. In addition, the Adabas version used by the Event Replicator Server must be greater than or equal to the Adabas version used in an ADALOD LOAD utility job that specifies RPLLOAD=YES.

The RPLLOAD parameter indicates whether or not data, and possibly the FCB/FDT, loaded to the Adabas database via the ADALOD LOAD utility will be replicated to the Event Replicator Server. Valid values are "YES", "FILE", and "NO"; the default is "NO":

- When RPLLOAD=YES is specified, ADALOD LOAD replicates data to the Event Replicator Server that it loads to the Adabas database.
- When RPLLOAD=FILE is specified, ADALOD LOAD replicates the FCB/FDT and data to the Event Replicator Server. When the data is replicated to an Adabas destination, the file is first allocated on the target database. The DESTINATION parameters DREPLICATEUTI for the target destination and DAREPLICATEUTI for the target file must be set to YES to use this option.
- When RPLLOAD=NO is specified, ADALOD LOAD does not replicate its load data to the Event Replicator Server.



Caution: If ADALOD LOAD ends abnormally (due to insufficient space, for example), updates made to the file before the abnormal ending cannot be backed out; there is no automated recovery for the updated data or for the replicated data. We therefore recommend that you perform an ADASAV SAVE on the file before you run ADALOD LOAD.

The ADALOD LOAD RPLLOAD parameter may not be set for a file with spanned DS record support.

RPLTARGETID: Replication Target ID

The RPLTARGETID parameter is an Event Replicator for Adabas parameter for an Adabas database that specifies the Event Replicator target ID used when the Adabas file data is replicated.

The RPLTARGETID parameter may not be specified in the same ADALOD LOAD as the REPLICATOR parameter. For more information about how this target ID is used in Adabas database processing during replication, read *Adabas Nucleus Replication Setup* in the *Event Replicator for Adabas Concepts Manual* documentation.



Note: Replication may not be turned on for an Adabas system file or a ciphered file.

RPLUPDATEONLY: Allow Only Event Replicator Processing Updates

The RPLUPDATEONLY parameter is an Event Replicator for Adabas parameter for an Adabas database that can be used in the ADALOD LOAD function to indicate whether an Adabas database file may be updated only by the Event Replicator Server as part of Adabas-to-Adabas replication or by other means as well. This parameter is optional. Valid values are "YES" or "NO". A value of "YES" indicates that the file can only be updated via Event Replicator processing; a value of NO indicates that the file can be updated by any normal means, including Event Replicator processing.

If the file is a new file, the default for this parameter is "NO".

However, if the file specified in the ADALOD LOAD function is an existing file, there is no default for this parameter. If no value is specified for the RPLUPDATEONLY parameter in the ADALOD LOAD function for an existing file, the value used previously for the file is used.

SKIPREC: Number of Records to Be Skipped

SKIPREC specifies the number of input records to be skipped before beginning load processing. The default is 0 (no records are skipped).



Note: If a *LOB file* is being loaded (read about the **FILE parameter**), the SKIPREC parameter cannot be specified in the ADALOD LOAD run.

SLOG: Load an Adabas SLOG File

The SLOG parameter is an Event Replicator for Adabas parameter for an Event Replicator Server. It is a file type parameter. Use this parameter to load the SLOG system file into the Event Replicator Server.

An SLOG file is an Adabas system file used only by the Event Replicator Server to store subscription logging data. The SLOG parameter may not be specified when loading a file on a database that is not an Event Replicator Server. The contents of DDEBAND are ignored when loading an SLOG file.

SORTDEV: Sort Device Type

ADALOD uses the sort data set to sort descriptor values. The SORTDEV parameter indicates the device type to be used for this data set. Refer to ***Enhancements to TEMP and SORT Pro-***

cessing in ADAINV, ADALOD and ADAULD for more details on specifying the SORTDEV parameter.

SORTSIZE: Sort Size

SORTSIZE specifies the space available for the sort data set or data sets R1/2. The value can be either cylinders (a numeric value only) or blocks (a numeric value followed by a "B"). If blocks are specified, they should be equivalent to a full number of cylinders. Refer to the *Adabas DBA Reference* documentation for more information on estimating the sort space. Refer also to *Enhancements to TEMP and SORT Processing in ADAINV, ADALOD and ADAULD* for more details on specifying the SORTSIZE parameter.

SYFMAXUV: Maximum MU System Field Values

The SYFMAXUV parameter can be used to specify the maximum number of values kept for a system field with the MU option during the execution of an update (A1) command (in other words, the maximum number of occurrences allowed for MU system fields during the execution of an update command). The value set for SYFMAXUV applies to all system fields in the file with the MU option. Valid values are integers from 1 through 20. The maximum value for SYFMAXUV is 20.

The internal default, if SYFMAXUV is not specified, is zero (0), which Adabas interprets to mean that there is no setting for this parameter at the file level. In this case, Adabas will assume a default of 1.

TEMPDEV: Temporary Storage Device Type

ADALOD uses the temp data set to store intermediate data. The TEMPDEV parameter indicates the device type to be used for this data set. Refer to *Enhancements to TEMP and SORT Processing in ADAINV, ADALOD and ADAULD* for more details on specifying the TEMPDEV parameter.

TEMPSIZE: Temporary Storage Size

TEMPSIZE specifies the size of the temp data set for the file. The Temp size equals the total of TEMP space required for each descriptor in the file; see the section *LOAD File Space Allocation* in the *LOAD Data and Space Requirements* section for more information. The size can be either in cylinders or blocks (followed by a "B"). Refer also to *Enhancements to TEMP and SORT Processing in ADAINV, ADALOD and ADAULD* for more details on specifying the TEMPSIZE parameter.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

UISIZE: Upper Index Size

UISIZE specifies the number of blocks or cylinders to be assigned to the upper index. A block value must be followed by a "B" (for example, "5500B").

If the specified extent size exceeds the largest free size, ADALOD allocates as many file extents as necessary to satisfy the request.

If the UISIZE parameter is omitted:

- ADALOD determines the space allocation for the upper index based on a sampling of records taken from the input data set. Since this calculation requires additional CPU time and I/O operations, we recommend setting this parameter if the size is known so that no estimation is performed.
- and INDEXCOMPRESSION=YES is set, the index size estimation made by ADALOD does not consider the index compression as it has no knowledge of the rate of compression to be expected. ADALOD may thus allocate a larger index than necessary.

If a small number of records is being loaded and a larger number of records are to be added later, the UISIZE parameter should be set to increase the upper index to accommodate the total record amount. For more information, see the section *LOAD File Space Allocation* in the ***LOAD Data and Space Requirements*** section.

UQDE: Unique Descriptors

UQDE defines one or more descriptors as unique. Each descriptor specified must contain a different value in each input record. If a non-unique value is detected during ADALOD processing, ADALOD terminates with an error message.

If the unique descriptor (UQ) option was specified with the ADACMP utility, the UQDE parameter here is not required.

Adabas prevents a descriptor defined with the unique descriptor (UQ) option from being updated with an add or update command if the update would cause a duplicate value for the descriptor.

Note: For Adabas expanded files, ADALOD can only detect unique descriptor violations within the component file. If an identical value exists for a unique descriptor in one of the other component files, ADALOD cannot detect it. You must therefore ensure that unique descriptor values remain unique throughout an expanded file.

Note: If a *LOB file* is being loaded (read about the **FILE parameter**), the UQDE parameter cannot be specified in the ADALOD LOAD run.

USERISN: User ISN Assignment

USERISN=YES indicates that the USERISN option for the loaded file is to be in effect, and that the ISN for each new record is being supplied by the user in the input data. In the case that USERISN=YES, the max ISN of the file will not be increased if (MAXISN - MINISN) + 1 records are to be loaded.

If USERISN=NO, Adabas assigns the ISN for each new record.

If USERISN is not specified, a default setting is assumed that depends on the input file itself. If the input file was created by ADACMP with the USERISN option or by ADAULD from a file having the USERISN option, the default for ADALOD operation is USERISN=YES; other-

wise, the default is USERISN=NO. Specifying USERISN here overrides the existing default value.

Note: Adabas 5.2 files initially loaded with the USERISN option do not require USERISN=YES to again be specified when the files are reloaded; ADALOD assumes the default as described above. However, Adabas 5.1 files initially loaded with the USERISN option *must* have USERISN=YES specified whenever they are reloaded.

VERSION: Input Data Format

Originally, this parameter specified the Adabas version of the output (ADACMP) data sets to be loaded into Adabas.

Because ADALOD determines the version of the sequential input data set itself, this parameter is ignored. It is available only for compatibility with old ADALOD jobs.

Examples

Example 1:

```
ADALOD  LOAD FILE=6,MAXISN=20000,DSSIZE=20,ASSOPFAC=15,
ADALOD       DATAPFAC=15,TEMPSIZE=20,SORTSIZE=10
```

File 6 is to be loaded. The number of records initially permitted for the file is 20,000. 20 cylinders are to be allocated for Data Storage. The Associator and Data Storage block padding factors are both 15 percent. The temp and sort data sets are 20 and 10 cylinders, respectively.

Example 2:

```
ADALOD  LOAD FILE=7,MAXISN=350000,ASSOPFAC=5,MINISN=100001
ADALOD       DATAPFAC=15,DSSIZE=100,USERISN=YES
ADALOD       TEMPSIZE=200,SORTSIZE=100
```

File 7 is to be loaded. The number of records initially allocated for the file is 250,000, and the minimum is 100,001. The Associator padding factor is 5 percent. The Data Storage padding factor is 15 percent. 100 cylinders are to be allocated for Data Storage. ISNs are contained in the input. The temp and sort data sets are equal to 200 and 100 cylinders, respectively.

Example 3:

```
ADALOD  LOAD FILE=8,ADAMFILE,ADAMDE='AK'
ADALOD       ADAMPARM=4,ADAMOFLOW=5,UQDE='AK',MINISN=1
ADALOD       MAXISN=10000,DSSIZE=20,ASSOPFAC=5,DATAPFAC=5
ADALOD       TEMPSIZE=10,SORTSIZE=5
```

File 8 is to be loaded as an ADAM file. Field AK is the ADAM key. 4 bits are to be truncated from each value of AK before using the value to calculate the Data Storage RABN for the record. The size of the ADAM overflow area is 5 cylinders. The field AK is defined as a unique descriptor. The

maximum number of records initially allocated for the file is 10,000. Twenty (20) cylinders are to be allocated to Data Storage, from which the five ADAM overflow cylinders are taken. The padding factor for both the Associator and Data Storage is five percent. The sizes of the Temp and Sort data sets are ten and five cylinders, respectively.

Example 4:

```
ADALOD  LOAD FILE=9,NAME=INVENTORY,MAXISN=5000
ADALOD       DSSIZE=2000B,DSRABN=30629,NISIZE=300B,UISIZE=50B
ADALOD       MAXDS=1000B,MAXNI=50B,MAXUI=1B
ADALOD       INDEXCOMPRESSION=YES
ADALOD       ASSOPFAC=20,DATAPFAC=10
ADALOD       TEMPSIZE=10,SORTSIZE=5,UQDE='U1,U2'
```

File 9 is to be loaded. The text name for the file is INVENTORY. The initial space allocation for the file is for 5000 records. Two thousand (2000) blocks are to be allocated for Data Storage, beginning with RABN 30,629. 300 blocks are to be allocated for the normal index. 50 blocks are to be allocated to the upper index. The maximum allocations per secondary extent for Data Storage, normal index and upper index are 1000 blocks, 50 blocks, and 1 block respectively. The index is to be compressed. The padding factor for the Associator is 20 percent. The padding factor for Data Storage is 10 percent. The sizes of the temp and sort data sets are 10 and 5 cylinders respectively. Descriptors U1 and U2 are defined as unique descriptors.

Example 5:

```
ADALOD  LOAD FILE=2,SECURITY
ADALOD       DSSIZE=20B,MAXISN=2000,NISIZE=20B,UISIZE=5B
ADALOD       TEMPSIZE=10,SORTSIZE=5
```

File 2 is to be loaded as an Adabas security file. The DDEBAND contents are ignored. Space is allocated for Data Storage (20 blocks), for the address converter (2000 ISNs), the normal index (20 blocks), and the upper index (5 blocks). The temp size is 10 cylinders, and the sort area size is 5 cylinders.

LOAD Data and Space Requirements

The following general information describes data requirements for LOAD operation, and how ADALOD LOAD allocates space. For more information about space allocation, refer to the *Adabas DBA Reference* documentation.

Input Data for LOAD Operations

Compressed data records produced by the ADACMP or ADAULD utility may be used as input to ADALOD. If output from an ADAULD utility run made with the MODE=SHORT option is used as ADALOD input, any descriptor information will be removed from the FDT, and no index will exist for the file.

LOAD File Space Allocation

ADALOD allocates space for the normal index (NI), upper index (UI), address converter (AC), Data Storage, and the temp area for the file being loaded.

Index Space Allocation

If the NISIZE and/or the UISIZE parameters are supplied, allocation is made using the user-supplied values. If these parameters are not supplied, ADALOD allocates space for these indexes based on a sampling of the values present for each descriptor.

Descriptor values are sampled as follows:

1. ADALOD reads the compressed input, stores the records into Data Storage, extracts each value for each descriptor and writes these values to the temp data set. Each temp block contains values for one descriptor only. At the end of this processing phase, the following information is present:
   - number of values for each descriptor
   - number of bytes required for each descriptor
   - temp RABNs used for each descriptor

   For unique descriptors, the NI space requirement is equal to the temp size used. For non-unique descriptors, the number of duplicate values must be determined. Each duplicate value's space requirement must be estimated and then subtracted from the number of bytes required. The result is the NI size required for the duplicate descriptor.

   The number of duplicate values is determined by reading up to 16 temp blocks containing values for a single descriptor. These values are sorted to determine how many are duplicates. The resulting count of duplicate values is multiplied by the factor:

   $$\frac{\text{total number of values}}{\text{number of values in the sample}}$$

   The result is the estimated number of identical descriptor values present in the entire file for this descriptor. This space requirement is subtracted from the temp size estimate.
2. The upper index (UI) size is computed after all normal index (NI) and temp sizes are available.
3. The NI and UI sizes are each multiplied by the result of:

$$\frac{(MAXISN - MINISN) + 1}{\text{number of records being loaded}}$$

For example, if 10000 records require 10 blocks of UI space and 500 blocks of NI space with MINISN = 1 (the default), the specification of MAXISN = 60000 causes 60 UI blocks and 3000 NI blocks to be allocated:

$$10 \times \frac{60000}{10000} = 60 \text{ UI Blocks}$$

$$500 \times \frac{60000}{10000} = 3000 \text{ NI Blocks}$$

However, this calculation is not made if USERISN=YES is in effect.

By setting MAXISN appropriately, it is therefore possible to increase the size allocation for files in which a small number of records are being loaded and for which a much larger number of records are to be added subsequently.

If the NISIZE and UISIZE parameters have been specified, the space allocation is made using unassigned Associator RABNs. If the NIRABN and/or the UIRABN parameters are supplied, space allocation is made at the user-specified RABN.

Address Converter Space Allocation

The address converter allocation is based on the MAXISN and MINISN values for the file. ADALOD allocates the blocks needed to contain the number of bytes calculated by the formula:

```
RABNSIZE x((MAXISN - MINISN)+ 1)
```

If the ACRABN parameter has been specified, ADALOD allocates the address converter beginning with the user-specified block number; otherwise, it uses unassigned Associator RABNs.

Data Storage Space Allocation

Data Storage allocation is based upon the value specified with the DSSIZE parameter. If the DSRABN parameter has been specified, the allocation is made beginning with the user-specified block number; otherwise, unassigned Data Storage RABNs are used.

If there are different device types in the database, Data Storage allocation can be forced on a specified device type by specifying DSDEV. The MIXDSDEV parameter permits Data Storage allocation on different device types, assuming the device types can store records with the length specified by MAXRECL.

Temp Area Space Allocation

For each descriptor, ADALOD generates a list of the values and ISNs of the records containing the value, and writes this information to the Temp data set. The space required for descriptor information is equal to the sum of the space required for each descriptor. The space needed for each descriptor can be calculated using the following formula:

```
SP = N x  NPE x  NMU x  (L + 4)
```

where

SP	is the space required for the descriptor (in bytes).
N	is the number of records being loaded.
NPE	is the average number of occurrences, if the descriptor is contained in a periodic group. If not in a periodic group, NPE equals 1.
NMU	is the average number of occurrences, if the descriptor is a multiple-value field. If not a multiple-value field, NMU equals 1.
L	is the average length (after compression) of each value for the descriptor.

Example:

A file containing 20,000 records is being loaded. The file contains two descriptors (AA and CC). Descriptor AA has 1 value in each record and the average compressed value length is 3 bytes. Descriptor CC has an average of 10 values in each record and the average compressed value length is equal to 4 bytes.

Field Definitions:

```
01,AA,5,U,DE
01,CC,12,A,DE,MU
```

Space requirement for AA.

```
SP = 20,000 •  1 •  (3 + 4)
SP = 140,000 bytes
```

Space requirement for CC.

```
SP = 20,000 •  10 •  (4 + 4)
SP = 1,600,000 bytes
```

Total space requirement = *1,740,000 bytes* .

The number of cylinders required may be calculated by dividing the number of blocks required by the number of blocks per cylinder.

For a model 3390 device type:

$$\text{Blocks required} = \frac{1{,}740{,}000}{8904 \text{ bytes per block}} = 196 \text{ blocks}$$

$$\text{Cylinders required} = \frac{196 \text{ blocks}}{90 \text{ blocks per cylinder}} = 3 \text{ cylinders}$$

Associator Updating by LOAD

ADALOD then sorts the descriptor values collected in the input phase and enters the sorted values into the normal index and upper index. If the allocated index space is not enough for the normal index or upper index, ADALOD allocates additional extents.

Each additional extent allocated is equal to about 25 percent of the total current space allocated to the index. If insufficient space is available for the additional extent or if the maximum number of allocated extents has been reached, ADALOD terminates with an error message.

Loading Expanded Files

An expanded file is made up of a series of normal Adabas physical files. The number sequence of the files within the expanded file is arbitrary. The first file may be file 53; the second, file 127; the third, 13, and so on. ISNs assigned to each component file must be unique; no two files can contain the same ISN. The ISN range over all files must be in ascending order; however, there can be gaps in the sequence.

The total number of records in an expanded-file chain cannot exceed 4,294,967,294.

The sequence of physical component files that build an expanded logical file is defined by the ANCHOR parameter, which defines the first component file (anchor) in the sequence. The anchor file is loaded just as any other Adabas file; each additional component file must be loaded with the ANCHOR parameter referring to the anchor file. ADALOD inserts the new physical file into the existing expanded file chain according to the range of ISNs assigned to the added file. Each added component file must also specify the NOACEXTENSION parameter when being loaded to prevent Adabas from assigning new ISNs to a component file.

ADALOD processes only the anchor file and the single physical (component) files that compose an expanded file, and not the complete expanded file itself.

Loading Data into an Expanded File

To load data (for example, several million records) into different physical files, the input data must first be divided into several DDEBAND input files. The DDEBAND file data may be mapped into the component files using the SKIPREC and NUMREC parameters; however, one-to-one mapping without skipping or limits is recommended. This avoids the need to read records that will not be used later, and thus improves performance.

Examples:

The following examples, which show parts of one or more jobs for loading an expanded file, illustrate the mapping of DDEBAND file data into component files:

```
//DDEBAND    DD DSN=LOAD.DATA.FILE1,...
//DDKARTE    DD *
ADALOD LOAD FILE=40,NAME='XXX_Part1'
ADALOD      MINISN=1,MAXISN=10000000,NOACEXTENSION
ADALOD      NUMREC=10000000
ADALOD        DSSIZE=...,NISIZE=...,UISIZE...
ADALOD        SORTSIZE=...,TEMPSIZE=...
            .
            .

//DDEBAND    DD DSN=LOAD.DATA.FILE1,...
//DDKARTE    DD *
ADALOD LOAD FILE=41,NAME='XXX_Part2',ANCHOR=40
ADALOD      MINISN=10000001,MAXISN=20000000,NOACEXTENSION
ADALOD      NUMREC=10000000,SKIPREC=10000000
ADALOD        DSSIZE=...,NISIZE=...,UISIZE...
ADALOD        SORTSIZE=...,TEMPSIZE=...
            .
            .

//DDEBAND    DD DSN=LOAD.DATA.FILE2,...
//DDKARTE    DD *
ADALOD LOAD FILE=35,NAME='XXX_Part2',ANCHOR=40
ADALOD      MINISN=20000001,MAXISN=30000000,NOACEXTENSION
ADALOD      NUMREC=10000000
ADALOD        DSSIZE=...,NISIZE=...,UISIZE...
ADALOD        SORTSIZE=...,TEMPSIZE=...
            .
            .
```

Loading Multiclient Files



Note: A multiclient file cannot be made part of an expanded file, and an expanded file cannot be converted to a multiclient file.

A multiclient file stores records for multiple users or groups of users. It divides the physical file into multiple logical files by attaching an owner ID to each record. Each user can access only the subset of records that is associated with the user's owner ID.

For any installed external security package such as RACF or CA-Top Secret, a user is still identified by either Natural ETID or LOGON ID. The owner ID is assigned to a user ID. A user ID can have only one owner ID, but an owner ID can belong to more than one user.

The ADALOD LOAD function uses the LOWNERID and ETID parameters to support the migration of an application from a standard to a multiclient environment. The parameters work together to define owner IDs and determine whether a file is a multiclient file.

LOWNERID specifies the length of the internal owner ID values assigned to each record for multiclient files. In combination with the ETID parameter, the LOWNERID parameter can be used to reload a standard file as a multiclient file, change the length of the owner ID for the file, or remove the owner ID from the records of a file.

If the LOWNERID parameter is not specified, the length of the owner ID for the input file (if any) remains the same.

ETID assigns a new owner ID to all records being loaded into a multiclient file, and must be specified if the input file contains no owner IDs; that is, the input file was not unloaded from a multiclient source file.

Examples of Loading/Updating Multiclient Files

```
ADALOD LOAD FILE=20,LOWNERID=2,NUMREC=0
```

Creates file 20 as a multiclient file. The length of the internal owner ID is two bytes, but no actual owner ID (ETID) is specified. No records are actually loaded in the file (NUMREC=0).

```
ADALOD LOAD FILE=20,LOWNERID=2,ETID=USER1
```

Creates file 20 as a multiclient file, load all supplied records, and assign them to user USER1. The length of the internal owner ID is two bytes.

```
ADALOD UPDATE FILE=20,ETID=USER2
```

Performs a mass update to add records to file 20, a multiclient file. Load all the new records and assign them to USER2.

Enhanced Temp and Sort Processing

The TEMPSIZE and SORTSIZE parameters are now optional. When omitted, various enhancements to TEMP/SORT processing are available to the user. See *Enhancements to TEMP and SORT Processing in ADAINV, ADALOD and ADAULD* for details.

114 UPDATE: Add/Delete Records

- Essential Parameters 831
- Optional Parameters and Subparameters 832
- Examples 838
- Formats for Specifying ISNs 839
- UPDATE Data and Space Requirements 840
- Mass Updates of Expanded Files 842
- Enhanced Temp and Sort Processing 843

Caution: If ADALOD UPDATE ends abnormally (due to insufficient space, for example), updates made to the file before the abnormal ending cannot be backed out. Software GmbH therefore recommends that you perform ADASAV SAVE on the file before you run ADALOD UPDATE.

The UPDATE function adds or deletes a large number of records (ISNs) to or from an existing file. A single UPDATE operation can both add and delete ISNs.

Records to be added must be in compressed (ADACMP or ADAULD output) form and be in the DDEBAND input data set.

ISNs to be deleted must be specified by either or both of the DDISN and DELISN parameters.

Notes:

1. The UPDATE function cannot be used with an Adabas system file if the Adabas nucleus is active, and cannot be used to change the checkpoint or security files.
2. A multiclient file cannot be made part of an expanded file, and an expanded file cannot be converted to a multiclient file.

```
ADALOD UPDATE   FILE = file-number
                [DDISN]
                [DELISN = isn-list ]
                [DSREUSE = { YES | NO } ]
                [ETID = multiclient-file-owner-id ]
                [ISNREUSE = { YES | NO } ]
                [LIP = { isn-pool-size | 2000 } ]
                [LWP = { work-pool-size | 1048576 } ]
                [MAXISN = number
                    [ACRABN = starting-rabn ]
                    [ASSOVOLUME = 'Associator-extent-volume' ] ]
                [MAXISN2 = number
                    [AC2RABN = starting-rabn]
                    [ASSOVOLUME = 'Associator-extent-volume' ] ]
                [NOUSERABEND]
                [NUMREC = number]
                [PASSWORD = 'password ' ]
                [RESTART]
                [RPLLOAD = { YES | NO } ]
                [SAVEDREC]
                [SKIPREC = {number | 0 } ]
                [SORTDEV = device-type ]
                [SORTSIZE = size ]
                [TEMPDEV = device-type ]
                [TEMPSIZE = size ]
                [TEST]
                [USERISN = { YES | NO } ]
```

Essential Parameters

FILE: File Number

FILE specifies the number of the file to be updated. If a component file of an Adabas expanded file is specified, only that component file is updated; the other component files must be updated in separate UPDATE operations.

Optional Parameters and Subparameters

ACRABN: Starting RABN for Address Converter

ACRABN causes additional space allocation for the address converter to begin at the specified RABN. ACRABN is effective only if MAXISN specifies an increase for the file's address converter.

AC2RABN: Starting RABN for Secondary Address Converter

AC2RABN causes additional space allocation for the secondary address converter to begin at the specified RABN. AC2RABN is effective only if MAXISN2 specifies an increase for the file's secondary address converter. The secondary address converter is used to map the secondary ISNs of secondary spanned records to the RABNs of the Data Storage blocks where the secondary records are stored. There is no default value.

ASSOVOLUME: Associator Extent Volume



Note: The value for ASSOVOLUME must be enclosed in apostrophes.

ASSOVOLUME is effective only if MAXISN or MAXISN2 specify an increase for the file's address converter.

ASSOVOLUME specifies the volume on which the file's address converter extents is to be allocated. If the requested number of blocks cannot be found on the specified volume, ADALOD retries the allocation while disregarding the ASSOVOLUME parameter.

If ACRABN or AC2RABN is specified, ADALOD ignores the ASSOVOLUME value when allocating the address converter extent type. If ASSOVOLUME is not specified, the file's Associator space is allocated according to ADALOD's default allocation rules.

DDISN: Read ISNs to be Deleted from Sequential Data Set

If DDISN is specified, ISNs to be deleted are read from the DDISN sequential data set. If both the DDISN and DELISN parameters are specified, the ISNs from the two lists are merged. The DDISN data set must have variable or variable blocked records. See the section **Formats for Specifying ISNs** for more information.

When the UPDATE function is executed, all ISNs are first read and stored in the ISN pool in the order they occur. The size of the ISN pool (specified by LIP) must be large enough to store all data read from DDISN. The records are then sorted in ascending order. Overlapping ranges and duplicate ISNs are not allowed. ISNs not found during processing are ignored.

When deleting ISNs from an Adabas expanded file, you can specify the complete ISN list for all component files; the UPDATE function automatically selects only the ISNs that are appropriate for the component file being processed.



Note: This parameter cannot be specified in an ADALOD UPDATE operation on a *LOB file*.

DELISN: ISNs to be Deleted

DELISN specifies a list of the ISNs of records to be deleted. If both DDISN and DELISN are specified, the ISNs from the two lists are merged. A range list may be specified as:

```
DELISN=10-80,90,100-110 ↩
```

Overlapping ranges and duplicate ISNs are not allowed. You can specify, at most, 32 single ISNs or ISN ranges. When deleting ISNs from an Adabas expanded file, you can specify the complete list for all component files. The UPDATE function selects the appropriate ISNs from the list and deletes them from the component file.



Note: This parameter cannot be specified in an ADALOD UPDATE operation on a *LOB file*.

DSREUSE: Data Storage Reusage

DSREUSE indicates whether or not Data Storage space that becomes available as a result of a record deletion is to be reused.

This parameter is in effect for the execution of the UPDATE function only. The permanent setting of DSREUSE is not changed. That permanent setting is the default if this value is not specified.

ETID: Multiclient File Owner ID

The ETID parameter assigns a new owner ID to all records being added to an existing multiclient file. The owner ID is automatically adjusted to the length for owner IDs specified by LOWNERID when the multiclient file was last loaded. If no ETID is specified, all loaded records keep their owner IDs specified on the input source.

The ETID parameter must be specified if the existing file is multiclient and the input file was not unloaded from a multiclient file. ETID must not be specified if the existing file is a non-multiclient file.



Note: If the ETID parameter is used, the ADALOD utility requires an active nucleus. The nucleus will translate the ETID value into the internal owner ID value.

ISNREUSE: ISN Reusage

ISNREUSE indicates whether the ISN for a deleted record can be reassigned to a new record. The default is NO. NO is recommended for files that mostly have inserts for performance reasons.



Note: ISNREUSE can cause excessive fragmentation of the file if the file is very volatile having many deletes and inserts. Adabas uses a performance based algorithm to search for a free ISN when doing an insert. RSP078 may be returned indicating that the file has reached the TOPISN and the file is severely fragmented.

LIP: ISN Work Pool Size

LIP specifies the size of the work pool for containing ISNs to be deleted. Four bytes per ISN and eight bytes per ISN range are required in this pool. The value may be specified in bytes as a numeric value ("2048") or in kilobytes as a value followed by a "K" ("2K"). The default for LIP is 2000 bytes.

LWP: Work Pool Size

LWP specifies the size of the work pool to be used for descriptor value sorting. The value can be specified in bytes or kilobytes followed by a "K". If no value is specified, the default is 1048576 bytes (or 1024K); however, to shorten ADALOD run time for files with very long descriptors or an unusually large number of descriptors, set LWP to a higher value. To avoid problems with the Sort data set, a smaller LWP value should be specified when updating relatively small files.

The minimum work pool size depends on the sort data set's device type:

Sort Device	Minimum LWP	Minimum LWP
	Bytes	Kilobytes
3380	139264	136K
3390	159744	156K

MAXISN: Highest ISN to be Allocated to the File

The MAXISN parameter may be used to specify a new ISN setting for the file. This parameter should be used if the current record count plus the number of ISNs (records) to be added exceeds the current MAXISN setting. The specified larger value determines the additional space required for the address converter, and causes ADALOD to allocate a new extent. A smaller MAXISN value causes no change in the address converter space.



Note: The MAXISN setting for a file cannot be increased if the file was last loaded with NOACEXTENSION active.

The MAXISN setting should be increased by an amount suitable for all planned expansion; this avoids using up the address converter extent too quickly, and alleviates the need to either unload and reload the file or run the ADAORD REORFASSO utility because the maximum number of address converter extents has been allocated.

With the optional ACRABN parameter, the beginning of the new address converter extent can be set to a specific RABN number. See the ACRABN parameter description for more information.

If the MAXISN parameter is omitted, ADALOD allocates new address converter extents only if the old MAXISN value is exceeded.

MAXISN2: Highest ISN to be Allocated in the Secondary Address Converter for the File

The MAXISN2 parameter is optional, regardless of whether or not spanned records exist in the ADALOD input file or not. Use this parameter to specify the desired size of the secondary address converter (AC2) in ISNs. The secondary address converter is used to map secondary

ISNs of secondary spanned records to the RABNs of the Data Storage blocks where the secondary records are stored.

ADALOD determines the number of ISN mappings for which to allow space in the secondary AC using the following method:

1. If a secondary AC is not yet allocated, one secondary AC block is allocated.
2. If the current MAXISN2 setting is greater than or equal to four times the MAXISN value (the current maximum primary ISNs expected), the same algorithm is used to determine the number of secondary ISNs as is used to allocate additional primary ISNs.
3. If none of the above conditions is met, then secondary AC space is allocated as the smaller of the following two calculations:
   - The product of 10 times the old MAXISN2 setting (10 x oldMAXISN2)
   - The sum of the old MAXISN2 setting and the MAXISN setting (oldMAXISN2 + MAXISN)

There is no default value.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

NUMREC: Limit Number of Records to Be Added

NUMREC limits the number of records to be added. If NUMREC is specified, ADALOD processing terminates after adding the number of records specified (unless an end-of-file condition on the input data set has already caused ADALOD termination). If this parameter is omitted, *all* input records are added.

If the input data set contains more records than specified by NUMREC, ADALOD adds the number of records specified by NUMREC and then terminates with condition code 4.

Note: This parameter cannot be specified in an ADALOD UPDATE operation on a *LOB file*.

PASSWORD: File Password

If the file to be updated is password-protected, the parameter must be used to provide a valid password. There is no default for PASSWORD.

RESTART: Restart Interrupted ADALOD Execution

RESTART forces an interrupted ADALOD run to be restarted, beginning with the last *restart point* reached before the interruption. The *restart point* is the latest point of execution that can be restored from the Temp data set.

If ADALOD is interrupted by a defined error condition, ADALOD issues a message indicating whether or not a restart is possible.

When restarting the ADALOD operation, the following parameters may be changed:

- TEMPSIZE can be increased to make the temp data set larger. Note, however, that the temp data set contents *must not be changed* because it contains information necessary for the restart operation;
- The SORTSIZE and SORTDEV parameters and the sort data set can be changed.

No other parameters can be changed. The DDEBAND, DDFILEA, and DDISN data sets must remain the same.

RPLLOAD: Replicate Update Data

The ADALOD UPDATE RPLLOAD parameter is an Event Replicator for Adabas parameter that can be specified for the Adabas database files when using replication. It is only allowed for ADALOD UPDATE if replication is already turned on for a database.



Note: The version of Event Replicator specified in an ADALOD UPDATE utility job that specifies RPLLOAD=YES must match the version of Event Replicator used by the Event Replicator Server. In addition, the Adabas version used by the Event Replicator Server must be greater than or equal to the Adabas version used in an ADALOD UPDATE utility job that specifies RPLLOAD=YES.

The `RPLLOAD` parameter indicates whether or not inserts and deletes to the Adabas database via the ADALOD UPDATE utility will be replicated to the Event Replicator Server. Valid values are "YES" and "NO"; the default is "NO".

When RPLLOAD=YES is specified, ADALOD UPDATE inserts and deletes to the Adabas database will be replicated to the Event Replicator Server. When RPLLOAD=NO is specified, ADALOD UPDATE inserts and deletes are *not* replicated.



Caution: If ADALOD UPDATE ends abnormally (due to insufficient space, for example), updates made to the file before the abnormal ending cannot be backed out; there is no automated recovery for the updated data or for the replicated data. We therefore recommend that you perform an ADASAV SAVE on the file before you run ADALOD UPDATE.

SAVEDREC: Save Deleted Records on a Sequential File

SAVEDREC indicates that deleted records are to be written to a sequential data set. The format of the data set is identical to that created by the ADAULD utility with the MODE=SHORT option.

Note: This parameter cannot be specified in an ADALOD UPDATE operation on a *LOB file*.

SKIPREC: Number of Records to Be Skipped

SKIPREC is the number of input records to be skipped before beginning to process updates. The default is 0 (no records are skipped).

Note: This parameter cannot be specified in an ADALOD UPDATE operation on a *LOB file*.

SORTDEV: Sort Device Type

ADALOD uses the sort data set to sort descriptor values. The SORTDEV parameter indicates the device type to be used for this data set. Refer to ***Enhancements to TEMP and SORT Processing in ADAINV, ADALOD and ADAULD*** for more details on specifying the SORTDEV parameter.

SORTSIZE: Sort Size

SORTSIZE is the number of blocks or cylinders available for the sort data set. Refer to ***Enhancements to TEMP and SORT Processing in ADAINV, ADALOD and ADAULD*** for more details on specifying the SORTSIZE parameter.

TEMPDEV: Temporary Storage Device Type

ADALOD uses the temp data set to store intermediate data. The TEMPDEV parameter indicates the device type to be used for this data set. Refer to ***Enhancements to TEMP and SORT Processing in ADAINV, ADALOD and ADAULD*** for more details on specifying the TEMPDEV parameter.

TEMPSIZE: Temporary Storage Size

TEMPSIZE is the number of blocks or cylinders available for the temp data set. Refer to ***Enhancements to TEMP and SORT Processing in ADAINV, ADALOD and ADAULD*** for more details on specifying the TEMPSIZE parameter.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

USERISN: User ISN Assignment

USERISN=YES indicates that the USERISN option for the file is to be in effect, and that the ISN for each new record is being supplied by the user in the input data. If USERISN=NO, Adabas assigns the ISN for each new record.

The specified USERISN setting is effective only while the UPDATE function is executing. The permanent USERISN setting is not changed, and is the default if this parameter is not specified.

When performing an ADALOD UPDATE function on a file with a hyperdescriptor for which the hyperdescriptor exit changed the ISNs of descriptor values, USERISN=YES is no longer required for the add/load operation.

When adding records *from* a non-USERISN=YES file, the ADALOD parameter USERISN=NO must be specified and the file to be updated must have the USERISN option. This feature is useful for Adabas Text Retrieval (TRS).

Examples

Example 1:

```
ADALOD  UPDATE FILE=6,MAXISN=18000
ADALOD         TEMPSIZE=10,SORTSIZE=5
```

Records are to be added to file 6. The MAXISN for the file is to be increased to 18000.

Example 2:

```
ADALOD  UPDATE FILE=7,TEMPSIZE=10,
ADALOD         ETID=USER3,SORTSIZE=5
```

Records with user's owner ID of USER3 are to be added to multiclient file 7.

Example 3:

```
ADALOD  UPDATE FILE=8,DELISN=1000-1999,5000-5999
ADALOD         TEMPSIZE=10,SORTSIZE=5
```

The records with ISNs 1000 to 1999 and 5000 to 5999 are to be deleted from file 8. If an input data set is provided, records are to be added.

Example 4:

```
ADALOD  UPDATE FILE=6
ADALOD         DDISN,SAVEDREC
ADALOD         TEMPSIZE=10,SORTSIZE=5
```

Records are to be deleted from file 6. The ISNs of the records to be deleted are contained in an input data set. The deleted records are to be saved on an output data set.

Example 5:

```
ADALOD  UPDATE FILE=6,DDISN,LIP=20K,SKIPREC=500
ADALOD         TEMPSIZE=5,SORTSIZE=10
```

Records are to be added and deleted from file 6. The ISNs which identify the records to be deleted are contained in an input data set (DDISN). The size of the ISN pool is set to 20K. The first 500 records on the input data set are to be skipped.

Formats for Specifying ISNs

There are two formats for specifying ISNs in the DDISN data set. The first format can be used in all cases where only 31-bit ISNs are specified. A record can contain a mix of single ISNs and ranges of ISNs.

The second format supports 32-bit ISNs and can only be used with Adabas version 6 and above. Each record can specify *either* single ISNs (indicated by X'00000000' in the first fullword) *or* ranges of ISNs (indicated by X'FFFFFFFF' in the first fullword).

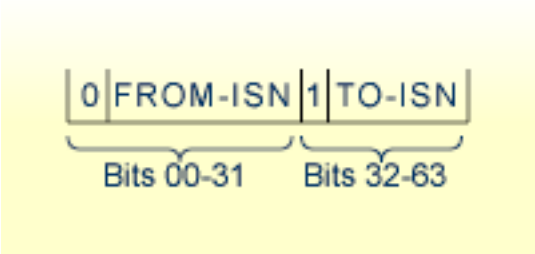
If the first fullword in a record contains a value other than X'00000000' or X'FFFFFFFF', it is assumed to be the 31-bit format. The DDISN/ISN data set can contain records in both formats.

Format 1: 31-Bit Format

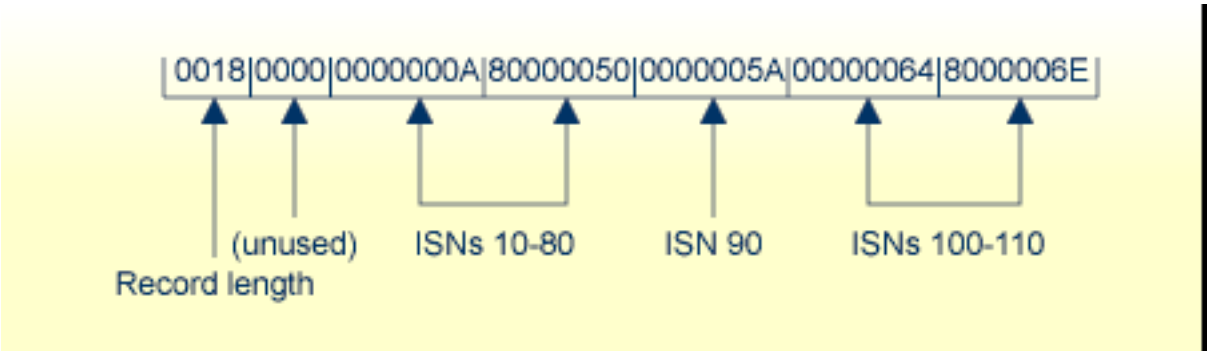
A single ISN requires 4 bytes. Set the high-order bit to 0 and specify the ISN in bits 01-31:



A range of ISNs requires 8 bytes. In the first four bytes, specify the first ISN in the range as a single ISN; in the next four bytes, set the high-order bit to 1 and specify the last ISN:

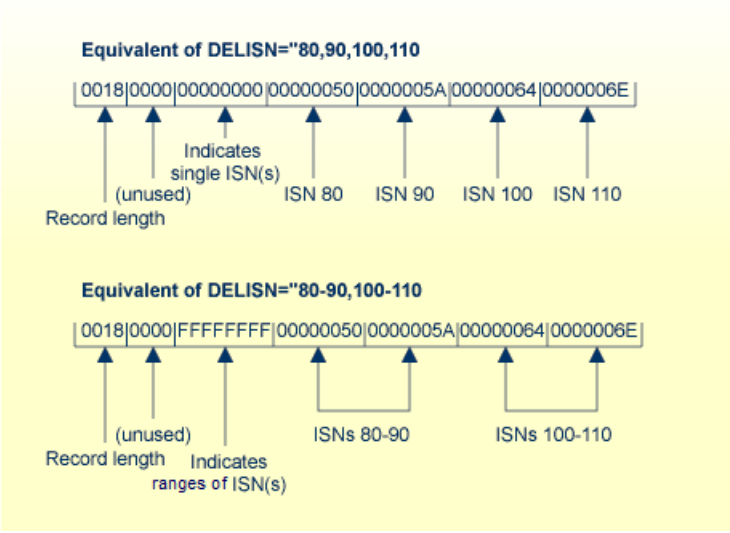


The following example shows a variable-length record containing the equivalent of DELISN=10-80,90,100-110:



Format 2: 32-Bit Format

In the 32-bit format, the first fullword in each record indicates whether the record contains single ISNs or ranges of ISNs. To indicate single ISNs, put zero in the first fullword (X'00000000'). To indicate ranges of ISNs, put -1 (X'FFFFFFFF'). In the following example, the first record contains single ISNs; the second record contains ranges. The two records are identical except for the indicator in the first fullword.



UPDATE Data and Space Requirements

The following general information describes data requirements for UPDATE operation, and how ADALOD UPDATE allocates space. For more information about space allocation, refer to the *Adabas DBA Reference* documentation.

Input Data for UPDATE Operations

Records to be added must be in the form of compressed data records produced by the ADACMP or ADAULD utility. The field definitions used for the ADACMP run must agree with the definitions for the file to which the records will be added as contained in the field definition table (FDT).



Note: Records being added to a ciphered file must already be encrypted using the same cipher code as was used for the records already in the file.

The ISNs of records to be deleted may be provided with the DELISN parameter and/or in an input data set. If provided in an input data set, each ISN must be provided as a 4-byte binary number. The data set must have the record format VARIABLE BLOCKED. If desired, all ISNs to be added to or deleted from an Adabas expanded file can be specified; the UPDATE function selects the appropriate ISNs for the component file being processed.

UPDATE Space Allocation

If records are to be added and a larger MAXISN value has been specified, an additional address converter extent will be allocated by ADALOD. The size of the new extent is based on the difference between the new MAXISN and the previous MAXISN setting. If either insufficient space is available for the new extent or the maximum number of extents has already been allocated, processing ends with an error message.

If an additional Data Storage extent is required, ADALOD allocates an additional extent equal to approximately 25 percent of the total size of the Data Storage extents currently allocated to the file. As for the address converter, processing ends with an error message if either sufficient space is not available for the added extent or the maximum number of extents has already been allocated.

Generating UPDATE Descriptor Information

When adding records, ADALOD UPDATE generates a list of all descriptor values and the corresponding ISNs of the new records, and writes this information to the temp data set.

Associator Updating with UPDATE

Before processing the input, ADALOD UPDATE copies the file's existing normal index to the temp data set, but removes the descriptor values of any ISNs to be deleted.

ADALOD sorts the information written to temp during the input phase and merges the sorted values with the current normal index. The normal index is reordered during this process, and the Associator block padding factor is reestablished for each block. A new upper index is then created.

Empty space in partially filled blocks resulting from descriptor updating is reused. This can increase the number of empty blocks at the end of the index. Although one or more normal index and/or upper index extents may become empty as the result of the reorder process, ADALOD does not condense, delete, or change the size of these extents.

If new free space is needed for the normal index or upper index, ADALOD allocates an additional extent (or extents). Each additional extent allocated is equal to approximately 25 percent

of the total current space allocated to the index. If insufficient space is available for the additional extent or if the maximum number of extents has already been allocated, ADALOD terminates with an error message.

Mass Updates of Expanded Files

Using ADALOD UPDATE for a mass update to an expanded file, records must be added to or deleted from each component file individually. However, each component file can be processed using the same ADALOD commands.

When deleting a record with DELISN or DDISN, the complete list of ISNs to be deleted from all component files can be supplied. ADALOD automatically selects only the ISN values from the specified range that is appropriate for the component file currently being processed.

The same is true when adding new records with USERISN=YES.

When new expanded file records are being added with USERISN=NO but no free ISN is found, the loader cannot allocate a new address converter extent since the ISN range cannot be increased (NOACEXTENSION is active for all component files). Instead, ADALOD creates the index as though end-of-file had been reached. The remaining records not loaded may be added later to another component file using the SKIPREC parameter.

ADALOD does not check for unique descriptor values across component file boundaries.

Example:

The following is an example for performing a mass update to an expanded file (only the relevant parts of the complete jobs are shown):

```
              .
              .
//DDEBAND    DD DSN=MOREDATA.LOAD.PART1-2,...
//DDKARTE    DD *
ADALOD UPDATE FILE=40,USERISN=YES
ADALOD        DELISN=9000001-9500000,12000001-14000000
ADALOD        SORTSIZE=...,TEMPSIZE=...
              .
              .
//DDEBAND    DD DSN=MOREDATA.LOAD.PART1-2,...
//DDKARTE    DD *
ADALOD LOAD   FILE=41,USERISN=YES
ADALOD        DELISN=9000001-9500000,12000001-14000000
ADALOD        SORTSIZE=...,TEMPSIZE=...
              .
              -
```

Enhanced Temp and Sort Processing

The TEMPSIZE and SORTSIZE parameters are now optional. When omitted, various enhancements to TEMP/SORT processing are available to the user. See *Enhancements to TEMP and SORT Processing in ADAINV, ADALOD and ADAULD* for details.

115 Loader Storage Requirements and Use

Static Storage

Static	Type*	Size
Modules ADARUN, ADALOD	A	approximately 180 kilobytes

Dynamic Storage

Dynamic	Type*	Size
Sort work pool	A	LWP
General work pool	A	6 • (Associator block size)
I/O buffer for Associator	A	Associator block size
ISN pool	A	LIP
I/O buffer for Data Storage	A	Data Storage block size
AC bitmap	A	4K bytes
I/O buffer for temp	A	temp block size
DVT splitting	A	temp block size • number of descriptors
Internal descriptor table	A	number of descriptors • 74
I/O buffer for DDEBAND	O	DDEBAND block size
I/O buffer for DDOLD	O	DDOLD block size
I/O buffer for DDISN data	O	DDISN block size
I/O buffer if records must be written to temp overflow	O	DDFILEA block size

** Type A is always used; type O is used only if needed.*

116 TEMP Data Set Space Usage

- Sequential TEMP Data Set 848

ADALOD uses the TEMP data set to store the following information:

- restart information;
- Data Storage RABN/ISN for each record to be deleted (UPDATE only);
- contents of the normal index at the start of the operation (UPDATE only);
- descriptor values obtained from the input data set;
- ADAM overflow area (ADAM files only).

Sequential TEMP Data Set

If the TEMP data set is filled while collecting descriptor values from the input data set, ADALOD temporarily writes the remaining descriptors to the sequential temp file DDFILEA (if specified in the JCL). The descriptors are later read back in when the new index is built.

If actually called, DDFILEA makes ADALOD operation considerably slower than specifying a TEMP data set that is large enough to hold all descriptor values. The DDFILEA TEMP data set should normally be used only as a safety net to ensure adequate space for all descriptors during ADALOD operation. Specifying the DDFILEA TEMP file therefore avoids an ADALOD abend caused by a temp area overrun.



Notes:

1. ADALOD writes only descriptor values from the DDEBAND input file to DDFILEA.
2. The normal TEMP data set must be large enough to hold all values for each single descriptor.
3. The estimated number of TEMP blocks for ADALOD UPDATE may need to be adjusted upwards if forward index compression is in effect for the file since the NI is written to the TEMP data set without forward compression.

If you are running ADALOD UPDATE only to delete ISNs, the size of the TEMP data set must be calculated as the sum of the calculations in the following three steps:

1. Part 1 of TEMP data set contains:
   - RABN 1 : information for restart
   - RABN 2 : FCB of the file
   - RABN 3 - n : DETAB (count of DEs * 116). For example, one element is 116 bytes. The count of DEs is the number of DEs in the FDT.
2. Part 2 of the TEMP data set contains the list of ISNs to be deleted, plus the belonging DS-RABNs, 8 bytes per element. Calculate:

```
(8 * to-be-deleted-ISNs) / (TEMPBLKSIZE - 16) = needed-blocks
```

3. Part 3 contains the complete NI:

```
count NI-BLKS * ASSOBLKSIZE) / (TEMPBLKSIZE - 16) = needed blocks
```

117 ADALOD Space/Statistics Report

During LOAD or UPDATE operation, ADALOD prints a report on the message output data set DDDRUCK. The report shows the following information:

- ADALOD function executed (LOAD or UPDATE), and the database/file affected;
- Estimated NI/UI sizes (shown for the LOAD function only if the NI/UISIZE parameters were not specified);
- Available and used file space, by Adabas component (shown for the LOAD function only);
- Current RABNs assigned for the file (shown for the LOAD function only);
- For spanned record files, the number INPUT RECORDS PROCESSED includes all physical record segments;
- For spanned record files, the number RECORDS WRITTEN is the number of logical records; and
- File processing statistics (records processed and system storage used).

Example of the ADALOD LOAD report:

```
  PARAMETERS:ADALOD  LOAD FILE...
        .
        .

  FUNCTION TO BE EXECUTED:

LOAD FILE NUMBER    7  (MYOWNFILE)
INTO DATABASE    0013  (MYBESTDB)

AVAILABLE SPACE:

(LOAD function only)
```

```
        I FILE I  DEV   I  NUMBER OF  I      FROM      TO    I
        I LAY- I  TYPE  I   BLOCKS    I      RABN      RABN  I
        I OUT  I        I             I                      I
        I------I--------I-------------I----------------------I
        I ASSO I  3380  I      2695   I       137      2831  I
        I DATA I  3380  I      1339   I         3      1341  I
        ------------------------------------------------------


ESTIMATED NORMAL INDEX SIZE = 37 BLOCKS
ESTIMATED UPPER INDEX SIZE = 8 BLOCKS

TOP ISN = 773, MAX ISN EXPECTED = 1335

I FILE  I  DEV   I  LIST  I   ALLOC    I    FROM      TO  I   UNUSED  I
I LAY-  I  TYPE  I  TYPE  I   SPACE    I    RABN    RABN  I   SPACE   I
I OUT   I        I        I  (BLOCKS)  I                  I  (BLOCKS) I
-------------------------------------------------------------------------
I ASSO  I  3380  I   AC   I         2  I      137     138 I          0 I
I ASSO  I  3380  I   UI   I         8  I      139     146 I          0 I
I ASSO  I  3380  I   NI   I        37  I      147     183 I         15 I
I DATA  I  3380  I   DS   I        60  I        3      62 I         48 I
-------------------------------------------------------------------------

   PROCESSING STATISTICS


            773 INPUT RECORDS PROCESSED
            773 RECORDS WRITTEN
             14 BLOCKS USED ON TEMP-DATASET (0%)
              0 BLOCKS USED ON SORT PART 1  (0%)
              0 BLOCKS USED ON SORT PART 2  (0%)
          51824 BYTES OF STORAGE USED TO STORE RECORDS
```

118 JCL Requirements and Examples

- Collation with User Exit 854
- Requirements to Run ADALOD JCL 854

This section describes the job control information required to run ADALOD on the z/OS operating systems and shows examples of job streams.

Note: When running with the optional Recovery Aid (RLOG), all temporary data sets must also be cataloged in the job control.

Collation with User Exit

If a collation user exit is to be used during ADALOD execution, the ADARUN CDXnn parameter must be specified for the utility run.

Used in conjunction with the universal encoding support (UES), the format of the collation descriptor user exit parameter is

```
ADARUN CDXnn= exit-name
```

where

nn	is the number of the collation descriptor exit, a two-digit decimal integer in the range 01-08 inclusive.
exit-name	is the name of the user routine that gets control at the collation descriptor exit; the name can be up to 8 characters long.

Only one program may be specified for each collation descriptor exit. Up to 8 collation descriptor exits may be specified (in any order). See the *Adabas DBA Reference* documentation for more information.

Requirements to Run ADALOD JCL

Data Set	DD Name	Storage	More Information
Associator	DDASSORn	disk	
Data Storage	DDDATARn	disk	
Work	DDWORKR1 DDWORKR4	disk	Required only if Adabas nucleus is not active
Temp area	DDTEMPR1	disk	

Data Set	DD Name	Storage	More Information
Temp overflow (optional)	DDFILEA	disk/ tape	Stores descriptor values if the temp data set is too small
Sort area	DDSORTR1	disk	
Sort area	DDSORTR2	disk	Optional additional SORT area [1]
Recovery log (RLOG)	DDRLOGR1	disk	Required for the recovery log option
Compressed data	DDEBAND	disk/ tape	Output of ADACMP or ADAULD utility
ISNs to be deleted	DDISN	disk/ tape	ISNs to be deleted [2]
Deleted records	DDOLD	disk/ tape	Deleted records, if any [3]
ADARUN parameters	DDCARD	reader	*Operations*
ADALOD parameters	DDKARTE	reader	
ADARUN messages	DDPRINT	printer	ADALOD report, see also *Messages and Codes*
ADALOD messages	DDDRUCK	printer	*Messages and Codes*

Notes:

1. 1. A second sort data set can optionally be defined using DD DDSORTR2. If DDSORTR2 is specified, both sort data sets must be on the same device type (SORTDEV parameter). If the SORTSIZE parameter is specified, both sort data sets must be exactly half the size specified in the SORTSIZE parameter. If the SORTSIZE parameter is not specified, and DDSORTR1 and DDSORTR2 fill up, an attempt to allocate additional space at the end of DDSORTR2 will be made.
2. Four bytes per ISN, RECFM=VB, BLKSIZE as in sequential file description, LRECL maximum equals BLKSIZE - 4.
3. RECFM=VB, BLKSIZE as in sequential file description, LRECL maximum equals BLKSIZE - 4.

ADALOD JCL Examples

Refer also to ADALODE, ADALODA, ADALODM, and ADALODV in the JOBS data set for additional ADALOD examples on loading an ADAM file or the Adabas demo files.

Load File

Refer to ADALOD in the JOBS data set for this example.

```
//ADALOD    JOB
//*
//*    ADALOD: LOAD FILE
//*
//LOD      EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD      <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <=== WORK
//DDTEMPR1  DD   DISP=OLD,DSN=EXAMPLE.DByyyyy.TEMPR1 <=== TEMP
//DDSORTR1  DD   DISP=OLD,DSN=EXAMPLE.DByyyyy.SORTR1 <=== SORT
//DDEBAND   DD   DISP=OLD,DSN=EXAMPLE.DByyyyy.DDEBAND <=== INPUT
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADALOD,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADALOD LOAD FILE=1
ADALOD NAME='TESTFILE-1'
ADALOD MAXISN=10000,DSSIZE=10
ADALOD TEMPSIZE=100,SORTSIZE=100
/*
```

Load File Using Unformatted Temporary Data Sets for TEMP and SORT

Refer to ADALODTM in the JOBS data set for this example.

```
//LOD      EXEC PGM=ADARUN
//STEPLIB  DD  DISP=SHR,DSN=ADABAS.ADAvrs.LOAD           <=== LOAD
//*
//DDASSOR1 DD  DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1       <=== ASSO
//DDDATAR1 DD  DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1       <=== DATA
//DDWORKR1 DD  DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1       <=== WORK
//DDSORTR1 DD  DISP=NEW,UNIT=SYSDA,SPACE=(CYL,(100,80)),
//             DSN=&&SORTR1                              <=== SORT
//DDTEMPR1 DD  DISP=NEW,UNIT=SYSDA,SPACE=(CYL,(100,80)),
//             DSN=&&TEMPR1                              <=== TEMP
//DDEBAND  DD  DISP=OLD,DSN=EXAMPLE.DByyyyy.DDEBAND      <=== INPUT
//DDDRUCK  DD  SYSOUT=*
//DDPRINT  DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//DDCARD   DD  *
ADARUN PROG=ADALOD,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADALOD LOAD FILE=1
ADALOD      NAME='TESTFILE-1'
ADALOD      MAXISN=10000,DSSIZE=10
/*
```

The TEMP and SORT data sets each have an initial allocation of 100 cylinders. If either TEMP or SORT fills up during processing, additional extents of 80 cylinders will be allocated and the job will continue processing.

Update

Refer to ADALODMU in the JOBS data set for this example.

```
//ADALODMU  JOB
//*
//*      ADALOD: MASS UPDATE
//*
//LOD       EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD              <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1         <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1         <=== DATA
//DDTEMPR1  DD   DISP=OLD,DSN=EXAMPLE.DByyyyy.TEMPR1         <=== TEMP
//DDSORTR1  DD   DISP=OLD,DSN=EXAMPLE.DByyyyy.SORTR1         <=== SORT
//DDEBAND   DD   DISP=OLD,DSN=EXAMPLE.DByyyyy.DDEBAND        <=== INPUT
//DDISN     DD   DISP=OLD,DSN=EXAMPLE.DByyyyy.DDISN          <=== ISNS TO DEL
//DDOLD     DD   DISP=(NEW,CATLG),DSN=EXAMPLE.DByyyyy.DDOLD, <=== DEL REC
//            SPACE=(TRK,(100,20),RLSE),UNIT=DISK,VOL=SER=VOLvvv
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADALOD,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADALOD UPDATE FILE=1,LWP=400K,SAVEDREC
ADALOD TEMPSIZE=100,SORTSIZE=100
ADALOD DELISN=100-199,230,301-399
/*
```

XII ADAMER Utility: ADAM Estimation

This chapter covers the following topics:

Functional Overview
Estimate ADAM Access Requirements
ADAMER Output Report Description
JCL Requirements and Examples

119 Functional Overview

The ADAMER utility produces statistics that indicate the number of Data Storage accesses required to find and read a record when using an ADAM descriptor. This information is used to determine

- whether usage of the ADAM option would reduce the number of accesses required to retrieve a record using an ADAM descriptor as opposed to the standard Adabas accessing method;
- the amount of Data Storage space required to produce an optimum distribution of records based on the randomization of the ADAM descriptor.

The input data for ADAMER is a data set containing the compressed records of a file produced by the ADACMP or ADAULD utility.

The field to be used as the ADAM descriptor is specified with the ADAMDE parameter. A multiple value field or a field contained within a periodic group may not be used. The ISN assigned to the record may be used instead of a descriptor as the basis for randomization (ADAMDE=ISN parameter).

The ADAM descriptor must contain a different value in each record, since the file cannot be successfully loaded with the ADAM option of the ADALOD utility if duplicate values are present for the ADAM descriptor. The ADAMER utility requires a descriptor field defined as unique (UQ), but does not check for unique values; checking for unique descriptor values is done by the ADALOD utility when loading the file as an ADAM file.

The BITRANGE parameter may be used to specify that a given number of bits are to be truncated from each ADAM descriptor value before the value is used as input to the randomization algorithm. This permits records containing ADAM descriptor values beginning with the same value (for example, 40643210, 40643220, 40643344) to be loaded into the same physical block in Data Storage. This technique can be used to optimize sequential reading of the file when using the ADAM descriptor to control the read sequence, or to remove insignificant information such as a check digit.

120 Estimate ADAM Access Requirements

- Essential Parameters 864
- Optional Parameters 864
- Examples 866

```
ADAMER  ADAMDE = { descriptor | ISN }
        MAXISN = maximum-number-of-records
        [BITRANGE = { minimum | 0 } {, maximum | 18 } {, increment | 2 } ]
        [DATADEV = device-type ]
        [DATAPFAC = padding-factor ]
        [DATASIZE = minimum , maximum [, increment ] ]
        [NOUSERABEND]
        [NUMREC = number-of-records ]
```

This chapter describes the syntax and parameters of the ADAMER utility.

Essential Parameters

ADAMDE: ADAM Key

Specifies the descriptor to be used as the ADAM key. If ISN is specified, ADAMER uses the ISN of each input record as input for the randomization algorithm.

The ADAM descriptor must be found in the field definition table (FDT) and be defined as a unique descriptor (UQ). It cannot be a sub-, super-, hyper-, collation, or phonetic descriptor. The descriptor also cannot specify the NU option, cannot be an MU field or a field within a periodic group, and cannot be a variable-length field.

MAXISN: Highest ISN to be Allocated for the File

The total number of records expected to be contained in the file.

MAXISN should include the number of records to be originally loaded plus the number of records that are likely to be added to the file.

Optional Parameters

BITRANGE: Bit Truncation for ADAM Key

The minimum, maximum, and incremental number of bits to be truncated from each ADAM descriptor value before the value is used as input to the ADAM randomization algorithm. Bits are always truncated from the rightmost portion of the compressed value.

A maximum of 20 different bit truncations is permitted for each ADAMER execution.

Example:

The following specification results in the truncation of 0 bits, 2 bits, and 4 bits for each Data Storage size for which statistics are provided.

```
BITRANGE=0,4,2 ↩
```

If this parameter is omitted, a default BITRANGE equal to 0,18,2 is used.

DATADEV: Data Storage Device Type

The device type to be used for Data Storage. If DATADEV is not specified, the device type specified by the ADARUN DEVICE parameter is the default.

DATAPFAC: Data Storage Padding Factor

The Data Storage padding factor to be used for the file. The number specified represents the percent of each Data Storage physical block that is not to be used during initial file loading. A value in the range 1-90 may be specified.

If this parameter is omitted, a padding factor of 10 percent is used during ADAMER execution.

DATASIZE: Data Storage Sizes for ADAM Estimates

The Data Storage sizes, in cylinders, for which ADAM statistics are to be provided. A maximum of four Data Storage sizes can be calculated per ADAM execution. The minimum and maximum values may be specified without the increment. ADAMER calculates two increments to produce a report based on all four values.

Example:

The following specification results in statistics for Data Storage sizes of 100, 125, 150, and 175 cylinders.

```
DATASIZE=100,175,25
```

If DATASIZE is omitted, ADAMER provides statistics for four Data Storage sizes as follows:

Size 1:	The first 100 input records are read and the Data Storage size requirement is based on the ADAM descriptor values present in these records and the value specified for MAXISN. The resulting Data Storage size is used as Data Storage Size 1.
Size 2:	Data Storage Size 1 x 1.33.
Size 3:	Data Storage Size 2 x 1.33.
Size 4:	Data Storage Size 3 x 1.33.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

NUMREC: Maximum Number of Records to Read

The maximum number of records to be read from the input file. If NUMREC is not specified, *all* records are read.

Examples

Example 1:

```
ADAMER ADAMDE=CC,
ADAMER DATADEV=3390,DATASIZE=50,110,20,
ADAMER DATAPFAC=10,MAXISN=225000,BITRANGE=2,6,1
```

The ADAM descriptor is CC. Device type 3390 is to be used for Data Storage. Statistics for Data Storage sizes of 50, 70, 90, and 110 cylinders are to be provided. Data Storage padding factor of 10 percent is to be used. The planned number of records for the file is 225000. For each Data Storage size, statistics are to be provided for bit truncations of 2, 3, 4, 5, and 6 bits.

Example 2:

```
ADAMER ADAMDE=CD,DATADEV=3380,DATAPFAC=5,MAXISN=80000
```

The ADAM descriptor is CD. Device type 3380 is to be used for Data Storage. Data Storage padding factor of 5 percent is to be used. The planned number of records for the file is 80,000. Default values are to be used for all other parameters.

121 ADAMER Output Report Description

The following entries appear on the report produced by ADAMER:

Field	Explanation
LOADISNS	Number of records contained in the input data set.
MAXISN	Total file records (see the MAXISN parameter description).
DATA DEVICE	Data Storage device type (see the DATADEV parameter description).
DATAPFAC	Data Storage padding factor (see DATAPFAC parameter description).

The following fields appear under "AVERAGE NUMBER OF EXCPs":

Field	Explanation
Data Storage SIZE	See the DATASIZE parameter description. The number of cylinders is rounded up to the nearest integer.
BIT-PARM	See the BITRANGE parameter description.
FOR LOADISNS	The average number of I/Os required to find and read a record when the ADAM descriptor is used. This result assumes that the number of records in the file is equal to the number of records contained in the input data set.
DISK USAGE	The percentage of Data Storage space occupied after initial loading of the file. This result assumes that the number of records to be loaded is equal to the number of records contained in the input data set.
FOR MAXISN	The average number of I/Os required to find and read a record when using the ADAM descriptor. This result assumes that the number of records in the file is equal to the value specified with the MAXISN parameter.
DISK USAGE	The percentage of Data Storage space occupied after initial loading of the file. This result assumes that the number of records to be loaded is equal to the number of records specified with the MAXISN parameter.

Using the information contained on the ADAMER report, the user can determine

- the optimum balance between access and Data Storage space requirements; and
- the optimum number of bits that should be truncated from each ADAM descriptor value so that records containing similar beginning values are loaded into the same physical block. This is necessary only if optimization of sequential reading is desired.

122 JCL Requirements and Examples

- Requirements to Run ADAMER JCL 893

This section describes the job control information required to run ADAMER on the z/OS operating system and shows an example job stream.

Requirements to Run ADAMER JCL

Data Set	DD Name	Storage	More Information
Input data	DDEBAND	tape/ disk	Output of ADACMP or ADAULD utility
ADARUN parameters	DDCARD	reader	*Operations*
ADAMER parameters	DDKARTE	reader	
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADAMER messages/report	DDDRUCK	printer	*Messages and Codes*

ADAMER JCL Example

Refer to ADAMER in the JOBS data set for this example.

```
//ADAMER    JOB
//*
//*   ADAMER:
//*      ADAM ESTIMATION
//*
//MER      EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD       <=== ADABAS LOAD
//*
//DDEBAND   DD   DISP=OLD,DSN=EXAMPLE.DByyyyy.COMPR1 <=== COMPRESS DATA
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADAMER,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADAMER MAXISN=1000,ADAMDE=AA,BITRANGE=0,2,4
ADAMER DATADEV=eeee,DATAPFAC=10,DATASIZE=100,175,25
/*
```

XIII ADAORD Utility: Reordering Functions

This chapter covers the following topics:

Functional Overview
REORASSO: Reorder Associator
REORDATA: Reorder Data Storage
REORDB: Reorder Database
REORFASSO: Reorder Associator for a Single File
REORFDATA: Reorder Data Storage for a Single File
REORFILE: Reorder File
RESTRUCTUREDB: Restructure Database
RESTRUCTUREF: Restructure Single Files
STORE: Store Files
JCL Requirements and Examples

123 Functional Overview

- Reorder Functions 874
- Restructure Functions 875
- Store Function 875
- Space Allocation 876
- Adabas 8 Considerations 876

The ADAORD utility can be used to:

- Reorder the Associator or Data Storage for a database or a single file in a database (REORASSO, REORFASSO, REORDATA, REORFDATA, REORDB, and REORFILE functions)
- Restructure a database or a single file in a database and store the resulting output files into an existing database (RESTRUCTUREDB, RESTRUCTUREF, and STORE functions).

Only one function may be executed during a given execution of ADAORD.

Parts of the database are overwritten during ADAORD execution. We therefore recommend that you back up the database (or file) using the ADASAV utility before running ADAORD functions.

In addition, all ADAORD functions except RESTRUCTUREF (file) require exclusive EXF control of the database files involved in the operation. RESTRUCTUREF requires EXU control; other users may access database files being used by RESTRUCTUREF, but only for reading. Note, however, that operations involving checkpoint, security, or files loaded using ADALOD's SYSFILE option require exclusive database control.

Notes:

1. When specifying the starting RABN for Associator extents, the space needed for the FCBs, FDTs, and DSST should also be considered.
2. Logically deleted fields will be present in ADAORD utility output.
3. Logically deleted field data in the file is loaded by the ADAORD STORE utility function.

Reorder Functions

The REORASSO function physically reorders all Associator blocks for all files; the REORFASSO function reorders the Associator for a single file. This eliminates Associator space fragmentation and combines multiple address converter, normal and upper index, and Data Storage Space Table (DSST) component extents into a single logical extent for each component.

The REORDATA function reorders Data Storage for all files in the database; the REORFDATA function reorders Data Storage for a single file. This condenses extents containing only empty blocks, and also eliminates any Data Storage fragmentation caused by file deletion.

The REORDB function performs both the REORASSO and REORDATA functions in a single execution of ADAORD.

The REORFILE function performs both the REORFASSO and REORFDATA functions in a single execution of ADAORD. The records may be reordered in the logical sequence by a descriptor, by ISN, or in the current sequence.

The REORDATA, REORDB, REORFDATA and STORE functions do not reorder ADAM files. However, these functions can be used to relocate an ADAM file to different RABNs.

Restructure Functions

The RESTRUCTUREDB function unloads an entire database to a sequential data set; the RESTRUCTUREF function unloads one or more files to a sequential data set. This data set containing unloaded data can be used as input to the STORE function.

The RESTRUCTURE functions are used to relocate the database to a different physical device or a file or files to another device.

The format of the sequential data set produced by the RESTRUCTURE functions is independent of the database device type, and is *not* compatible with the format required by the ADALOD or ADASAV utilities. Therefore, the target database may be contained on a device type different from the source database.

The Associator and Data Storage are reordered as part of RESTRUCTURE/STORE processing.

When RESTRUCTUREDB/F restructures an ADAM file that uses the overflow area, and then STORE stores the restructured file in a database with a smaller DATA block size, an ADAORD ERROR-103 may occur. Use the ADAULD and ADALOD utilities to move ADAM files instead.

Store Function

The STORE function loads one or more files into an existing database using the DDFILEA output created by the RESTRUCTUREDB, RESTRUCTUREF, or REORDB function.

The Associator and Data Storage are reordered as part of RESTRUCTURE/STORE processing.

The STORE function does not reorder ADAM files. However, it can be used, in combination with other ADAORD functions, to relocate an ADAM file to different RABNs. When the RESTRUCTUREDB or RESTRUCTUREF functions restructure an ADAM file that uses the overflow area, and then STORE stores the restructured file in a database with a smaller DATA block size, an ADAORD ERROR-103 may occur. Use the ADAULD and ADALOD utilities to move ADAM files instead.



Note: Logically deleted field data in the file is loaded by the ADAORD STORE utility function.

Space Allocation

ADAORD allocates the amount of space required by the *xx*SIZE or MAXISN and MAXISN2 parameters, if specified. Otherwise, ADAORD allocates space based on the current size of the file. Note that the *xx*RELEASE parameters affect the amount of space required.

If possible, space is allocated on the volume specified by the *xxxx*VOLUME parameter. If insufficient free space is available on the specified volume, ADAORD allocates the remainder of the required space on other volumes, according to its default rules of allocation.

An *xx*RABN parameter overrides the associated *xxxx*VOLUME parameter.

Adabas 8 Considerations

You can restructure databases and files from an Adabas version prior to Adabas 8 and store them in an Adabas 8 (or later) database using ADAORD STORE. However, you cannot store the restructured output of an Adabas 8 database or file in a database running with any prior Adabas version (for example, Adabas 7). If you attempt this, the following warning will be generated and ADAORD will end with a CC=4:

```
*** Warning: The input data set is from V8 and will not be processed
```

124 REORASSO: Reorder Associator

- Optional Parameters and Subparameters 879
- Examples 883

The REORASSO function reorders the entire Associator. If a file is not explicitly specified, its related Associator information is reordered according to its existing definition. To reorder Associator information for specific files, use the REORFASSO function.

This function requires exclusive EXF control of the database files involved in the operation. In addition, parts of the database are overwritten during ADAORD execution, so we recommend that you back up the database (or file) using the ADASAV utility first, before running ADAORD functions.

If the file specified for this function was originally loaded with ISNREUSE=YES active, this reorder function will reset the first unused ISN value in that file's control block (FCB) to the actual first unused ISN found in the address converter.

This is the syntax of the ADAORD REORASSO function:

```
ADAORD  REORASSO  [DBINDEXCOMPRESSION = { YES | NO } ]
                  [FILE = file-number ]
                      [ACRABN = starting-rabn ]
                      [AC2RABN = starting-rabn ]
                      [ALLOCATION = { FORCE | NOFORCE } ]
                      [ASSOPFAC = padding-factor  ]
                      [ASSOVOLUME = 'Associator-extent-volume ' ]
                      [INDEXCOMPRESSION = { YES | NO } ]
                      [ISNSIZE = { 3 | 4 } ]
                      [MAXISN = highest-isn ]
                      [MAXISN2 = highest-isn ]
                      [NIRABN = starting-rabn ]
                      [NIRELEASE]
                      [NISIZE = size ]
                      [UIRABN = starting-rabn ]
                      [UIRELEASE]
                      [UISIZE = size ]
                  [LPB = prefetch-buffer-size ]
                  [MAXFILES = maximum-number-files ]
                  [NEWDBID = database-identifier ]
                  [NEWDBNAME = database-name ]
                  [NOUSERABEND]
                  [RAID]
                  [RPLUPDATEONLY = { YES | NO ]
                  [TEST]
```

Note: If the parameter MAXFILES or NEWDBID is specified, an active nucleus will terminate automatically at the end of the REORASSO function.

Optional Parameters and Subparameters

ACRABN: Starting RABN for Address Converter

The beginning RABN for the file's address converter extent. If this parameter is omitted, ADAORD assigns the starting RABN. The space requested must be available in one extent.

When specifying the starting RABN for Associator extents, the space needed for the FCBs, FDTs, and DSST should also be considered.

AC2RABN: Starting RABN for Secondary Address Converter

The beginning RABN for the file's secondary address converter extent. The secondary address converter is used to map the secondary ISNs of secondary spanned records to the RABNs of the Data Storage blocks where the secondary records are stored.

If this parameter is omitted, ADAORD assigns the starting RABN. The space requested must be available in one extent. If the file contains no secondary address converter extents, this parameter is ignored.

When specifying the starting RABN for Associator extents, the space needed for the FCBs, FDTs, and DSST should also be considered.

ALLOCATION: Action to Follow File Extent Allocation Failure

ALLOCATION specifies the action to be taken if file extent allocations cannot be obtained according to the placement parameters ACRABN, NIRABN, or UIRABN.

By default (that is, ALLOCATION=FORCE), the utility terminates with error if any file extent allocation cannot be met according to RABN placement parameters.

If ALLOCATION=NOFORCE is specified and any allocation with placement parameters fails, the utility retries the allocation without the placement parameter.

ASSOPFAC: Associator Padding Factor

The new Associator block padding factor. The number specified represents the percentage of each Associator block not to be used during the reorder process. A value in the range 1-90 may be specified. The remaining number of bytes after padding must be greater than the largest descriptor value plus 10.

If this parameter is omitted, the current Associator padding factor in effect for the file is used.

ASSOVOLUME: Associator Extent Volume

Note: The value for the ASSOVOLUME parameter must be enclosed in apostrophes.

ASSOVOLUME identifies the volume on which the corresponding file's Associator space (that is, the AC, NI, and UI extents) should be allocated. If the requested number of blocks cannot be found on the specified volume, ADAORD allocates the remaining blocks on other volumes according to its default rules of allocation.

If ACRABN, UIRABN, or NIRABN is specified, ADAORD ignores the ASSOVOLUME value when allocating the corresponding extent type.

If ASSOVOLUME is not specified, the file's Associator space is allocated according to ADAORD's default allocation rules.

DBINDEXCOMPRESSION: Compress Database Indexes

DBINDEXCOMPRESSION indicates whether the indexes of files are rebuilt in compressed or uncompressed form. It applies to all files for which no INDEXCOMPRESSION parameter is specified.

DBINDEXCOMPRESSION can be used to build compressed or uncompressed indexes for all files of the database, making it unnecessary to specify index compression for each file.

FILE: File Number

The file number to which the following parameters apply. Each specified file and its parameters should be on a separate ADAORD statement following the ADAORD REORASSO function statement.

For any file whose number is not specified, current Associator block padding factor and MAXISN value are retained, and all Associator space allocations remain the same.

INDEXCOMPRESSION: Compress File Index

INDEXCOMPRESSION indicates whether the index for the file is rebuilt in compressed or uncompressed form. A compressed index usually requires less index space and improves the efficiency of index operations in the Adabas nucleus.

If INDEXCOMPRESSION is *not* specified

- but the DBINDEXCOMPRESSION parameter is specified for the database as a whole, the default is the database value.
- and DBINDEXCOMPRESSION is also *not* specified, the default is the current compression form of the file.

ISNSIZE: 3- or 4-Byte ISN

ISNSIZE specifies whether ISNs in the file are to be 3 or 4 bytes long. The default is the value currently used for the file; this value is stored in the file control block (FCB).



Note: It is not possible to change the ISNSIZE of a physically coupled file using ADAORD.

LPB: Prefetch Buffer Size

Specifies the size, in bytes, of the internal prefetch buffer. The maximum value is 32760 bytes. The default depends on the ADARUN LU parameter. ADAORD may reduce a specified LPB value if the LU value is too small.

MAXFILES: Maximum Number of Files

MAXFILES specifies the maximum number of files that can be loaded into the database. The minimum value for this parameter is 3. The highest value permitted is 5000 or one less than the ASSOR1 block size, whichever is lower. For example, 2543 is the highest MAXFILES value for a database whose ASSOR1 is stored on a 3390 DASD.

If this parameter is omitted, the current value for MAXFILES is retained.

When MAXFILES is specified, the nucleus terminates after the ADAORD REORASSO function is completed.

MAXISN: Highest ISN Permitted for the File

MAXISN specifies the highest ISN that can be allocated for the file. This value must be greater than the current TOPISN value displayed in the ADAREP database report.

ADAORD uses the specified value to calculate the address converter space required. If this parameter is omitted, the current MAXISN value for the file is retained.

MAXISN2: Highest Secondary ISN Permitted for the File

MAXISN specifies the desired size of the secondary address converter (AC2) in ISNs. This value must be greater than the current TOP AC2 ISN value displayed in the ADAREP database report. The secondary address converter is used to map the secondary ISNs of secondary spanned records to the RABNs of the Data Storage blocks where the secondary records are stored.

ADAORD uses the specified value to calculate the space required in the secondary address converter for the file. If this parameter is omitted, the current MAXISN2 value for the file is retained. If the file contains no secondary address converter extents, this parameter is ignored.

NEWDBID: Database Identifier

NEWDBID is the ID to be assigned to the database. A value in the range 1-65535 may be used. For systems using Online System Security, the value 999 is reserved. If this parameter is omitted, the current database ID is retained.

When NEWDBID is specified, the nucleus terminates after the ADAORD REORASSO function is completed.

NEWDBNAME: Database Name

The name to be assigned to the database. The name assigned may be from 1 to 16 characters. If this parameter is omitted, the current database name is retained.

If the database name contains special characters or embedded blanks, the name must be enclosed within apostrophes ('...'), which themselves must be doubled if included in the name; for example, 'JAN''S DB'.

NIRABN: Starting RABN for Normal Index

NIRABN specifies the beginning RABN number for the normal index extent. If this parameter is omitted, ADAORD assigns the starting RABN.

NIRELEASE: Release Unused Normal Index Blocks

Specifying NIRELEASE releases unused normal index (NI) blocks belonging to the specified file. If NIRELEASE is not specified, ADAORD allocates *at least* the number of NI blocks that were allocated before the file was reordered.



Note: Adabas calculates the file extent size using any changed padding factor or block size values *before* the file is reordered.

NISIZE: Normal Index Size

NISIZE is the number of blocks or cylinders to be allocated for the normal index. If the value is blocks, it must be followed by a "B" (for example, "2000B").

If this parameter is omitted, ADAORD computes the file extent size in proportion to any increase or decrease in the ASSOPFAC padding factor.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

RAID: Action to Follow Determination That File Does Not Exist

The RAID parameter instructs ADAORD to ignore any FILE parameters that refer to a file that does not exist in the database.

If RAID is not specified (the default), ADAORD terminates with an error message when it encounters a FILE parameter referring to a file that does not exist in the database.

The RAID parameter is provided for use in recovery jobs built by the Adabas Recovery Aid (ADARAI).

RPLUPDATEONLY: Allow Only Event Replicator Processing Updates

The `RPLUPDATEONLY` parameter can be used in the ADAORD REORASSO function to indicate whether this Adabas database file may be updated only by the Event Replicator Server as part of Adabas-to-Adabas replication or by other means as well. This parameter is optional. Valid values are "YES" or "NO". A value of "YES" indicates that the file can only be updated via Event Replicator processing; a value of NO indicates that the file can be updated by any normal means, including Event Replicator processing. There is no default; if no value is specified for

the RPLUPDATEONLY parameter in the ADAORD REORASSO function, the value used previously for the file is used.

TEST: Test Syntax

This parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested.

UIRABN: Starting RABN for Upper Index

UIRABN is the beginning RABN number for the file's upper index extent. If this parameter is omitted, ADAORD assigns the starting RABN for each of these extents.

UIRELEASE: Release Unused Upper Index Blocks

Specifying UIRELEASE releases unused upper index (UI) blocks belonging to the specified file. If UIRELEASE is not specified, ADAORD allocates *at least* the number of UI blocks that were allocated before the file was reordered.



Note: Adabas calculates the file extent size using any changed padding factor or block size values *before* the file is reordered.

UISIZE: Upper Index Size

UISIZE is the number of blocks or cylinders to be allocated for the upper index. If the value is blocks, it must be followed by a "B" (for example, "2000B").

If this parameter is omitted, ADAORD computes the file extent size in proportion to any increase or decrease in the ASSOPFAC padding factor.

Examples

Example 1:

```
ADAORD REORASSO ↩
```

The Associator is to be reordered.

Example 2:

```
ADAORD   REORASSO
ADAORD      MAXFILES=200
ADAORD      NEWDBID=6,NEWDBNAME=DATABASE-6
```

The Associator is to be reordered. A maximum of 200 files are permitted for the database. The database ID and name are to be 6 and DATABASE-6, respectively.

Example 3:

```
ADAORD   REORASSO
ADAORD      FILE=1,ACRABN=1000,NIRABN=2200,
```

```
ADAORD      FILE=2,MAXISN=500000,
ADAORD      FILE=4,ASSOPFAC=5
```

The Associator is to be reordered. The address converter allocation for file 1 is to begin with RABN 1000. The normal index for file 1 is to begin with RABN 2200. The MAXISN for file 2 is to be set to 500000. The Associator block padding factor for file 4 is to be set to 5 percent. The Associator information for all other database files is reordered according to each file's current definition.

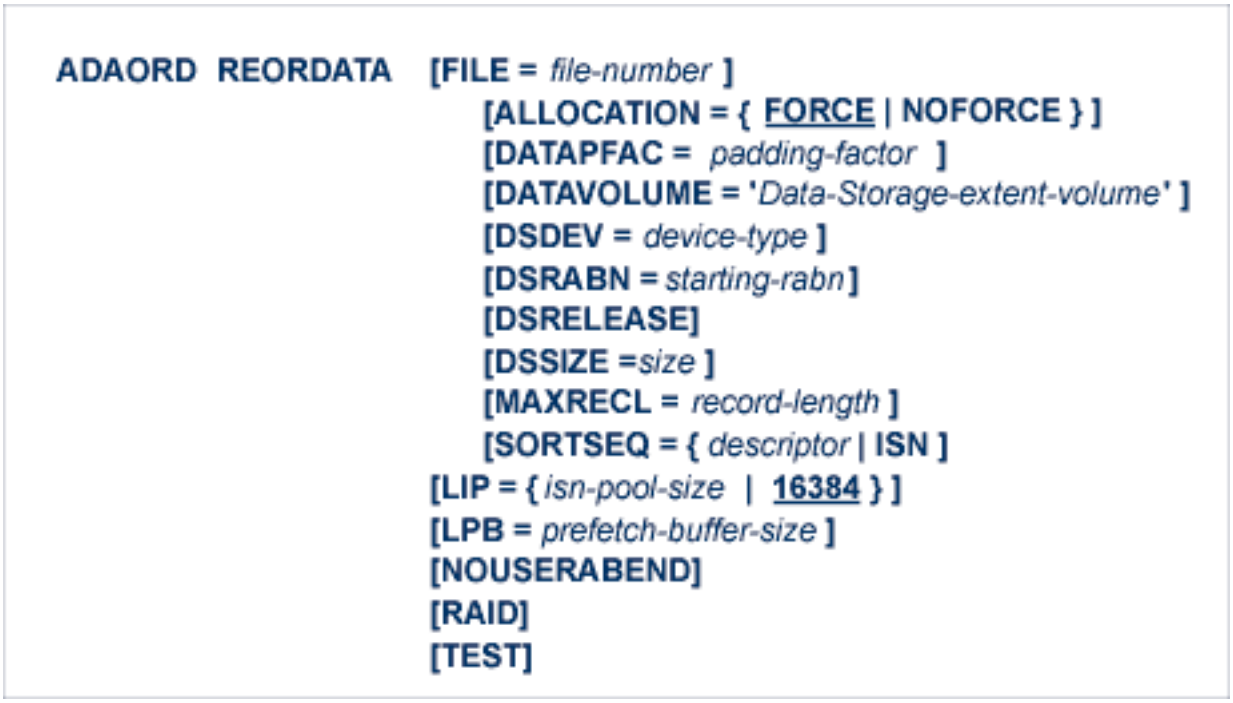
125 REORDATA: Reorder Data Storage

- Optional Parameters and Their Subparameters 886
- Examples 889

The REORDATA function reorders Data Storage for *all* files. Files not specified are reordered according to their existing definitions. The REORDATA function does not reorder ADAM files. However, it can be used to relocate an ADAM file to different RABNs.

This function requires exclusive EXF control of the database files involved in the operation. In addition, parts of the database are overwritten during ADAORD execution, so we recommend that you back up the database (or file) using the ADASAV utility first, before running ADAORD functions.

This is the syntax of the ADAORD REORDATA function:



Optional Parameters and Their Subparameters

ALLOCATION: Action to Follow File Extent Allocation Failure

ALLOCATION specifies the action to be taken if file extent allocations cannot be obtained according to the placement parameter DSRABN.

By default (that is, ALLOCATION=FORCE), the utility terminates with error if any file extent allocation cannot be met according to RABN placement parameters.

If ALLOCATION=NOFORCE is specified and any allocation with placement parameter fails, the utility retries the allocation without the placement parameter.

DATAPFAC: Data Storage Padding Factor

DATAPFAC specifies the new Data Storage padding factor, which is the percentage of each Data Storage block reserved for record expansion when the file is reordered. A value in the range 1-90 may be specified (see the ADALOD LOAD DATAPFAC parameter discussion for more information about setting the padding factor). If this parameter is omitted, the current padding factor for the file is used.

DATAVOLUME: Data Storage Extent Volume



Note: The value for the DATAVOLUME parameter must be enclosed in apostrophes.

DATAVOLUME specifies the volume on which the file's Data Storage space (DS extents) are allocated. If the number of blocks requested with DSSIZE cannot be found on the specified volume, ADAORD allocates the remaining blocks on other volumes according to its default allocation rules.

If DSRABN is specified, DATAVOLUME is ignored for the related file.

If DATAVOLUME is not specified, the Data Storage space is allocated based on the current size of the file. The DSRELEASE parameter also affects the amount of space required.

DSDEV: Data Storage Device Type

DSDEV is the file's Data Storage device type. The specified device type must already be defined to Adabas, normally when the database was created or by the ADADBS utility's ADD function.

If DSDEV is not specified, ADAORD attempts to allocate the file on the device type used before reordering.

DSRABN: Data Storage Starting RABN

The beginning RABN for the specified file's Data Storage extent. If this parameter is omitted, ADAORD assigns the starting RABN.

DSRELEASE: Release Unused Data Storage Blocks

Specifying DSRELEASE releases unused Data Storage (DS) blocks belonging to the specified file. If DSRELEASE is not specified, ADAORD allocates *at least* the number of DS blocks that were allocated before the file was reordered.



Note: Adabas calculates the file extent size using any changed padding factor or block size values *before* the file is reordered.

DSSIZE: Data Storage Size

DSSIZE is the number of blocks or cylinders to be allocated for the file's Data Storage (DS) logical extent. If the value is blocks, it must be followed by a "B" (for example, "2000B").

If this parameter is omitted, ADAORD computes the file extent size in proportion to any increase or decrease in the DATAPFAC padding factor used.

FILE: File Number

FILE is the file to which the following parameters apply. Each specified file and its parameters should be on a separate ADAORD statement following the ADAORD REORDATA statement.

For any file whose number is not specified, the file is reordered using the current physical sequence, and the current Data Storage padding factor and space allocation are retained.

LIP: ISN Buffer Pool Size

The LIP parameter can be used to decrease the number of Associator I/O operations when recreating the address converter. For best performance, specify a size that accepts all ISNs of the largest file to be processed.

LIP specifies the size of the ISN pool for containing ISNs and their assigned Data Storage RABNs. The value may be specified in bytes as a numeric value ("2048") or in kilobytes as a value followed by a "K" ("2K"). The default for LIP is 16384 bytes (or 16K).

The length of one input record is ISNSIZE + RABNSIZE. Thus the entry length is at least 6 bytes (the ISNSIZE of the file is 3 and the RABNSIZE of the database is 3) and at most 8 bytes (the ISNSIZE is 4 and the RABNSIZE is 4).



Note: When ADAORD is processing files that contain spanned records with secondary ISNs, a second LIP will be allocated to contain these ISNs.

LPB: Prefetch Buffer Size

LPB specifies the size, in bytes, of the internal prefetch buffer. The maximum value is 32760 bytes. The default depends on the ADARUN LU parameter. ADAORD may reduce a specified LPB value if the LU value is too small.

MAXRECL: Maximum Compressed Record Length

Use the MAXRECL parameter to change the maximum record length, after compression, permitted in the file. Specifying MAXRECL has two effects:

- The file's DATA data set is allocated only to devices that support the specified length.
- If the file contains Data Storage records that exceed the specified length, ADAORD abends and prints the ERROR-126 message (Data Storage record too long).



Note: MAXRECL is ignored if specified for files enabled for spanned data storage records.

If MAXRECL is not specified, the maximum compressed record length does not change.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

RAID: Action to Follow Determination That File Does Not Exist

The RAID parameter instructs ADAORD to ignore any FILE parameters that refer to a file that does not exist in the database.

If RAID is not specified (the default), ADAORD terminates with an error message when it encounters a FILE parameter referring to a file that does not exist in the database.

The RAID parameter is provided for use in recovery jobs built by the Adabas Recovery Aid (ADARAI).

SORTSEQ: Record Processing Sequence

SORTSEQ determines the sequence in which the file is to be processed. If this parameter is omitted, the records are processed in physical sequence.

If a descriptor is specified, the file is processed in the logical sequence of the descriptor values. *Do not* use a null-suppressed descriptor field, a hyperdescriptor, a phonetic descriptor, a multiple-value field, or a descriptor contained in a periodic group.



Note: Even when the descriptor field is not null suppressed, the record is *not* represented in the inverted list if the descriptor field or a field following it has never been initialized (held a value). Therefore, the record will be dropped when the utility is executed.

If ISN is specified, the file is processed in ascending ISN sequence. For the Adabas checkpoint or security file, only SORTSEQ=ISN is allowed.

TEST: Test Syntax

This parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

Examples

Example 1:

```
ADAORD REORDATA
```

Data Storage for the entire database is to be reordered.

Example 2:

```
ADAORD   REORDATA
ADAORD              FILE=1,DSRABN=1000,DSSIZE=200B
ADAORD              FILE=4,SORTSEQ=AA
ADAORD              FILE=5,DATAPFAC=15
```

Data Storage is to be reordered. Data Storage for file 1 is to begin with RABN 1000 with 200 blocks to be allocated. File 4 is to be reordered using descriptor AA for sequence control. The Data Storage block padding factor for file 5 is to be set to 15 percent. All other database files are reordered according to their existing definitions.

126 REORDB: Reorder Database

- Optional Parameters and Subparameters 894
- Examples 900

The REORDB function reorders the entire Associator and Data Storage for a database. Files that are not specified are reordered according to their existing definitions.

The REORDB function does not reorder ADAM files. However, it can be used to relocate an ADAM file to different RABNs.

This function requires exclusive EXF control of the database files involved in the operation. In addition, parts of the database are overwritten during ADAORD execution, so we recommend that you back up the database (or file) using the ADASAV utility first, before running ADAORD functions.

If the file specified for this function was originally loaded with ISNREUSE=YES active, this reorder function will reset the first unused ISN value in that file's control block (FCB) to the actual first unused ISN found in the address converter.

This is the syntax of the ADAORD REORDB function:

```
ADAORD  REORDB    [DBINDEXCOMPRESSION = { YES | NO } ]
                  [FILE = file-number ]
                      [ACRABN = starting-rabn ]
                      [AC2RABN = starting-rabn ]
                      [ALLOCATION = { FORCE | NOFORCE } ]
                      [ASSOPFAC = padding-factor  ]
                      [ASSOVOLUME = 'Associator-extent-volume ' ]
                      [DATAPFAC =  padding-factor  ]
                      [DATAVOLUME = 'Data-Storage-extent-volume' ]
                      [DSDEV = device-type ]
                      [DSRABN = starting-rabn]
                      [DSRELEASE]
                      [DSSIZE =size ]
                      [INDEXCOMPRESSION = { YES | NO } ]
                      [ISNSIZE = { 3 | 4 } ]
                      [MAXISN = highest-isn ]
                      [MAXISN2 = highest-isn ]
                      [MAXRECL = record-length ]
                      [NIRABN = starting-rabn]
                      [NIRELEASE]
                      [NISIZE =  size ]
                      [SORTSEQ = { descriptor | ISN ]
                      [UIRABN = starting-rabn ]
                      [UIRELEASE]
                      [UISIZE =  size ]
                  [LIP = { isn-pool-size  |  16384 } ]
                  [LPB = prefetch-buffer-size ]
                  [MAXFILES =  maximum-number-files ]
                  [NEWDBID = database-identifier ]
                  [NEWDBNAME = database-name]
                  [NOUSERABEND]
                  [RAID]
                  [TEST]
```

Note: If the parameter MAXFILES or NEWDBID is specified, the nucleus automatically terminates at the end of the REORDB function.

Optional Parameters and Subparameters

ACRABN: Starting RABN for Address Converter

The RABN with which the file's address converter extent is to begin. If this parameter is omitted, ADAORD assigns the starting RABN. The space requested must be available in one extent.

When specifying the starting RABN for Associator extents, the space needed for the FCBs, FDTs, and DSST should also be considered.

AC2RABN: Starting RABN for Secondary Address Converter

The beginning RABN for the file's secondary address converter extent. The secondary address converter is used to map the secondary ISNs of secondary spanned records to the RABNs of the Data Storage blocks where the secondary records are stored.

If this parameter is omitted, ADAORD assigns the starting RABN. The space requested must be available in one extent. If the file contains no secondary address converter extents, this parameter is ignored.

ALLOCATION: Action to Follow File Extent Allocation Failure

ALLOCATION specifies the action to be taken if file extent allocations cannot be obtained according to the placement parameters ACRABN, DSRABN, NIRABN, or UIRABN.

By default (that is, ALLOCATION=FORCE), the utility terminates with error if any file extent allocation cannot be met according to RABN placement parameters.

If ALLOCATION=NOFORCE is specified and any allocation with placement parameters fails, the utility retries the allocation without the placement parameter.

ASSOPFAC: Associator Padding Factor

The new Associator block padding factor. The number specified represents the percentage of each Associator block not to be used during the reorder process. A value in the range 1-90 may be specified. The remaining number of bytes after padding must be greater than the largest descriptor value plus 10.

If this parameter is omitted, the current Associator padding factor in effect for the file is used.

ASSOVOLUME: Associator Extent Volume

Note: The value for the ASSOVOLUME parameter must be enclosed in apostrophes.

ASSOVOLUME identifies the volume on which the file's Associator space (that is, the AC, NI, and UI extents) should be allocated. If the requested number of blocks cannot be found on the specified volume, ADAORD allocates the remaining blocks on other volumes according to its default allocation rules.

If ACRABN, UIRABN, or NIRABN is specified, ADAORD ignores the ASSOVOLUME value when allocating the corresponding extent type.

If ASSOVOLUME is not specified, the file's Associator space is allocated according to ADAORD's default allocation rules.

DATAPFAC: Data Storage Padding Factor

DATAPFAC specifies the new Data Storage padding factor, which is the percentage of each Data Storage block reserved for record expansion when the file is reordered. A value in the range 1-90 may be specified (see the ADALOD LOAD DATAPFAC parameter discussion for more information about setting the padding factor). If this parameter is omitted, the current padding factor for the file is used.

DATAVOLUME: Data Storage Extent Volume



Note: The value for the DATAVOLUME parameter must be enclosed in apostrophes.

DATAVOLUME specifies the volume on which the file's Data Storage space (DS extents) are allocated. If the requested number of blocks requested with DSSIZE cannot be found on the specified volume, ADAORD allocates the remaining blocks on other volumes according to its default allocation rules.

If DSRABN is specified, DATAVOLUME is ignored for the related file.

If DATAVOLUME is not specified, the Data Storage space is allocated according to ADAORD's default allocation rules.

DBINDEXCOMPRESSION: Compress Database Indexes

DBINDEXCOMPRESSION indicates whether the indexes of files are rebuilt in compressed or uncompressed form. It applies to all files for which no INDEXCOMPRESSION parameter is specified.

DBINDEXCOMPRESSION can be used to build compressed or uncompressed indexes for all files of the database, making it unnecessary to specify index compression for each file.

DSDEV: Data Storage Device Type

DSDEV is the file's Data Storage device type. The specified device type must already be defined to Adabas, normally when the database was created or by the ADADBS utility's ADD function.

If DSDEV is not specified, ADAORD attempts to allocate the file on the device type used before reordering.

DSRABN: Data Storage Starting RABN

The beginning RABN for the file's Data Storage extent. If this parameter is omitted, ADAORD assigns the starting RABN.

DSRELEASE: Release Unused Data Storage Blocks

Specifying DSRELEASE releases unused Data Storage (DS) blocks belonging to the specified file. If DSRELEASE is not specified, ADAORD allocates *at least* the number of DS blocks that were allocated before the file was reordered.

Note: Adabas calculates the file extent size using any changed padding factor or block size values *before* the file is reordered.

DSSIZE: Data Storage Size

DSSIZE is the number of blocks or cylinders to be allocated for the file's Data Storage (DS) logical extent. If the value is blocks, it must be followed by a "B" (for example, "2000B").

If this parameter is omitted, ADAORD computes the file extent size in proportion to any increase or decrease in the DATAPFAC padding factor.

FILE: File Number

The file to which the following parameters apply. Each specified file must be on a separate ADAORD statement following the ADAORD REORDB function statement, and must be immediately followed by the applicable parameters for the file.

For any file whose number is not specified, the current Associator and Data Storage block padding factors and MAXISN value are retained, and all Associator and Data Storage space allocations remain the same.

INDEXCOMPRESSION: Compress File Index

INDEXCOMPRESSION indicates whether the index for the file is rebuilt in compressed or uncompressed form. A compressed index usually requires less index space and improves the efficiency of index operations in the Adabas nucleus.

If INDEXCOMPRESSION is *not* specified

- but the DBINDEXCOMPRESSION parameter is specified for the database as a whole, the default is the database value.
- and DBINDEXCOMPRESSION is also *not* specified, the default is the current compression form of the file.

ISNSIZE: 3- or 4-Byte ISN

ISNSIZE specifies whether ISNs in the file are to be 3 or 4 bytes long. The default is the value currently used for the file; this value is stored in the file control block (FCB).

Note: It is not possible to change the ISNSIZE of a physically coupled file using ADAORD.

LIP: ISN Buffer Pool Size

The LIP parameter can be used to decrease the number of Associator I/O operations when recreating the address converter. For best performance, specify a size that accepts all ISNs of the largest file to be processed.

LIP specifies the size of the ISN pool for containing ISNs and their assigned Data Storage RABNs. The value may be specified in bytes as a numeric value ("2048") or in kilobytes as a value followed by a "K" ("2K"). The default for LIP is 16384 bytes (or 16K).

The length of one input record is ISNSIZE + RABNSIZE. Thus the entry length is at least 6 bytes (the ISNSIZE of the file is 3 and the RABNSIZE of the database is 3) and at most 8 bytes (the ISNSIZE is 4 and the RABNSIZE is 4).



Note: When ADAORD is processing files that contain spanned records with secondary ISNs, a second LIP will be allocated to contain these ISNs.

LPB: Prefetch Buffer Size

LPB specifies the size, in bytes, of the internal prefetch buffer. The maximum value is 32760 bytes. The default depends on the ADARUN LU parameter. ADAORD may reduce a specified LPB value if the LU value is too small.

MAXFILES: Maximum Number of Files

MAXFILES specifies the maximum number of files that can be loaded into the database. The minimum value for this parameter is 3. The highest value permitted is 5000 or one less than the ASSOR1 block size, whichever is lower. For example, 2543 is the highest MAXFILES value for a database whose ASSOR1 is stored on a 3390 DASD.

If this parameter is omitted, the current value for MAXFILES is retained.

When MAXFILES is specified, the nucleus terminates after the ADAORD REORDB function is completed.

MAXISN: Highest ISN Permitted in the File

The highest ISN that can be allocated for the file. This value must be greater than the current TOPISN value displayed in the ADAREP database report.

ADAORD uses the specified value to calculate the address converter space required. If this parameter is omitted, the current MAXISN value for the file is retained.

MAXISN2: Highest Secondary ISN Permitted for the File

MAXISN specifies the desired size of the secondary address converter (AC2) in ISNs. This value must be greater than the current TOP AC2 ISN value displayed in the ADAREP database report. The secondary address converter is used to map the secondary ISNs of secondary spanned records to the RABNs of the Data Storage blocks where the secondary records are stored.

ADAORD uses the specified value to calculate the space required in the secondary address converter for the file. If this parameter is omitted, the current MAXISN2 value for the file is retained. If the file contains no secondary address converter extents, this parameter is ignored.

MAXRECL: Maximum Compressed Record Length

Use the MAXRECL parameter to change the maximum record length, after compression, permitted in the file. Specifying MAXRECL has two effects:

- The file's DATA data set is allocated only to devices that support the specified length.
- If the file contains Data Storage records that exceed the specified length, ADAORD abends and prints the ERROR-126 message (Data Storage record too long).

Note: MAXRECL is ignored if specified for files enabled for spanned data storage records.

If MAXRECL is not specified, the maximum compressed record length does not change.

NEWDBID: Database Identifier

NEWDBID is the ID to be assigned to the database. A value in the range 1-64535 may be used. For systems using Adabas Online System Security, the value 999 is reserved. If this parameter is omitted, the current database ID is retained.

When NEWDBID is specified, the nucleus terminates after the ADAORD REORDB function is completed.

NEWDBNAME: Database Name

NEWDBNAME specifies the name to be assigned to the database. The name can contain up to 16 characters. If the name contains special characters or embedded blanks, it must be enclosed in apostrophes ('...'); for example, 'JAN''S DB'. If this parameter is omitted, the current database name is retained.

NIRABN: Starting RABN for Normal Index

NIRABN specifies the RABN with which the file's normal index extent is to begin. If this parameter is omitted, ADAORD assigns the starting RABN.

NIRELEASE: Release Unused Normal Index Blocks

Specifying NIRELEASE releases unused normal index (NI) blocks belonging to the specified file. If NIRELEASE is not specified, ADAORD allocates *at least* the number of NI blocks that were allocated before the file was reordered.

Note: Adabas calculates the file extent size using any changed padding factor or block size values *before* the file is reordered.

NISIZE: Normal Index Size

NISIZE specifies the number of blocks or cylinders to be allocated for the file's normal index. A block count must be followed by a "B" (for example, "2000B").

If this parameter is omitted, ADAORD computes the file extent size in proportion to any increase or decrease in the ASSOPFAC padding factor.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

RAID: Action to Follow Determination That File Does Not Exist

The RAID parameter instructs ADAORD to ignore any FILE parameters that refer to a file that does not exist in the database.

If RAID is not specified (the default), ADAORD terminates with an error message when it encounters a FILE parameter referring to a file that does not exist in the database.

The RAID parameter is provided for use in recovery jobs built by the Adabas Recovery Aid (ADARAI).

SORTSEQ: File Processing Sequence

SORTSEQ determines the sequence in which the file is to be processed. If this parameter is omitted, the records are processed in physical sequence.

If a descriptor is specified, the file is processed in the logical sequence of the descriptor values. *Do not* use a null-suppressed descriptor field, a hyperdescriptor, a phonetic descriptor, a multiple-value field, or a descriptor contained in a periodic group.



Note: Even when the descriptor field is not null suppressed, the record is *not* represented in the inverted list if the descriptor field or a field following it has never been initialized (held a value). Therefore, the record will be dropped when the utility is executed.

If ISN is specified, the file is processed in ascending ISN sequence. For the Adabas checkpoint or security file, only SORTSEQ=ISN is allowed.

TEST: Test Syntax

This parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

UIRABN: Starting RABN for Upper Index

UIRABN defines the beginning RABN for the Associator's upper index extent for the file. If this parameter is omitted, ADAORD assigns the starting RABN.

UIRELEASE: Release Unused Upper Index Blocks

UIRELEASE releases unused upper index (UI) blocks belonging to the file. If UIRELEASE is not specified, ADAORD allocates *at least* the number of UI blocks that were allocated before the file was reordered.



Note: Adabas calculates the file extent size using any changed padding factor or block size values *before* the file is reordered.

UISIZE: Upper Index Size

UISIZE specifies the number of blocks or cylinders to allocate for the upper index. A block count must be followed by a "B" (for example, "2000B"). If UISIZE is omitted, ADAORD allocates space in proportion to an increase or decrease in the ASSOPFAC padding factor.

Examples

Example 1:

```
ADAORD   REORDB
```

The Associator and Data Storage are to be reordered. No changes are to be made to the current database parameters.

Example 2:

```
ADAORD   REORDB
ADAORD          MAXFILES=200
ADAORD          NEWDBID=6,NEWDBNAME=DATABASE-6
```

The Associator and Data Storage are to be reordered. A maximum of 200 files are permitted for the database. The database ID and name are to be 6 and DATABASE-6, respectively. If the nucleus is active during the REORDB operation, it will be stopped following the operation (NEWDBID was specified).

Example 3:

```
ADAORD   REORDB
ADAORD          FILE=1,ACRABN=1000,NIRABN=2200,SORTSEQ=ISN
ADAORD          FILE=2,MAXISN=500000
ADAORD          FILE=4,ASSOPFAC=5,DATAPFAC=20,DSSIZE=5,DSRABN=1
```

The Associator and Data Storage are to be reordered. The address converter allocation for file 1 is to begin with RABN 1000. The normal index allocation for file 1 is to begin with RABN 2200.

The Data Storage portion of file 1 is to be reordered in ascending ISN sequence. The MAXISN for file 2 is to be set to 500000. The following assignments are made for file 4: the Associator block padding factor is to be changed to 5 percent, the Data Storage block padding factor is set to 20 per cent, and a new DSSIZE of 5 cylinders is assigned starting at RABN 1. All other files are reordered according to their existing definitions.

Example 4:

```
ADAORD REORDB
ADAORD FILE=66
ADAORD DSRELEASE
ADAORD NIRELEASE
ADAORD UIRELEASE
ADAORD RAID
```

1. ADAORD reorders the entire database.
2. ADAORD releases all unused storage from the Data Storage, normal index, and upper index of file 66.

3. However, if file 66 does not exist in the database, ADAORD does not terminate with an error message; rather, ADAORD ignores this condition and proceeds.

Example 5:

```
ADAORD REORDB
ADAORD DBINDEXCOMPRESSION=YES
ADAORD FILE=1
ADAORD FILE=2,INDEXCOMPRESSION=NO
ADAORD FILE=3
```

All files are reordered and rebuilt with compressed indexes, except for file 2, which is rebuilt with an uncompressed index.

127 REORFASSO: Reorder Associator for a Single File

- Essential Parameter 905
- Optional Parameters 905
- Examples 908

The REORFASSO function reorders the Associator for a single file. Associator information for unspecified files is not reordered.

This function requires exclusive EXU control of the database files involved in the operation. In addition, parts of the database are overwritten during ADAORD execution, so we recommend that you back up the database (or file) using the ADASAV utility first, before running ADAORD functions.

If the file specified for this function was originally loaded with ISNREUSE=YES active, this reorder function will reset the first unused ISN value in that file's control block (FCB) to the actual first unused ISN found in the address converter.

This is the syntax of the ADAORD REORFASSO function:

```
ADAORD REORFASSO FILE = file-number
                        [ACRABN = starting-rabn ]
                        [AC2RABN = starting-rabn ]
                        [ALLOCATION = { FORCE | NOFORCE } ]
                        [ASSOPFAC = padding-factor  ]
                        [ASSOVOLUME = 'Associator-extent-volume ' ]
                        [INDEXCOMPRESSION = { YES | NO } ]
                        [ISNSIZE = { 3 | 4 } ]
                        [MAXISN = highest-isn ]
                        [MAXISN2 = highest-isn ]
                        [NIRABN = starting-rabn ]
                        [NIRELEASE]
                        [NISIZE = size ]
                        [PASSWORD = password ]
                        [UIRABN = starting-rabn ]
                        [UIRELEASE]
                        [UISIZE = size ]
                    [EXCLUDE = file-list ]
                    [LPB = prefetch-buffer-size ]
                    [NOUSERABEND]
                    [TEST]
```

Essential Parameter

FILE: File Number

FILE specifies the file to be processed, and to which the parameters that follow in the statement sequence apply. Several files and their related parameters may be specified within one REORFASSO operation; see the examples at the end of this section. If a component file of an Adabas expanded file is specified, only that file's Associator is reordered; this has no adverse effect on the other component files.

Optional Parameters

ACRABN: Starting RABN for Address Converter

ACRABN specifies the file's starting address converter RABN. If this parameter is omitted, ADAORD assigns the starting RABN. The space requested must be available in one extent.

When specifying the starting RABN for Associator extents, the space needed for the FCBs, FDTs, and DSST should also be considered.

AC2RABN: Starting RABN for Secondary Address Converter

The beginning RABN for the file's secondary address converter extent. The secondary address converter is used to map the secondary ISNs of secondary spanned records to the RABNs of the Data Storage blocks where the secondary records are stored.

If this parameter is omitted, ADAORD assigns the starting RABN. The space requested must be available in one extent. If the file contains no secondary address converter extents, this parameter is ignored.

ALLOCATION: Action to Follow File Extent Allocation Failure

ALLOCATION specifies the action to be taken if file extent allocations cannot be obtained according to the placement parameters ACRABN, NIRABN, or UIRABN.

By default (that is, ALLOCATION=FORCE), the utility terminates with error if any file extent allocation cannot be met according to RABN placement parameters.

If ALLOCATION=NOFORCE is specified and any allocation with placement parameters fails, the utility retries the allocation without the placement parameter.

ASSOPFAC: Associator Padding Factor

ASSOPFAC defines the new Associator block padding factor, which is the percentage of each Associator block *not* used during the reorder process. Specify a value in the range 1-90. The number of bytes free after padding must be greater than the largest descriptor value plus 10.

If this parameter is omitted, the current padding factor in effect for the file is used.

ASSOVOLUME: Associator Extent Volume



Note: The value for the ASSOVOLUME parameter must be enclosed in apostrophes.

ASSOVOLUME identifies the volume on which the file's Associator space (that is, the AC, NI, and UI extents) should be allocated. If the requested number of blocks cannot be found on the specified volume, ADAORD allocates the remaining blocks on other volumes according to its default allocation rules.

If ACRABN, UIRABN, or NIRABN is specified, ADAORD ignores the ASSOVOLUME value when allocating the corresponding extent type.

If ASSOVOLUME is not specified, the file's Associator space is allocated according to ADAORD's default allocation rules.

EXCLUDE: Exclude Specified Files from Reorder

EXCLUDE lists the numbers of the files to be excluded from REORDER processing; that is, the files that are not to be reordered.

The parameter is optional: if not specified, no files are excluded. A file number may be listed only once.

Files specified in the EXCLUDE parameter must also be specified in the FILE parameter.

The EXCLUDE parameter is provided for use in recovery jobs built by the Adabas Recovery Aid (ADARAI).

INDEXCOMPRESSION: Compress File Index

INDEXCOMPRESSION indicates whether the index for the file is rebuilt in compressed or uncompressed form. A compressed index usually requires less index space and improves the efficiency of index operations in the Adabas nucleus.

If INDEXCOMPRESSION is not specified, the default is the current form of the file.

ISNSIZE: 3- or 4-Byte ISN

ISNSIZE specifies whether ISNs in the file are to be 3 or 4 bytes long. The default is the value currently used for the file; this value is stored in the file control block (FCB).



Note: It is not possible to change the ISNSIZE of a physically coupled file using ADAORD.

LPB: Prefetch Buffer Size

LPB specifies the size, in bytes, of the internal prefetch buffer. The maximum value is 32760 bytes. The default depends on the ADARUN LU parameter. ADAORD may reduce a specified LPB value if the LU value is too small.

MAXISN: Highest ISN to Be Allocated

MAXISN is the highest ISN which may be allocated for the file. This value must be greater than the current TOPISN value displayed in the ADAREP database report.

ADAORD uses the specified value to calculate the address converter space required. If this parameter is omitted, the current MAXISN value for the file remains in effect.

MAXISN2: Highest Secondary ISN Permitted for the File

MAXISN specifies the desired size of the secondary address converter (AC2) in ISNs. This value must be greater than the current TOP AC2 ISN value displayed in the ADAREP database report. The secondary address converter is used to map the secondary ISNs of secondary spanned records to the RABNs of the Data Storage blocks where the secondary records are stored.

ADAORD uses the specified value to calculate the space required in the secondary address converter for the file. If this parameter is omitted, the current MAXISN2 value for the file is retained. If the file contains no secondary address converter extents, this parameter is ignored.

NIRABN: Starting RABN for Normal Index

NIRABN is the starting RABN to be used for the normal index. If NIRABN is omitted, ADAORD assigns the starting RABN.

NIRELEASE: Release Unused Normal Index Blocks

Specifying NIRELEASE releases unused normal index (NI) blocks belonging to the specified file. If NIRELEASE is not specified, ADAORD allocates *at least* the number of NI blocks that were allocated before the file was reordered.

Note: Adabas calculates the file extent size using any changed padding factor or block size values *before* the file is reordered.

NISIZE: Normal Index Size

NISIZE specifies the number of blocks or cylinders to be allocated for the file's normal index. A block count must be followed by a "B" (for example, "2000B").

If this parameter is omitted, ADAORD computes the file extent size in proportion to any increase or decrease in the ASSOPFAC padding factor.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PASSWORD: File Password

If the file is password-protected, use this parameter to specify the password.

TEST: Test Syntax

This parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

UIRABN: Starting RABN for Upper Index

UIRABN is the starting RABN for the upper index. If this parameter is omitted, ADAORD assigns the starting RABN.

UIRELEASE: Release Unused Upper Index Blocks

Specifying UIRELEASE releases unused upper index (UI) blocks belonging to the specified file. If UIRELEASE is not specified, ADAORD allocates *at least* the number of UI blocks that were allocated before the file was reordered.



Note: Adabas calculates the file extent size using any changed padding factor or block size values *before* the file is reordered.

UISIZE: Upper Index Size

UISIZE specifies the number of blocks or cylinders to be allocated for the file's upper index. A block count must be followed by a "B" (for example, "2000B").

If this parameter is omitted, ADAORD computes the file extent size in proportion to any increase or decrease in the ASSOPFAC padding factor.

Examples

Example 1:

```
ADAORD   REORFASSO   FILE=9
ADAORD                          ASSOPFAC=5
ADAORD                          FILE=23
ADAORD                          UIRABN=3151,UISIZE=50B
ADAORD                          NIRABN=3201
```

The Associator for files 9 and 23 is to be reordered; Associator data for other files is not changed.

The Associator padding factor is set to 5% for file 9. For file 23, the following Associator changes are being made: the new upper index starting RABN is 3151, with a new upper index size of 50 blocks. The new normal index starting RABN is 3201; the normal index size remains the same as before.

Example 2:

```
ADAORD   REORFASSO   FILE=104
ADAORD                          ASSOPFAC=5,NISIZE=5B,UISIZE=2B
ADAORD                          ACRABN=10000,NIRABN=10510,UIRABN=10515
ADAORD                          FILE=105
```

The Associator for files 104 and 105 is to be reordered; Associator information for all other files is unchanged.

For file 104, the Associator padding factor is to be set to 5. The sizes of the normal index and upper index are to be 5 blocks and 2 blocks respectively. The starting RABN for the address converter is to be 10000. The starting RABN for the normal index is to be 10510. The starting RABN for the upper index is to be 10515. Information for file 105 is reordered according to the file's existing definition.

128 REORFDATA: Reorder Data Storage for a Single File

- Essential Parameter 912
- Optional Parameters 913
- Examples 916

The REORFDATA function reorders Data Storage for a single file. Data Storage for unspecified files is not reordered.

The REORFDATA function does not reorder ADAM files. However, it can be used to relocate an ADAM file to different RABNs.

This function requires exclusive EXU control of the database files involved in the operation. In addition, parts of the database are overwritten during ADAORD execution, so we recommend that you back up the database (or file) using the ADASAV utility first, before running ADAORD functions.

This is the syntax of the ADAORD REORFDATA function:

```
ADAORD REORFDATA FILE = file-number
                        [ALLOCATION = { FORCE | NOFORCE } ]
                        [DATAPFAC =  padding-factor  ]
                        [DATAVOLUME = 'Data-Storage-extent-volume' ]
                        [DSDEV = device-type ]
                        [DSRABN = starting-rabn ]
                        [DSRELEASE]
                        [DSSIZE =size ]
                        [MAXRECL = record-length ]
                        [PASSWORD = password ]
                        [SORTSEQ = { descriptor | ISN } ]
                    [EXCLUDE = file-list ]
                    [LIP = { isn-pool-size  |  16384 } ]
                    [LPB = prefetch-buffer-size ]
                    [NOUSERABEND]
                    [TEST]
```

Essential Parameter

FILE: File Number

FILE specifies the file to be processed, and to which the parameters that follow in the statement sequence apply. Several files and their related parameters may be specified within one REORFDATA operation; see the examples at the end of this section.

Optional Parameters

ALLOCATION: Action to Follow File Extent Allocation Failure

ALLOCATION specifies the action to be taken if file extent allocations cannot be obtained according to the placement parameter DSRABN.

By default (that is, ALLOCATION=FORCE), the utility terminates with error if any file extent allocation cannot be met according to RABN placement parameter.

If ALLOCATION=NOFORCE is specified and any allocation with placement parameter fails, the utility retries the allocation without the placement parameter.

DATAPFAC: Data Storage Padding Factor

DATAPFAC specifies the new Data Storage padding factor, which is the percentage of each Data Storage block reserved for record expansion when the file is reordered. A value in the range 1-90 may be specified (see the ADALOD LOAD DATAPFAC parameter discussion for more information about setting the padding factor). If this parameter is omitted, the current padding factor for the file is used.

DATAVOLUME: Data Storage Extent Volume



Note: The value for the DATAVOLUME parameter must be enclosed in apostrophes.

DATAVOLUME specifies the volume on which the file's Data Storage space (DS extents) are allocated. If the number of blocks requested with DSSIZE cannot be found on the specified volume, ADAORD allocates the remaining blocks on other volumes according to its default allocation rules.

If DSRABN is specified, DATAVOLUME is ignored for the related file.

If DATAVOLUME is not specified, the Data Storage space is allocated according to ADAORD's default allocation rules.

DSDEV: Data Storage Device Type

DSDEV is the file's Data Storage device type. The specified device type must already be defined to Adabas, normally when the database was created or by the ADADBS utility's ADD function.

If DSDEV is not specified, ADAORD attempts to allocate the file on the device type used before reordering.

DSRABN: Data Storage Starting RABN

DSRABN is the beginning RABN for the file's Data Storage extent. If this parameter is omitted, ADAORD assigns the starting RABN.

DSRELEASE: Release Unused Data Storage Blocks

Specifying DSRELEASE releases unused Data Storage (DS) blocks belonging to the file. If DSRELEASE is not specified, ADAORD allocates *at least* the number of DS blocks that were allocated before the file was reordered.



Note: Adabas calculates the file extent size using any changed padding factor or block size values *before* the file is reordered.

DSSIZE: Data Storage Size

DSSIZE specifies the number of blocks or cylinders to be allocated for the Data Storage. A block count must be followed by a "B" (for example, "2000B").

If this parameter is omitted, ADAORD computes the file extent size in proportion to any increase or decrease in the DATAPFAC padding factor.

EXCLUDE: Exclude Specified Files from Reorder

EXCLUDE lists the numbers of the files to be excluded from REORDER processing; that is, the files that are not to be reordered.

The parameter is optional: if not specified, no files are excluded. A file number may be listed only once.

Files specified in the EXCLUDE parameter must also be specified in the FILE parameter.

The EXCLUDE parameter is provided for use in recovery jobs built by the Adabas Recovery Aid (ADARAI).

LIP: ISN Buffer Pool Size

The LIP parameter can be used to decrease the number of Associator I/O operations when re-creating the address converter. For best performance, specify a size that accepts all ISNs of the largest file to be processed.

LIP specifies the size of the ISN pool for containing ISNs and their assigned Data Storage RABNs. The value may be specified in bytes as a numeric value ("2048") or in kilobytes as a value followed by a "K" ("2K"). The default for LIP is 16384 bytes (or 16K).

The length of one input record is ISNSIZE + RABNSIZE. Thus the entry length is at least 6 bytes (the ISNSIZE of the file is 3 and the RABNSIZE of the database is 3) and at most 8 bytes (the ISNSIZE is 4 and the RABNSIZE is 4).



Note: When ADAORD is processing files that contain spanned records with secondary ISNs, a second LIP will be allocated to contain these ISNs.

LPB: Prefetch Buffer Size

LPB specifies the size, in bytes, of the internal prefetch buffer. The maximum value is 32760 bytes. The default depends on the ADARUN LU parameter. ADAORD may reduce a specified LPB value if the LU value is too small.

MAXRECL: Maximum Compressed Record Length

Use the MAXRECL parameter to change the maximum record length, after compression, permitted in the file. Specifying MAXRECL has two effects:

- The file's DATA data set is allocated only to devices that support the specified length.
- If the file contains Data Storage records that exceed the specified length, ADAORD abends and prints the ERROR-126 message (Data Storage record too long).



Note: MAXRECL is ignored if specified for files enabled for spanned data storage records.

If MAXRECL is not specified, the maximum compressed record length does not change.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PASSWORD: File Password

If the file is password-protected, use this parameter to specify the password.

SORTSEQ: File Reordering Sequence

SORTSEQ determines the sequence in which the file is to be processed. If this parameter is omitted, the records are processed in physical sequence.

If a descriptor is specified, the file is processed in the logical sequence of the descriptor values. *Do not* use a null-suppressed descriptor field, a hyperdescriptor, a phonetic descriptor, a multiple-value field, or a descriptor contained in a periodic group.



Note: Even when the descriptor field is not null suppressed, the record is *not* represented in the inverted list if the descriptor field or a field following it has never been initialized (held a value). Therefore, the record will be dropped when the utility is executed.

If ISN is specified, the file is processed in ascending ISN sequence. For the Adabas checkpoint or security file, only SORTSEQ=ISN is allowed.

TEST: Test Syntax

This parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested: only the syntax of the specified parameters can be tested.

Examples

Example 1:

```
ADAORD   REORFDATA   FILE=16
```

The Data Storage for file 16 is to be reordered. No other files are affected.

Example 2:

```
ADAORD   REORFDATA   FILE=246
ADAORD              DATAPFAC=5,DSSIZE=10,SORTSEQ=MZ
ADAORD              FILE=247
```

The Data Storage for files 246 and 247 is to be reordered. No other files' Data Storage will be reordered.

For file 246, the Data Storage padding factor is to be set to 5. Data Storage for file 247 is reordered according to the file's existing definition.

129 REORFILE: Reorder File

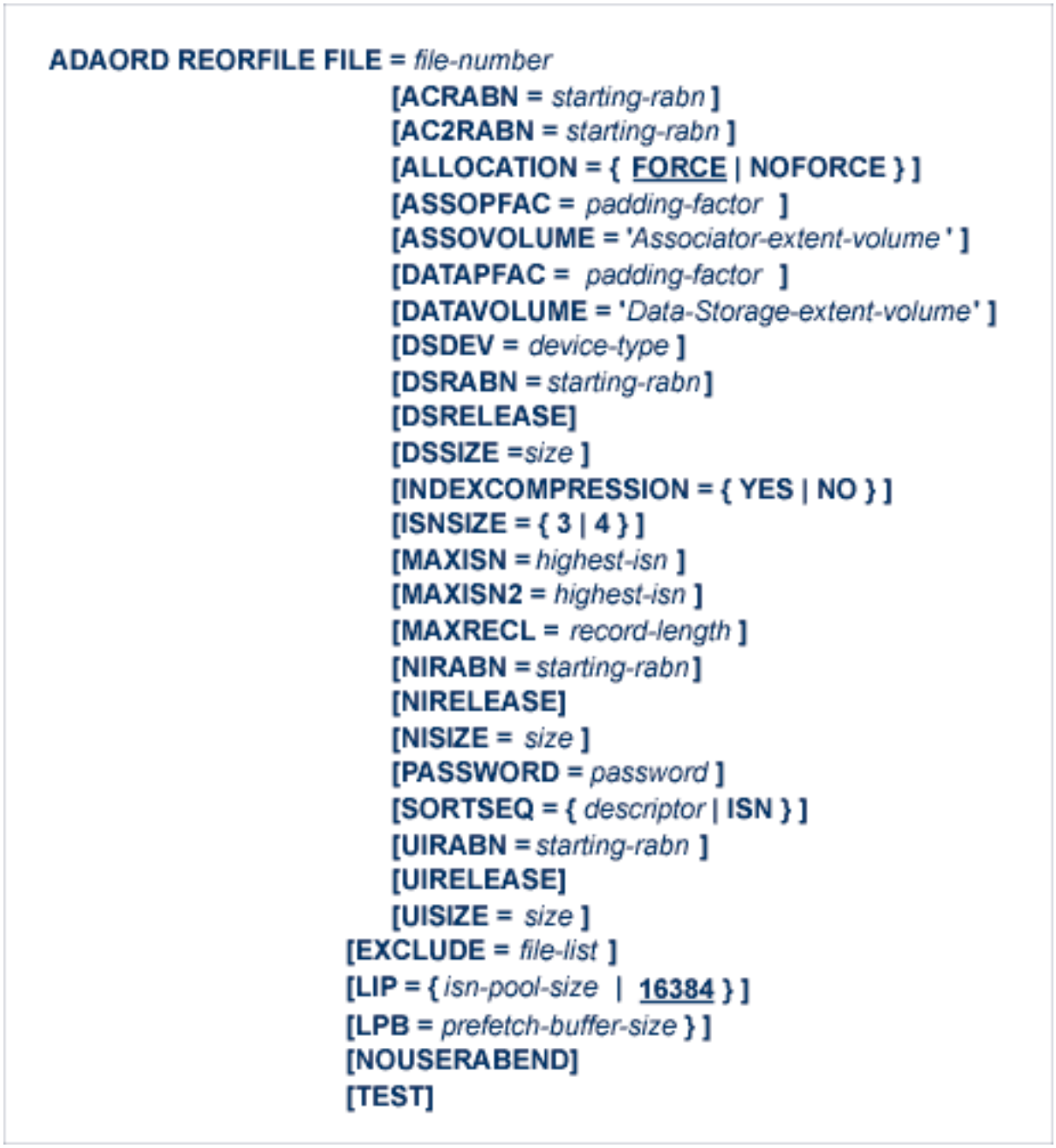
- Essential Parameter 920
- Optional Parameters 920
- Examples 925

The REORFILE function reorders the Associator and Data Storage for a single file. Associator and Data Storage for other files are not affected.

This function requires exclusive EXU control of the database files involved in the operation. In addition, parts of the database are overwritten during ADAORD execution, so we recommend that you back up the database (or file) using the ADASAV utility first, before running ADAORD functions.

If the file specified for this function was originally loaded with ISNREUSE=YES active, this reorder function will reset the first unused ISN value in that file's control block (FCB) to the actual first unused ISN found in the address converter.

This is the syntax of the ADAORD REORFILE function:


ADAORD REORFILE FILE = file-number
[ACRABN = starting-rabn]
[AC2RABN = starting-rabn]
[ALLOCATION = { FORCE | NOFORCE }]
[ASSOPFAC = padding-factor]
[ASSOVOLUME = 'Associator-extent-volume ']
[DATAPFAC = padding-factor]
[DATAVOLUME = 'Data-Storage-extent-volume']
[DSDEV = device-type]
[DSRABN = starting-rabn]
[DSRELEASE]
[DSSIZE =size]
[INDEXCOMPRESSION = { YES | NO }]
[ISNSIZE = { 3 | 4 }]
[MAXISN = highest-isn]
[MAXISN2 = highest-isn]
[MAXRECL = record-length]
[NIRABN = starting-rabn]
[NIRELEASE]
[NISIZE = size]
[PASSWORD = password]
[SORTSEQ = { descriptor | ISN }]
[UIRABN = starting-rabn]
[UIRELEASE]
[UISIZE = size]
[EXCLUDE = file-list]
[LIP = { isn-pool-size | 16384 }]
[LPB = prefetch-buffer-size }]
[NOUSERABEND]
[TEST]

Essential Parameter

FILE: File Number

FILE specifies the file to be processed, and to which the parameters that follow in the statement sequence apply. Several files and their related parameters may be specified within one REORFILE operation; see the examples at the end of this section. If a component file of an Adabas expanded file is specified, only that file's Associator and Data Storage are reordered; this has no adverse effect on the other component files.

Optional Parameters

ACRABN: Starting RABN for Address Converter

ACRABN is the beginning RABN for the file's address converter extent. If this parameter is omitted, ADAORD assigns the starting RABN. The space requested must be available in one extent.

When specifying the starting RABN for Associator extents, the space needed for the FCBs, FDTs, and DSST should also be considered.

AC2RABN: Starting RABN for Secondary Address Converter

The beginning RABN for the file's secondary address converter extent. The secondary address converter is used to map the secondary ISNs of secondary spanned records to the RABNs of the Data Storage blocks where the secondary records are stored.

If this parameter is omitted, ADAORD assigns the starting RABN. The space requested must be available in one extent. If the file contains no secondary address converter extents, this parameter is ignored.

ALLOCATION: Action to Follow File Extent Allocation Failure

ALLOCATION specifies the action to be taken if file extent allocations cannot be obtained according to the placement parameters ACRABN, DSRABN, NIRABN, or UIRABN.

By default (that is, ALLOCATION=FORCE), the utility terminates with error if any file extent allocation cannot be met according to RABN placement parameters.

If ALLOCATION=NOFORCE is specified and any allocation with placement parameters fails, the utility retries the allocation without the placement parameter.

ASSOPFAC: Associator Padding Factor

ASSOPFAC specifies the new Associator block padding factor. The number specified represents the percentage of each Associator block not to be used during the reorder process. A value in the range 1-90 may be specified. The remaining number of bytes after padding must be greater than the largest descriptor value plus 10.

If this parameter is omitted, the current Associator padding factor in effect for the file is used.

ASSOVOLUME: Associator Extent Volume

Note: The value for the ASSOVOLUME parameter must be enclosed in apostrophes.

ASSOVOLUME identifies the volume on which the file's Associator space (that is, the AC, NI, and UI extents) should be allocated. If the requested number of blocks cannot be found on the specified volume, ADAORD allocates the remaining blocks on other volumes according to its default allocation rules.

If ACRABN, UIRABN, or NIRABN is specified, ADAORD ignores the ASSOVOLUME value when allocating the corresponding extent type.

If ASSOVOLUME is not specified, the file's Associator space is allocated according to ADAORD's default allocation rules.

DATAPFAC: Data Storage Padding Factor

DATAPFAC specifies the new Data Storage padding factor, which is the percentage of each Data Storage block reserved for record expansion when the file is reordered. A value in the range 1-90 may be specified (see the ADALOD LOAD DATAPFAC parameter discussion for more information about setting the padding factor). If this parameter is omitted, the current padding factor for the file is used.

DATAVOLUME: Data Storage Extent Volume

Note: The value for the DATAVOLUME parameter must be enclosed in apostrophes.

DATAVOLUME specifies the volume on which the file's Data Storage space (DS extents) are allocated. If the number of blocks requested with DSSIZE cannot be found on the specified volume, ADAORD allocates the remaining blocks on other volumes according to its default allocation rules.

If DSRABN is specified, DATAVOLUME is ignored for the related file.

If DATAVOLUME is not specified, the Data Storage space is allocated according to ADAORD's default allocation rules.

DSDEV: Data Storage Device Type

DSDEV specifies the device type to be used for the file's Data Storage. The specified device type must already be defined to Adabas, normally when the database was created or by the ADADBS utility's ADD function.

If this parameter is not specified, ADAORD attempts to allocate the file on the device type used before reordering.

DSRABN: Data Storage Starting RABN

DSRABN specifies the beginning RABN for the file's Data Storage extent. If the DSRABN parameter is omitted, ADAORD assigns the starting RABN.

DSRELEASE: Release Unused Data Storage Blocks

Specifying DSRELEASE releases unused Data Storage (DS) blocks belonging to the file. If DSRELEASE is not specified, ADAORD allocates *at least* the number of DS blocks that were allocated before the file was reordered.



Note: Adabas calculates the file extent size using any changed padding factor or block size values *before* the file is reordered.

DSSIZE: Data Storage Size

DSSIZE specifies the number of blocks or cylinders to be allocated for the Data Storage. A block count must be followed by a "B" (for example, "2000B").

If this parameter is omitted, ADAORD calculates the file extent size in proportion to any increase or decrease in the DATAPFAC padding factor.

EXCLUDE: Exclude Specified Files from Reorder

EXCLUDE lists the numbers of the files to be excluded from REORDER processing; that is, the files that are not to be reordered.

The parameter is optional: if not specified, no files are excluded. A file number may be listed only once.

Files specified in the EXCLUDE parameter must also be specified in the FILE parameter.

The EXCLUDE parameter is provided for use in recovery jobs built by the Adabas Recovery Aid (ADARAI).

INDEXCOMPRESSION: Compress File Index

INDEXCOMPRESSION indicates whether the index for the file is rebuilt in compressed or uncompressed form. A compressed index usually requires less index space and improves the efficiency of index operations in the Adabas nucleus.

If INDEXCOMPRESSION is not specified, the default is the current form of the file.

ISNSIZE: 3- or 4-Byte ISN

ISNSIZE specifies whether ISNs in the file are to be 3 or 4 bytes long. The default is the value currently used for the file; this value is stored in the file control block (FCB).



Note: It is not possible to change the ISNSIZE of a physically coupled file using ADAORD.

LIP: ISN Buffer Pool Size

The LIP parameter can be used to decrease the number of Associator I/O operations when re-creating the address converter. For best performance, specify a size that accepts all ISNs of the largest file to be processed.

LIP specifies the size of the ISN pool for containing ISNs and their assigned Data Storage RABNs. The value may be specified in bytes as a numeric value ("2048") or in kilobytes as a value followed by a "K" ("2K"). The default for LIP is 16384 bytes (or 16K).

The length of one input record is ISNSIZE + RABNSIZE. Thus the entry length is at least 6 bytes (the ISNSIZE of the file is 3 and the RABNSIZE of the database is 3) and at most 8 bytes (the ISNSIZE is 4 and the RABNSIZE is 4).



Note: When ADAORD is processing files that contain spanned records with secondary ISNs, a second LIP will be allocated to contain these ISNs.

LPB: Prefetch Buffer Size

LPB specifies the size, in bytes, of the internal prefetch buffer. The maximum value is 32760 bytes. The default depends on the ADARUN LU parameter. ADAORD may reduce a specified LPB value if the LU value is too small.

MAXISN: Highest ISN Permitted for the File

MAXISN is the highest ISN which may be allocated for the file. This value must be greater than the current TOPISN value displayed in the ADAREP database report.

ADAORD uses the specified value to calculate the address converter space required. If this parameter is omitted, the current MAXISN value for the file is retained.

MAXISN2: Highest Secondary ISN Permitted for the File

MAXISN specifies the desired size of the secondary address converter (AC2) in ISNs. This value must be greater than the current TOP AC2 ISN value displayed in the ADAREP database report. The secondary address converter is used to map the secondary ISNs of secondary spanned records to the RABNs of the Data Storage blocks where the secondary records are stored.

ADAORD uses the specified value to calculate the space required in the secondary address converter for the file. If this parameter is omitted, the current MAXISN2 value for the file is retained. If the file contains no secondary address converter extents, this parameter is ignored.

MAXRECL: Maximum Compressed Record Length

Use the MAXRECL parameter to change the maximum record length, after compression, permitted in the file. Specifying MAXRECL has two effects:

- The file's DATA data set is allocated only to devices that support the specified length.
- If the file contains Data Storage records that exceed the specified length, ADAORD abends and prints the ERROR-126 message (Data Storage record too long).



Note: MAXRECL is ignored if specified for files enabled for spanned data storage records.

If MAXRECL is not specified, the maximum compressed record length does not change.

NIRABN: Starting RABN for Normal Index

NIRABN is the beginning RABN for the normal index extent. If this parameter is omitted, ADAORD assigns the starting RABN.

NIRELEASE: Release Unused Normal Index Blocks

Specifying NIRELEASE releases unused normal index (NI) blocks belonging to the file. If NIRELEASE is not specified, ADAORD allocates *at least* the number of NI blocks that were allocated before the file was reordered.



Note: Adabas calculates the file extent size using any changed padding factor or block size values *before* the file is reordered.

NISIZE: Normal Index Size

NISIZE specifies the number of blocks or cylinders to be allocated for the file's normal index. A block count must be followed by a "B" (for example, "2000B").

If this parameter is omitted, ADAORD computes the file extent size in proportion to any increase or decrease in the ASSOPFAC padding factor.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PASSWORD: File Password

If the file is password-protected, use this parameter to specify the password.

SORTSEQ: Reorder Sequence

SORTSEQ determines the sequence in which the file is to be processed. If this parameter is omitted, the records are processed in physical sequence.

If a descriptor is specified, the file is processed in the logical sequence of the descriptor values. *Do not* use a null-suppressed descriptor field, a hyperdescriptor, a phonetic descriptor, a multiple-value field, or a descriptor contained in a periodic group.



Note: Even when the descriptor field is not null suppressed, the record is *not* represented in the inverted list if the descriptor field or a field following it has never been initialized (held a value). Therefore, the record will be dropped when the utility is executed.

If ISN is specified, the file is processed in ascending ISN sequence. For the Adabas checkpoint or security file, only SORTSEQ=ISN is allowed.

TEST: Test Syntax

This parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

UIRABN: Starting RABN for Upper Index

UIRABN is the beginning RABN for the file's upper index extent. If this parameter is omitted, ADAORD assigns the starting RABN.

UIRELEASE: Release Unused Upper Index Blocks

Specifying UIRELEASE releases unused upper index (UI) blocks belonging to the file. If UIRELEASE is not specified, ADAORD allocates *at least* the number of UI blocks that were allocated before the file was reordered.

Note: Adabas calculates the file extent size using any changed padding factor or block size values *before* the file is reordered.

UISIZE: Index Size for Upper Index

UISIZE specifies the number of blocks or cylinders to be allocated for the upper index. A block count must be followed by a "B" (for example, "2000B").

If this parameter is omitted, ADAORD computes the file extent size in proportion to any increase or decrease in the ASSOPFAC padding factor.

Examples

Example 1:

```
ADAORD   REORFILE FILE=16
```

Associator and Data Storage for file 16 are to be reordered. All current extent sizes and padding factors for the file are to be retained. No other files are reordered.

Example 2:

```
ADAORD   REORFILE    FILE=246
ADAORD              DATAPFAC=5,DSSIZE=10B,SORTSEQ=MZ
ADAORD              ASSOPFAC=20,MAXISN=5000
ADAORD              FILE=20
```

File 246 is to be reordered; the Associator and Data Storage for all other files remain unchanged. The Data Storage padding factor is to be set to 5. A new Data Storage size of 10 blocks is to be used. The Data Storage is to be reordered in the logical sequence of descriptor MZ. The new Associator padding factor is 20. The highest ISN which may be assigned is 5000. File 20 is to be reordered with no changes to its current parameters.

Example 3:

```
ADAORD   REORFILE    FILE=9
ADAORD              ASSOPFAC=5,DATAPFAC=15
ADAORD              FILE=23
ADAORD              DSRABN=24032
```

```
ADAORD              UIRABN=3151,UISIZE=50B
ADAORD              NIRABN=3201
```

File 9 is to be reordered. The Associator padding factor is set to 5% and the Data Storage padding factor to 15%.

File 23 will also be reordered, with a new starting Data Storage RABN of 24032. In addition, the following Associator changes are being made for file 23: the new upper index starting RABN is 3151, with a new upper index size of 50 blocks. The new normal index starting RABN is 3201; the size remains the same.

130 RESTRUCTUREDB: Restructure Database

- Optional Parameters and Subparameters 929
- Examples 982

The RESTRUCTUREDB function unloads an entire database to a sequential data set, which can be used as input to the STORE function to load the data into a new database. The target database may be located on a physical device type different from the source database. The Associator and Data Storage are reordered as part of RESTRUCTURE/STORE processing.

This function requires exclusive EXF control of the database files involved in the operation. In addition, parts of the database are overwritten during ADAORD execution, so we recommend that you back up the database (or file) using the ADASAV utility first, before running ADAORD functions.

If the file specified for this function was originally loaded with ISNREUSE=YES active, this reorder function will reset the first unused ISN value in that file's control block (FCB) to the actual first unused ISN found in the address converter.

When the RESTRUCTUREDB function restructures an ADAM file that uses the overflow area, and then the STORE function stores the restructured file in a database with a smaller DATA block size, an ADAORD ERROR-103 may occur. Use the ADAULD and ADALOD utilities to move ADAM files, instead.



Note: You can restructure databases and files from an Adabas version prior to Adabas 8 and store them in an Adabas 8 (or later) database using ADAORD STORE. However, you cannot store the restructured output of an Adabas 8 database or file in a database running with any prior Adabas version (for example, Adabas 7). If you attempt this, the following warning will be generated and ADAORD will end with a CC=4:

```
*** Warning: The input data set is from V8 and will not be processed
```

This is the syntax of the ADAORD RESTRUCTUREDB function:

```
ADAORD  { RESTRUCTUREDB | REDB }
                [DBASSODEV = { device-type ]
                [DBDATADEV = { device-type ]
                [DBINDEXCOMPRESSION = { YES | NO } ]
                [FILE = file-number ]
                    [ASSOPFAC = padding-factor ]
                    [ASSODEV = device-type ] [DATADEV = device-type ]
                    [DATAPFAC = padding-factor ]
                    [INDEXCOMPRESSION = { YES | NO } ]
                    [ISNSIZE = { 3 | 4 } ]
                    [SORTSEQ = { descriptor | ISN } ]
                [LPB = prefetch-buffer-size ]
                [NOUSERABEND]
                [TEST]
```

Optional Parameters and Subparameters

ASSODEV: Associator Device Type

ASSODEV specifies the device type to be used in the new database for the file's ASSO data set. This parameter is required only when the device type to be used is different from the default device type. The default device type is specified by the DBASSODEV parameter; if DBASSODEV is not specified, the default is the device type specified by the ADARUN DEVICE parameter. These parameters have no effect on the data written to the DDFILEA data set.

ASSOPFAC: Associator Padding Factor

ADAORD uses the ASSOPFAC value to calculate the space required to perform the STORE function for the specified file. Valid values are 1-90. The number of AC, NI, and UI blocks is calculated for the device type specified by ASSODEV and the padding factor specified by ASSOPFAC. These parameters have no effect on the data written to DDFILEA. If ASSOPFAC is not specified, the current padding factor for the file is used.

DATADEV: Data Storage Device Type

DATADEV specifies the device type to be used for the specified file's new DATA data set. This parameter is required only when the device type to be used is different from the default device type. The default device type is specified by the DBDATADEV parameter; if DBDATADEV is not specified, the default is the device type specified by the ADARUN DEVICE parameter. These parameters have no effect on the data written to DDFILEA.

DATAPFAC: Data Storage Padding Factor

ADAORD uses DATAPFAC to calculate the space required to perform the STORE function for the specified file. Valid values are 1-90 (see the ADALOD LOAD DATAPFAC parameter discussion for more information about setting the padding factor). The number of Data Storage blocks is calculated for the device type specified by DATADEV and the padding factor specified by DATAPFAC. If DATAPFAC is not specified, the current padding factor for the file is used. These parameters have no effect on the data written to DDFILEA.

DBASSODEV: Default Associator Device Type

DBASSODEV specifies a default device type for the new ASSO data set. ADAORD uses the device type specified here to calculate the ASSO space requirements for each restructured file. If DBASSODEV is not specified, the default is the device type specified by the ADARUN DEVICE parameter.

To override the default device type for a file, use the FILE and ASSODEV parameters. The DBASSODEV parameter has no effect on the data written to the DDFILEA data set.

DBDATADEV: Default Data Storage Device Type

DBDATADEV specifies a default device type for the new DATA data set. ADAORD uses the device type specified here to calculate the DATA space requirements for each restructured file. If DBDATADEV is not specified, the default is the device type specified by the ADARUN DEVICE parameter.

To override the default device type for a file, use the FILE and DATADEV parameters. The DATADEV parameter has no effect on the data written to the DDFILEA data set.

DBINDEXCOMPRESSION: Calculate Index Sizes for Database

DBINDEXCOMPRESSION indicates for all files whether the index space calculation performed and displayed by ADAORD is based on compressed or uncompressed indexes. It applies to all files for which no INDEXCOMPRESSION parameter is specified.

DBINDEXCOMPRESSION can be used to calculate the sizes of compressed or uncompressed indexes for all files of the database, making it unnecessary to calculate the sizes for each file.

FILE: File Number

FILE specifies the file to which the following parameters apply. The records for all files not specified by this parameter are unloaded in physical sequence, by file.

If an Adabas checkpoint or security file is specified, do not specify the SORTSEQ parameter.

INDEXCOMPRESSION: Calculate Index Sizes for File

INDEXCOMPRESSION indicates for its associated file whether the index space calculation performed and displayed by ADAORD is based on a compressed or uncompressed index.

If INDEXCOMPRESSION is *not* specified

- but the DBINDEXCOMPRESSION parameter is specified for the database as a whole, the default is the database value.

- and DBINDEXCOMPRESSION is also *not* specified, the default is the current compression form of the file.

ISNSIZE: 3- or 4-Byte ISN

ISNSIZE specifies whether ISNs in the file are to be 3 or 4 bytes long. The default is the value currently used for the file; this value is stored in the file control block (FCB).



Note: It is not possible to change the ISNSIZE of a physically coupled file using ADAORD.

LPB: Prefetch Buffer Size

LPB specifies the size, in bytes, of the internal prefetch buffer. The maximum value is 32760 bytes. The default depends on the ADARUN LU parameter. ADAORD may reduce a specified LPB value if the LU value is too small.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

SORTSEQ: File Processing Sequence

SORTSEQ determines the sequence in which the file is to be processed. If this parameter is omitted, the records are processed in physical sequence.

If a descriptor is specified, the file is processed in the logical sequence of the descriptor values. *Do not* use a null-suppressed descriptor field, a hyperdescriptor, a phonetic descriptor, a multiple-value field, or a descriptor contained in a periodic group.



Note: Even when the descriptor field is not null suppressed, the record is *not* represented in the inverted list if the descriptor field or a field following it has never been initialized (held a value). Therefore, the record will be dropped when the utility is executed.

If ISN is specified, the file is processed in ascending ISN sequence. For the Adabas checkpoint or security file, only SORTSEQ=ISN is allowed.

TEST: Test Syntax

This parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

Examples

Example 1:

```
ADAORD RESTRUCTUREDB
```

The RESTRUCTUREDB function is to be executed. All files are to be unloaded in physical sequence.

Example 2:

```
ADAORD RESTRUCTUREDB FILE=146,SORTSEQ=MZ
ADAORD               FILE=151,SORTSEQ=TF
```

The RESTRUCTUREDB function is to be executed. File 146 is to be unloaded in the sequence of descriptor MZ. File 151 is to be unloaded in the sequence of descriptor TF. All other files are to be unloaded in physical sequence.

131 RESTRUCTUREF: Restructure Single Files

- Essential Parameter 935
- Optional Parameters 935
- Examples 938

The RESTRUCTUREF function unloads one or more files to a sequential data set, which can be used as input to the STORE function to load the files into another database. The target database may be located on a physical device type different from the originating database. The Associator and Data Storage are reordered as part of RESTRUCTURE/STORE processing.

RESTRUCTUREF requires EXU control; other users may access database files being used by RESTRUCTUREF, but only for reading. Note, however, that operations involving either the checkpoint or security files require exclusive database control. In addition, parts of the database are overwritten during ADAORD execution. We therefore recommend that you back up the database (or file) using the ADASAV utility before running ADAORD functions.

If the file specified for this function was originally loaded with ISNREUSE=YES active, this reorder function will reset the first unused ISN value in that file's control block (FCB) to the actual first unused ISN found in the address converter.

When the RESTRUCTUREF function restructures an ADAM file that uses the overflow area, and then the STORE function stores the restructured file in a database with a smaller DATA block size, an ADAORD ERROR-103 may occur. Use the ADAULD and ADALOD utilities to move ADAM files, instead.



Note: You can restructure databases and files from an Adabas version prior to Adabas 8 and store them in an Adabas 8 (or later) database using ADAORD STORE. However, you cannot store the restructured output of an Adabas 8 database or file in a database running with any prior Adabas version (for example, Adabas 7). If you attempt this, the following warning will be generated and ADAORD will end with a CC=4:

```
*** Warning: The input data set is from V8 and will not be processed
```

This is the syntax of the ADAORD RESTRUCTUREF function:

```
ADAORD  { RESTRUCTUREF | REF }
                FILE = file-number
                    [ASSODEV = device-type ]
                    [ASSOPFAC = padding-factor ]
                    [DATADEV = device-type ]
                    [DATAPFAC = padding-factor ]
                    [DBASSODEV = device-type ]
                    [DBDATADEV = device-type ]
                    [INDEXCOMPRESSION = { YES | NO } ]
                    [ISNSIZE = { 3 | 4 } ]
                    [PASSWORD = password ]
                    [SORTSEQ = { descriptor | ISN } ]
                [LPB = prefetch-buffer-size } ]
                [NOUSERABEND]
                [TEST]
```

Essential Parameter

FILE: File Number

FILE specifies the file to be restructured. A separate ADAORD FILE statement must be provided for each file to be processed, followed by ADAORD statements containing the relevant parameters for that file.

If you specify a file that is either coupled or part of an expanded file, the related files are automatically added to the file list. A message indicating the files added appears in DDPRINT.

Optional Parameters

ASSODEV: Associator Device Type

ASSODEV specifies the device type to be used for the specified file's new ASSO data set. This parameter is required only when the device type to be used is different from the default device type. The default device type is specified by the DBASSODEV parameter; if DBASSODEV is not specified, the default is the device type specified by the ADARUN DEVICE parameter. These parameters have no effect on the data written to the DDFILEA data set.

ASSOPFAC: Associator Padding Factor

ADAORD uses ASSOPFAC to calculate the space required to perform the STORE function for the specified file. Valid values are 1-90. The number of AC, NI, and UI blocks is calculated for

the device type specified by ASSODEV and the padding factor specified by ASSOPFAC. If ASSOPFAC is not specified, the current padding factor for the file is used. These parameters have no effect on the data written to DDFILEA.

DATADEV: Data Storage Device Type

DATADEV specifies the device type to be used for the specified file's new DATA data set. This parameter is required only when the device type to be used is different from the default device type. The default device type is specified by the DBDATADEV parameter; if DBDATADEV is not specified, the default is the device type specified by the ADARUN DEVICE parameter. These parameters have no effect on the data written to DDFILEA.

DATAPFAC: Data Storage Padding Factor

ADAORD uses DATAPFAC to calculate the space required to perform the STORE function for the specified file. Valid values are 1-90 (see the ADALOD LOAD DATAPFAC parameter discussion for more information about setting the padding factor). The number of Data Storage blocks is calculated for the device type specified by DATADEV and the padding factor specified by DATAPFAC. If DATAPFAC is not specified, the current padding factor for the file is used. These parameters have no effect on the data written to DDFILEA.

DBASSODEV: Default Associator Device Type

DBASSODEV specifies a default device type for the new ASSO data set. ADAORD uses the device type specified here to calculate the ASSO space requirements for each restructured file. If DBASSODEV is not specified, the default is the device type specified by the ADARUN DEVICE parameter.

To override the default device type for a file, use the FILE and ASSODEV parameters. The DBASSODEV parameter has no effect on the data written to the DDFILEA data set.

DBDATADEV: Default Data Storage Device Type

DBDATADEV specifies a default device type for the new DATA data set. ADAORD uses the device type specified here to calculate the DATA space requirements for each restructured file. If DBDATADEV is not specified, the default is the device type specified by the ADARUN DEVICE parameter.

To override the default device type for a file, use the FILE and DATADEV parameters. The DBDATADEV parameter has no effect on the data written to the DDFILEA data set.

INDEXCOMPRESSION: Calculate Index Sizes for File

INDEXCOMPRESSION indicates for its associated file whether the index space calculation performed and displayed by ADAORD is based on a compressed or uncompressed index.

If INDEXCOMPRESSION is *not* specified

- but a compression value is specified for the database as a whole, the default is the database value.
- and *no* compression value is specified for the database, the default is the current compression form of the file.

ISNSIZE: 3- or 4-Byte ISN

ISNSIZE specifies whether ISNs in the file are to be 3 or 4 bytes long. The default is the value currently used for the file; this value is stored in the file control block (FCB).



Note: It is not possible to change the ISNSIZE of a physically coupled file using ADAORD.

LPB: Prefetch Buffer Size

LPB specifies the size, in bytes, of the internal prefetch buffer. The maximum size is 32760 bytes. The default depends on the ADARUN LU parameter. ADAORD may reduce a specified LPB value if the LU value is too small.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PASSWORD: File Password

If the file is password-protected, use this parameter to specify the password.

SORTSEQ: File Processing Sequence

SORTSEQ determines the sequence in which the file is to be processed. If this parameter is omitted, the records are processed in physical sequence.

If a descriptor is specified, the file is processed in the logical sequence of the descriptor values. *Do not* use a null-suppressed descriptor field, a hyperdescriptor, a phonetic descriptor, a multiple-value field, or a descriptor contained in a periodic group.



Note: Even when the descriptor field is not null suppressed, the record is *not* represented in the inverted list if the descriptor field or a field following it has never been initialized (held a value). Therefore, the record will be dropped when the utility is executed.

If ISN is specified, the file is processed in ascending ISN sequence. For the Adabas checkpoint or security file, only SORTSEQ=ISN is allowed.

TEST: Test Syntax

This parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not that the validity of values and variables.

Examples

Example 1:

```
ADAORD RESTRUCTUREF FILE=15                              ↩
```

The RESTRUCTUREF function is to be executed. File 15 is to be unloaded in physical sequence. No other files are to be unloaded.

Example 2:

```
ADAORD RESTRUCTUREF  FILE=25,SORTSEQ=KL
ADAORD               FILE=40,SORTSEQ=J3
```

The RESTRUCTUREF function is to be executed. Files 25 and 40 are to be unloaded. File 25 is to be unloaded in the sequence of descriptor KL. File 40 is to be unloaded in the sequence of descriptor J3. No other files are to be unloaded.

132 STORE: Store Files

- Optional Parameters and Subparameters 942
- Examples 949

The STORE function loads one or more files into an existing database using output produced by the RESTRUCTURE functions. The Associator and Data Storage are reordered as part of RESTRUCTURE/STORE processing.

If the ALLFILES parameter is specified, all files contained on the input data set are stored. If ALLFILES is not specified, only those files specified by FILE parameters are stored.

One or more files may be specified with FILE parameter statements, even when ALLFILES is also specified. The STORE function loads each file specified with a FILE statement according to the definition contained in any subparameters immediately following that file's FILE statement. All other files are loaded according to their existing definitions.

If existing files in the database are to be overwritten, the OVERWRITE parameter must be supplied.

This function requires exclusive EXF control of the database files involved in the operation. In addition, parts of the database are overwritten during ADAORD execution, so we recommend that you back up the database (or file) using the ADASAV utility first, before running ADAORD functions.

If the file specified for this function was originally loaded with ISNREUSE=YES active, this store function will reset the first unused ISN value in that file's control block (FCB) to the actual first unused ISN found in the address converter.



Notes:

1. The STORE function does not reorder ADAM files. However, it, in conjunction with other ADAORD functions, can be used to relocate an ADAM file to different RABNs.
2. Storing restructured ADAM files on a device with a smaller DATA blocking factor than before can result in utility ERROR 103 if the ADAM file previously used the overflow area. To relocate an ADAM file to a different device, use the ADAULD and ADALOD utilities.
3. Checkpoint and security files from Adabas version 5.1 or 5.2 cannot be stored due to internal structure changes to the files in version 5.3.
4. If ALLFILES is requested in the STORE function, be sure that the Associator (ASSO) data set is large enough to accommodate the all the files. Use the calculation `(file-count * 3) + 110` to verify this. Otherwise, an ERROR-068 is possible.
5. You can restructure databases and files from an Adabas version prior to Adabas 8 and store them in an Adabas 8 (or later) database using ADAORD STORE. However, you cannot store the restructured output of an Adabas 8 database or file in a database running with any prior Adabas version (for example, Adabas 7). If you attempt this, the following warning will be generated and ADAORD will end with a CC=4: `*** Warning: The input data set is from V8 and will not be processed`
6. Logically deleted field data in the file is loaded by the ADAORD STORE utility function.

This is the syntax of the ADAORD STORE function:

Optional Parameters and Subparameters

ACRABN: Starting RABN for Address Converter

ACRABN specifies the beginning RABN for the file's address converter extent. If this parameter is omitted, ADAORD assigns the starting RABN. The space requested must be available in one extent.

When specifying the starting RABN for Associator extents, the space needed for the FCBs, FDTs, and DSST should also be considered.

AC2RABN: Starting RABN for Secondary Address Converter

The beginning RABN for the file's secondary address converter extent. The secondary address converter is used to map the secondary ISNs of secondary spanned records to the RABNs of the Data Storage blocks where the secondary records are stored.

If this parameter is omitted, ADAORD assigns the starting RABN. The space requested must be available in one extent. If the file contains no secondary address converter extents, this parameter is ignored.

ALLFILES: Select All Files for Storing

ALLFILES causes all files in the input data set to be stored in the database. If ALLFILES is not supplied, only those files are stored for which FILE parameters have been specified.

If the input data set contains files that are coupled or part of an expanded file and the related files are not in the data set, ERROR-138 is returned indicating an inconsistent file list. You must add the related files before the STORE function will execute successfully.



Note: If ALLFILES is requested in the STORE function, be sure that the Associator (ASSO) data set is large enough to accommodate the all the files. Use the calculation `(file-count * 3) + 110` to verify this. Otherwise, an ERROR-068 is possible.

ALLOCATION: Action to Follow File Extent Allocation Failure

ALLOCATION specifies the action to be taken if file extent allocations cannot be obtained according to the placement parameters ACRABN, DSRABN, NIRABN, or UIRABN.

By default (that is, ALLOCATION=FORCE), the utility terminates with error if any file extent allocation cannot be met according to RABN placement parameters.

If ALLOCATION=NOFORCE is specified and any allocation with placement parameters fails, the utility retries the allocation without the placement parameter.

ASSOPFAC: Associator Padding Factor

ASSOPFAC specifies the new Associator block padding factor. The number specified represents the percentage of each Associator block not to be used during the reorder process. A value in the range 1-90 may be specified. The remaining number of bytes after padding must be greater than the largest descriptor value plus 10.

If this parameter is omitted, the current Associator padding factor in effect for the file is used.

ASSOVOLUME: Associator Extent Volume

Note: The value for the ASSOVOLUME parameter must be enclosed in apostrophes.

ASSOVOLUME identifies the volume on which to allocate the file's Associator space (the AC, NI, and UI extents). If the requested number of blocks cannot be found on the specified volume, ADAORD allocates the remaining blocks on other volumes according to its default allocation rules.

If ACRABN, UIRABN, or NIRABN is specified, ADAORD ignores the ASSOVOLUME value when allocating the corresponding extent type.

If ASSOVOLUME is not specified, the file's Associator space is allocated according to ADAORD's default allocation rules.

CHECKPOINT: Store the Checkpoint File

If either ALLFILES is specified or the FILE parameter specifies the checkpoint file, CHECKPOINT stores the checkpoint file from the DDFILEA tape in the database, making that file the new checkpoint file. The new checkpoint file must have the same file number as the old checkpoint file.

If the CHECKPOINT parameter is not specified, the checkpoint file on the FILEA tape is not stored in the database, even though the checkpoint file was specified by a FILE parameter or the ALLFILES parameter was specified.

DATAPFAC: Data Storage Padding Factor

DATAPFAC specifies the new Data Storage padding factor, which is the percentage of each Data Storage block reserved for record expansion when the file is reordered. A value in the range 1-90 may be specified (see the ADALOD LOAD DATAPFAC parameter discussion for more information about setting the padding factor). If this parameter is omitted, the current padding factor for the file is used.

DATAVOLUME: Data Storage Extent Volume

Note: The value of the DATAVOLUME parameter must be enclosed in apostrophes.

DATAVOLUME specifies the volume on which the file's Data Storage space (DS extents) are allocated. If the number of blocks requested with DSSIZE cannot be found on the specified volume, ADAORD allocates the remaining blocks on other volumes according to its default allocation rules.

If DSRABN is specified, DATAVOLUME is ignored for the related file.

If DATAVOLUME is not specified, the Data Storage space is allocated according to ADAORD's default allocation rules.

DSDEV: Data Storage Device Type

DSDEV specifies the device type to be used for the file's Data Storage. The specified device type must already be defined to Adabas, normally when the database was created or by the ADADBS utility's ADD function.

If this parameter is not specified, ADAORD attempts to allocate the file on the device type used before restructuring.

DSRABN: Data Storage Starting RABN

The beginning RABN for the Data Storage extent for the specified file. If this parameter is omitted, ADAORD assigns the starting RABN.

DSRELEASE: Release Unused Data Storage Blocks

Specifying DSRELEASE releases unused Data Storage (DS) blocks belonging to the specified file. If DSRELEASE is not specified, ADAORD allocates *at least* the number of DS blocks that were allocated before the file was reordered.



Note: Adabas calculates the file extent size using any changed padding factor or block size values *before* the file is reordered.

DSSIZE: Data Storage Size

DSSIZE specifies the number of blocks or cylinders to be allocated for the Data Storage. A block count must be followed by a "B" (for example, "2000B").

If this parameter is omitted, ADAORD will compute the file extent size (in blocks) in proportion to an increase or decrease in the DATAPFAC padding factor used.

EXCLUDE: Exclude Specified Files from Store

EXCLUDE lists the numbers of the files to be excluded from STORE processing; that is, the files that are not to be stored.

The parameter is optional: if not specified, no files are excluded. A file number may be listed only once.

The EXCLUDE parameter is provided for use in recovery jobs built by the Adabas Recovery Aid (ADARAI).

FILE: File Number

FILE specifies the file to be stored. A separate statement must be provided for each file to be processed, followed by ADAORD statements containing the relevant parameters for that file.

If you specify a file that is coupled or part of an expanded file and you do not also specify the related files, ERROR-138 is returned indicating an inconsistent file list. You must add the related files before the STORE function will execute successfully.

INDEXCOMPRESSION: Compress File Index

INDEXCOMPRESSION indicates whether the index for the file is rebuilt in compressed or uncompressed form. A compressed index usually requires less index space and improves the efficiency of index operations in the Adabas nucleus.

If INDEXCOMPRESSION is not specified, the default is the form of the file at the time of the corresponding restructure operation.

ISNSIZE: 3- or 4-Byte ISN

ISNSIZE specifies whether ISNs in the file are to be 3 or 4 bytes long. The default is the value currently used for the file; this value is stored in the file control block (FCB).



Note: It is not possible to change the ISNSIZE of a physically coupled file using ADAORD.

LIP: ISN Buffer Pool Size

The LIP parameter can be used to decrease the number of Associator I/O operations when re-creating the address converter. For best performance, specify a size that accepts all ISNs of the largest file to be processed.

LIP specifies the size of the ISN pool for containing ISNs and their assigned Data Storage RABNs. The value may be specified in bytes as a numeric value ("2048") or in kilobytes as a value followed by a "K" ("2K"). The default for LIP is 16384 bytes (or 16K).

The length of one input record is ISNSIZE + RABNSIZE. Thus the entry length is at least 6 bytes (the ISNSIZE of the file is 3 and the RABNSIZE of the database is 3) and at most 8 bytes (the ISNSIZE is 4 and the RABNSIZE is 4).



Note: When ADAORD is processing files that contain spanned records with secondary ISNs, a second LIP will be allocated to contain these ISNs.

MAXISN: Highest ISN Permitted for the File

MAXISN specifies the highest ISN which may be allocated for the file. This value must be greater than the current TOPISN value displayed in the ADAREP database report.

ADAORD uses the specified value to calculate the address converter space required. If this parameter is omitted, the current MAXISN value for the file is retained.

MAXISN2: Highest Secondary ISN Permitted for the File

MAXISN specifies the desired size of the secondary address converter (AC2) in ISNs. This value must be greater than the current TOP AC2 ISN value displayed in the ADAREP database report. The secondary address converter is used to map the secondary ISNs of secondary spanned records to the RABNs of the Data Storage blocks where the secondary records are stored.

ADAORD uses the specified value to calculate the space required in the secondary address converter for the file. If this parameter is omitted, the current MAXISN2 value for the file is retained. If the file contains no secondary address converter extents, this parameter is ignored.

MAXRECL: Maximum Compressed Record Length

Use the MAXRECL parameter to change the maximum record length, after compression, permitted in the file. Specifying MAXRECL has two effects:

- The file's DATA data set is allocated only to devices that support the specified length.

- If the file contains Data Storage records that exceed the specified length, ADAORD abends and prints the ERROR-126 message (Data Storage record too long).



Note: MAXRECL is ignored if specified for files enabled for spanned data storage records.

If MAXRECL is not specified, there are two possibilities for the default value:

- If the maximum compressed record length before the file was restructured was the default ADALOD MAXRECL value, then DATA is allocated to a device arbitrarily, and the new maximum record length is derived from the device type;
- Otherwise, the maximum compressed record length does not change.

NIRABN: Starting RABN for Normal Index

The beginning RABN for the file's normal index extent. If this parameter is omitted, ADAORD assigns the starting RABN.

NIRELEASE: Release Unused Normal Index Blocks

Specifying NIRELEASE releases unused normal index (NI) blocks belonging to the file. If NIRELEASE is not specified, ADAORD allocates *at least* the number of NI blocks that were allocated before the file was reordered.



Note: Adabas calculates the file extent size using any changed padding factor or block size values *before* the file is reordered.

NISIZE: Normal Index Size

The number of blocks or cylinders to be allocated for the normal index. The specified value cannot be larger than the largest single contiguous RABN area available; specifying blocks (a number of blocks followed by a "B") is therefore recommended.

If this parameter is omitted, ADAORD computes the file extent size (in blocks) in proportion to an increase or decrease in the ASSOPFAC padding factor used.

If this parameter is omitted and the INDEXCOMPRESSION parameter is specified, the ADAORD index size calculation does not consider the change in index size because ADAORD has no knowledge of the compression rate to be expected. Thus, ADAORD may allocate an index smaller than required causing secondary index extent allocations; or larger than necessary.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

OVERWRITE: Overwrite Existing File

If a file to be stored already exists in the database, ADAORD terminates with an error message unless the OVERWRITE parameter has been specified.

READONLY: Read-only Status Indicator

READONLY indicates whether the read-only status is on or off. Valid values for this parameter are "YES" (read-only status is on) and "NO" (read-only status is off).

If READONLY is not specified, the default status of the file at the time of the corresponding restructure operation is used.

REPLICATOR: Store the Replicator System File

Use this parameter to store the Replicator system file from the DDFILEA/FILEA tape as the new Replicator system file for the Event Replicator Server. The new Replicator system file must have the same file number as the old Replicator system file. This parameter can only be specified if the database is an Event Replicator Server.

When the `REPLICATOR` parameter is not specified, the Replicator system file on the DDFILEA tape is not stored in the database, even if it is specified by a `FILE` or `ALLFILES` parameter.

RPLDSBI: Before Image for Data Storage

The `RPLDSBI` parameter indicates whether the collection of before images of data storage during an update command to the file should be done if replication is turned on for the file.

Parameter `RPLDSBI` may only be specified if replication is turned on for the file.

If RPLDSBI is not specified, the default value depends on the target database ID (DBID) of the RESTORE and its replication state; if the ADAORD STORE DBID is the same as the original RESTRUCTURE DBID and REPLICATION=YES in the target DBID, the original RPLDSBI value is used. Otherwise, RPLDSBI is set to NO.

RPLKEY: Primary Key for Replication

The `RPLKEY` parameter specifies the primary key for replication. The format is RPLKEY='*primary-key-for-replication*'. When a valid descriptor name is specified for RPLKEY, the specified descriptor is used as the primary replication key for the corresponding file. When OFF is specified for RPLKEY, no primary key is used for replication (RPLKEY is turned off) for the corresponding file.

This parameter may only be specified if replication is turned on for the file and if the provided descriptor name is a valid Adabas descriptor.

If RPLKEY is not specified, the default value depends on the target database ID (DBID) of the RESTORE and its replication state; if the ADAORD STORE DBID is the same as the original RESTRUCTURE DBID and REPLICATION=YES in the target DBID, the original RPLKEY value is used. Otherwise, RPLDSBI is set to OFF.

RPLTARGETID: Replication Target ID

The `RPLTARGETID` parameter specifies the Event Replicator target ID used when the Adabas file data is replicated. The format is RPLTARGETID='*reptor-target-id*'. When a valid target

ID is specified for RPLTARGETID, the specified target is used as the target for replicated data for the corresponding file. When OFF or 0 (zero) are specified for RPLTARGETID, no target is used for replication for the corresponding file.

If RPLTARGETID is not specified, the default value depends on the target database ID (DBID) of the RESTORE and its replication state; if the ADAORD STORE DBID is the same as the original RESTRUCTURE DBID and REPLICATION=YES in the target DBID, the original RPLTARGETID value is used. Otherwise, RPLTARGETID is set to OFF.

RPLUPDATEONLY: Allow Only Event Replicator Processing Updates

The `RPLUPDATEONLY` parameter can be used in the ADAORD STORE function to indicate whether an Adabas database file may be updated only by the Event Replicator Server as part of Adabas-to-Adabas replication or by other means as well. This parameter is optional. Valid values are "YES" or "NO". A value of "YES" indicates that the file can only be updated via Event Replicator processing; a value of NO indicates that the file can be updated by any normal means, including Event Replicator processing.

If the file is a new file, the default for this parameter is "NO".

However, if the file specified in the ADAORD STORE function is an existing file, there is no default for this parameter. If no value is specified for the `RPLUPDATEONLY` parameter in the ADAORD STORE function for an existing file, the value used previously for the file is used.

SECURITY: Store the Security File

SECURITY stores the security file from the DDFILEA tape, making that file the new security file for the database. The new file must have the same number as the old security file.

If SECURITY is omitted, the security file on the DDFILEA tape is not stored in the database, even if it is specified by a FILE parameter or the ALLFILES parameter is specified.

SLOG: Store the SLOG System File

Use this parameter to store the SLOG system file from the DDFILEA tape as the new SLOG file for the Event Replicator Server. The new SLOG file must have the same file number as the old SLOG file. This parameter can only be specified if the database is an Event Replicator Server.

When the `SLOG` parameter is not specified, the SLOG system file on the DDFILEA tape is not stored in the database, even if it is specified by a `FILE` or `ALLFILES` parameter.

TEST: Test Syntax

This parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

TRIGGER: Store the Trigger File

Specify the TRIGGER parameter to store the trigger file from the DDFILEA tape as the new trigger file for the database. The new trigger file must have the same file number as the old trigger file.

When the TRIGGER parameter is *not* specified, the trigger file on the DDFILEA tape is *not* stored in the database, even if it is specified by the FILE or ALLFILES parameter.

UIRABN: Starting RABN for Upper Index

UIRABN specifies the beginning RABN for the upper index extent of the file. If this parameter is omitted, ADAORD assigns the starting RABN.

UIRELEASE: Release Unused Upper Index Blocks

Specifying UIRELEASE releases unused upper index (UI) blocks belonging to the file. If UIRELEASE is not specified, ADAORD allocates *at least* the number of UI blocks that were allocated before the file was reordered.

Note: Adabas calculates the file extent size using any changed padding factor or block size values *before* the file is reordered.

UISIZE: Upper Index Size

UISIZE specifies the number of blocks or cylinders to be allocated for the upper index. A block count must be followed by a "B" (for example, "2000B").

If this parameter is omitted, ADAORD computes the file extent size (in blocks) in proportion to an increase or decrease in the ASSOPFAC padding factor used.

If this parameter is omitted and the INDEXCOMPRESSION parameter is specified, the ADAORD index size calculation does not consider the change in index size because ADAORD has no knowledge of the compression rate to be expected. Thus, ADAORD may allocate an index smaller than required causing secondary index extent allocations; or larger than necessary.

Examples

Example 1:

```
ADAORD STORE FILE=14,OVERWRITE                    ↩
```

File 14, as unloaded by one of the RESTRUCTURE or the REORDB functions, is to be stored into an existing database. If the file already exists, it is deleted before being stored.

Example 2:

```
ADAORD   STORE   FILE=1,OVERWRITE
ADAORD           FILE=2,OVERWRITE
ADAORD           FILE=3,OVERWRITE
```

Files 1, 2 and 3 are written to the existing database. Old files 1, 2 and 3 are deleted.

Example 3:

```
ADAORD   STORE   OVERWRITE,ALLFILES
ADAORD           FILE=1,ACRABN=1000,NIRABN=2200
```

```
ADAORD           FILE=2,MAXISN=500000
ADAORD
FILE=4,ASSOPFAC=5,DATAPFAC=20,DSSIZE=5B,DSRABN=1
```

All files unloaded by the RESTRUCTURE function are to be stored into an existing database. The address converter for file 1 is to begin with RABN 1000. The normal index for file 1 is to begin with RABN 2200. The MAXISN for file 2 is to be set to 500000. The following assignments are made for file 4: the Associator block padding factor is set to 5 percent; the Data Storage block padding factor is set to 20 per cent; a new DSSIZE of 5 cylinders is assigned starting at RABN 1.

All other files contained in the input data set are restored with their default values. If a file already exists, it is deleted before the new file is stored.

Example 4:

```
ADAORD STORE ALLFILES,CHECKPOINT
ADAORD EXCLUDE=20,10
```

All files from the input data set (including the checkpoint file) are stored. However, files 10 and 20 are excluded; that is, not stored.

Example 5:

```
ADAORD STORE ALLOCATION=NOFORCE
ADAORD FILE=10
ADAORD DSRABN=12345
```

File 10 is stored in the database. Its data storage is allocated beginning at RABN 12345. If this allocation is not possible because the space is occupied by another file, ADAORD retries the allocation anywhere in the database's data storage.

133 JCL Requirements and Examples

- Requirements to Run ADAORD JCL 952

This section describes the job control information required to run ADAORD on the z/OS operating system and shows examples of job streams.

Note: When running with the optional Recovery Aid (ADARAI) for RESTRUCTUREx or STORE functions, all temporary data sets must also be cataloged in the job control.

Requirements to Run ADAORD JCL

Data Set	DD Name	Storage	More Information
Associator	DDASSORn	disk	
Data Storage	DDDATARn	disk	Not used for REORASSO or REORFASSO
Intermediate storage	DDFILEA	tape/ disk	
Recovery log (RLOG)	DDRLOGR1	disk	Required when using the ADARAI option
ADARUN parameters	DDCARD	reader	*Operations*
ADAORD parameters	DDKARTE	reader	
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADAORD messages	DDDRUCK	printer	*Messages and Codes*

ADAORD JCL Examples

Reorder File Associator

```
//ADAORDA   JOB
//*
//*     ADAORD: REORDER FILE ASSO,
//*               REORDER ASSO
//*
//ORD      EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD       <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDFILEA   DD   DSN=&&DDFILEA,DISP=(,PASS), <===INTERMEDIATE
//          UNIT=SYSDA,VOL=SER=vvvvvv,SPACE=(CYL,NN)          STORAGE
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADAORD,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADAORD REORFASSO
ADAORD FILE=1,ASSOPFAC=15,MAXISN=10000
/*
```

Refer to ADAORDA in the JOBS data set for this example.

Reorder File Data Storage, Reorder File, Reorder Data, Reorder Database

```
//ADAORDD   JOB
//*
//*     ADAORD: REORDER DATA STORAGE
//*
//ORD      EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD       <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDFILEA   DD   DSN=DDFILEA,                        <=== INTERMEDIATE
//           UNIT=TAPE,VOL=SER=vvvvvv,DISP=(,PASS)            FILE
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADAORD,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADAORD REORDATA
ADAORD FILE=1,DSSIZE=80,DATAPFAC=30
/*
```

Refer to ADAORDD in the JOBS data set for this example.

Restructure

```
//ADAORDR   JOB
//*
//*     ADAORD: RESTRUCTURE
//*
//ORD      EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD      <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <== DATA
//DDFILEA   DD   DSN=FILEA,                          <== INTERMEDIATE
//           UNIT=TAPE,VOL=SER=vvvvvv,DISP=(,KEEP)   <== FILE
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADAORD,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADAORD RESTRUCTUREF
```

```
ADAORD FILE=1,DATADEV=eeee
/*
```

Refer to ADAORDR in the JOBS data set for this example.

Store

```
//ADAORDS   JOB
//*
//*     ADAORD: STORE INTO A DIFFERENT DATABASE
//*              AFTER ADAORD RESTRUCTURE
//*
//ORD      EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD      <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDFILEA   DD   DSN=FILE1,                          <=== INTERMEDIATE
//               UNIT=TAPE,VOL=SER=vvvvvv,DISP=OLD             STORAGE
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADAORD,SVC=xxx,DEVICE=dddd,DBID=yyyyy   <=== DBID
/*
//DDKARTE   DD   *
ADAORD STORE
ADAORD FILE=1,DSSIZE=80,DATAPFAC=30,DSRABN=1234,MAXISN=200000
/*
```

Refer to ADAORDS in the JOBS data set for this example.

XIV ADAPLP Utility: Print Data Protection Records from PLOG/Work

This chapter covers the following topics:

Functional Overview
ADAPLP Syntax and Examples
JCL Requirements and Examples

134 Functional Overview

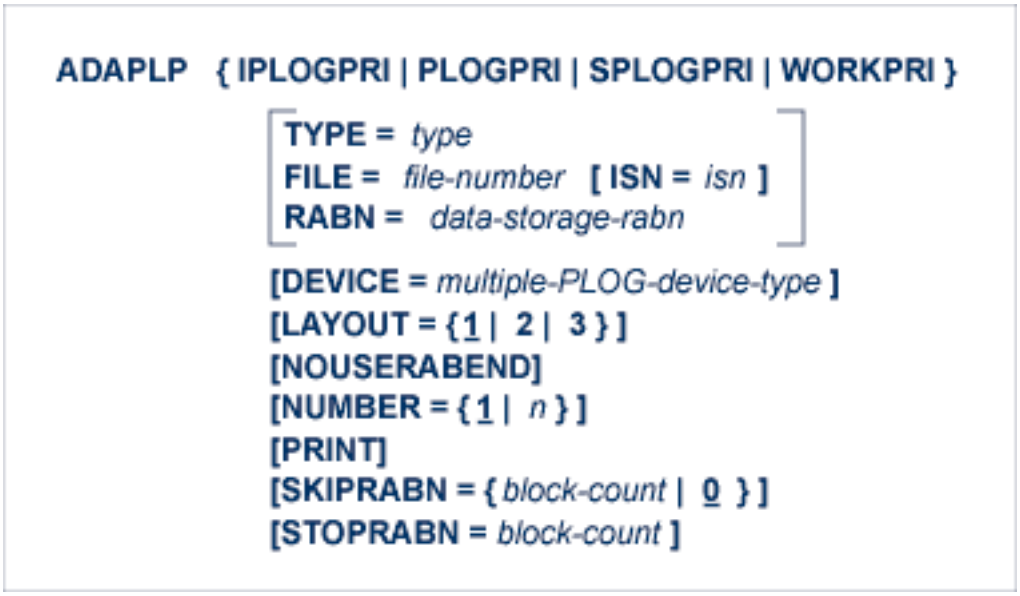
The ADAPLP utility prints data protection records contained on the Adabas Work data set or the Adabas data protection log.

Note: The ADAPLP utility is primarily a debugging tool for our product developers. Consequently, detailed information about its output is not documented.

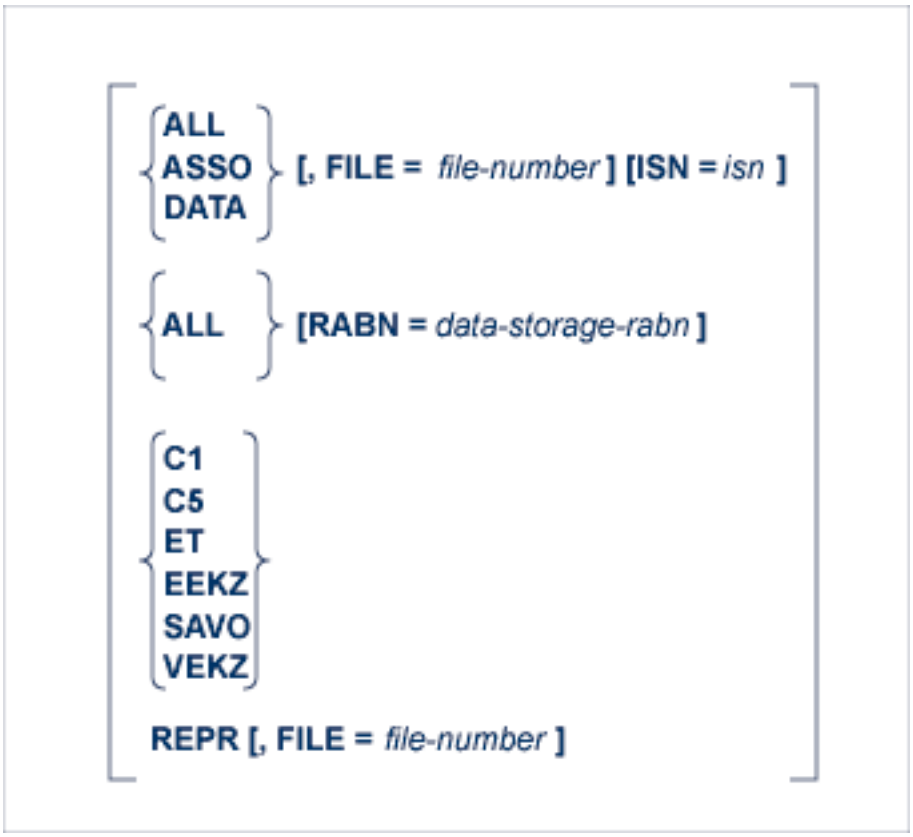
135 ADAPLP Syntax and Examples

- Optional Parameters and Subparameters 961
- Examples 1033

The following diagram shows the ADAPLP syntax for specifying sequential intermediate (IPLOGPRI), multiple (PLOG..) or sequential (SPLOG..) protection logs, or Work data set printing:



where *type* is one of the following:



The IPLOGPRI function is used to print the sequential intermediate data sets created from the PLOG merge process. Input to ADAPLP IPLOGPRI must be a MERGIN1/MERGIN2 data set created by the ADARES utility and is specified in the DDPLOG DD JCL statement. Operation is similar to the SPLOGPRI function.

Optional Parameters and Subparameters

DEVICE: Device Type

DEVICE specifies device type on which the multiple protection data set to be printed is contained. This parameter is required only if the device type is different from the standard ADARUN device.

FILE: File for Which Data is to Be Printed

The FILE parameter can be used to limit printing to those protection records containing information about the specified Adabas file.

The FILE parameter cannot be specified with the RABN parameter, or when TYPE=C1, C5, ET, EEKZ, SAVO, or VEKZ is specified. Do not specify ISN with the RABN parameter.

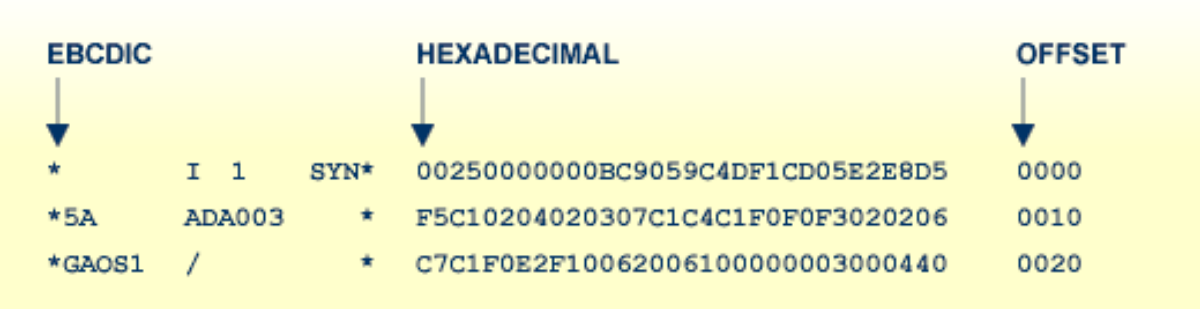
ISN: ISN for Which Data is to Be Printed

This parameter may be used to limit printing to the protection record identified by the specified ISN. The ISN parameter cannot be specified when the RABN parameter is specified, nor when TYPE=C1, C5, ET, EEKZ, SAVO, or VEKZ is specified.

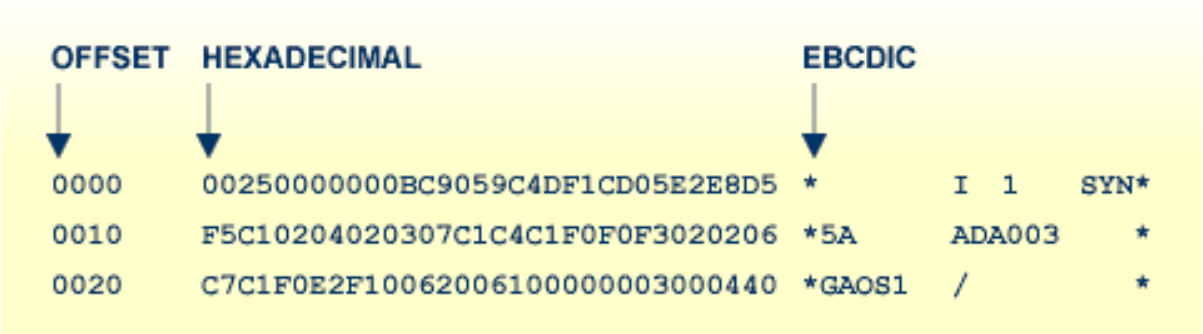
LAYOUT: Print Format

Controls the output format of the protection log record requested by the PRINT parameter. Specify either layout 1 (the default), 2, or 3:

LAYOUT=1 (the Default)



LAYOUT=2/3



LAYOUT=3 presents the same format as LAYOUT=2, and also includes an explanation of each PLOG record type.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

NUMBER: Multiple Protection Log Data Set Number

NUMBER specifies the one of multiple (two through eight) protection log (PLOG) data sets to be printed. Specifying "2" selects the DDPLOGR2 data set; specifying "3" selects the DDPLOGR3 data set, etc. The default of "1" selects DDPLOGR1.

PRINT: Print Entire Data Protection Record

The PRINT parameter prints the entire data protection log record. If this parameter is omitted, only the protection log record header is printed.

RABN: Print Only Updates for the Specified Data Storage Block

The RABN parameter can be used to track all updates to a particular Data Storage block that might be in error.

The parameter limits printing to the protection records that describe the before and after images of Data Storage records that have been removed from, updated in, or added to the specified Data Storage block.

The RABN parameter can only be specified for TYPE=ALL.

SKIPRABN: Number of Blocks to Be Skipped

SKIPRABN specifies the number of blocks to be skipped before printing starts. Counting for the number of blocks to be printed (see STOPRABN parameter) begins after the number of blocks specified with this parameter have been skipped.

STOPRABN: Number of Blocks to Be Printed

STOPRABN limits the number of blocks to be printed. If this parameter is omitted, all blocks up to the end of the protection log are printed. In addition to the RABN count specified by STOPRABN, RABN 1 is also printed; therefore, the total number of printed RABNs is always one more than the value specified by STOPRABN.

TYPE: Type of Record to Be Printed

TYPE specifies the type of protection records to be selected for printing. The following values may be specified:

ALL	all protection records-the default
ASSO	Associator protection records
DATA	Data Storage protection records
C1	records resulting from Adabas C1 commands
C5	records resulting from Adabas C5 commands
EEKZx	records written at completion of a nucleus buffer flush
ET	records resulting from Adabas ET commands
REPR	Work data set records used by autorestart to repair the index
SAVO	online SAVE database/file records
VEKZ	records written at completion of update commands



Notes:

1. The number of protection records is reduced further by specifying the FILE or ISN parameters.
2. The RABN parameter can only be specified for TYPE=ALL.

Examples

Example 1:

```
ADAPLP WORKPRI PRINT,TYPE=ALL,STOPRABN=40
```

41 data protection records from the Adabas Work are to be printed.

Example 2:

```
ADAPLP WORKPRI PRINT,TYPE=ASSO,STOPRABN=10
```

11 Associator data protection blocks from the Adabas Work are to be printed.

Example 3:

```
ADAPLP PLOGPRI PRINT
```

All data protection blocks contained on one of multiple protection log data sets are to be printed.

136 JCL Requirements and Examples

- Requirements to Run ADAPLP JCL 968

This section describes the job control information required to run ADAPLP on the z/OS operating system and shows examples of job streams.

Requirements to Run ADAPLP JCL

Data Set	DD Name	Storage	More Information
Associator	DDASSORn	disk	Required for WORKPRI.
Work	DDWORKR1 DDWORKR4	disk	Required for WORKPRI.
Sequential protection log	DDPLOG	tape/ disk	Required for SPLOGPRI or IPLOGPRI.
Multiple protection log	DDPLOGR1	disk	Required for PLOGPRI if NUMBER=1 (the default).
Multiple protection log	DDPLOGRn	disk	Required for PLOGPRI if NUMBER=n.
ADAPLP messages	DDDRUCK	printer	*Messages and Codes*
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADARUN parameters	DDCARD	reader	*Operations*
ADAPLP parameters	DDKARTE	reader	

ADAPLP JCL Examples

Refer to the JOBS data set for the following example jobs:

Job Member	Description
ADAPLP	Print protection log (from multiple data set PLOG)
ADAPLPS	Print protection log (from sequential PLOG)
ADAPLPW	Print Adabas Work

These jobs are listed in the following sections.

Print One of Multiple Protection Log Data Sets

```
//ADAPLP    JOB
//*
//*   ADAPLP: PROTECTION LOG PRINT (FROM MULTIPLE PLOG)
//*
//PLP       EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD      <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <=== WORK
```

```
//DDPLOGR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.PLOGR1 <=== PLOG DATASET
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADAPLP,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADAPLP PLOGPRI
/*
```

Print Sequential Protection Log

```
//ADAPLPS    JOB
//*
//*   ADAPLP: PROTECTION LOG PRINT (FROM SEQUENTIAL PLOG)
//*
//PLP       EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD      <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <=== WORK
//DDPLOG    DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.PLOG,  <=== PLOG DATASET
//             UNIT=TAPE,VOL=SER=PLOG5
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADAPLP,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADAPLP SPLOGPRI
/*
```

Print Sequential Intermediate Protection Log

```
//ADAPLPS    JOB
//*
//*   ADAPLP: PROTECTION LOG PRINT (FROM SEQUENTIAL PLOG)
//*
//PLP       EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD      <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <=== WORK
//DDPLOG    DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.PLOG,  <=== PLOG DATASET
//             UNIT=TAPE,VOL=SER=PLOG5
```

```
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADAPLP,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADAPLP IPLOGPRI
/*
```

Print Adabas Work

```
//ADAPLPW    JOB
//*
//*   ADAPLP: PRINT ADABAS WORK
//*
//PLP       EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD      <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <=== WORK
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADAPLP,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADAPLP WORKPRI PRINT
/*
```

XV ADARAI Utility: Adabas Recovery Aid

The following functions are available for use with ADARAI:

Functional Overview
CHKDB: Check the Database Recovery Status
DISABLE: Disable Recovery Logging
LIST: Display Current RLOG Generations
PREPARE: Initialize and Start the RLOG
RECOVER: Build a Recovery Job Stream
REMOVE: Remove the Recovery Aid
JCL Requirements and Examples

137 Functional Overview

- Concepts and Components 974

The ADARAI utility prepares the recovery log (RLOG), lists the information contained in the RLOG, creates the job control statements to recover the database, and disables ADARAI logging.

Transaction recovery is provided whenever an Adabas session is abnormally terminated. The Adabas autobackout routine, which is automatically invoked at the beginning of every Adabas session, removes the effects of all interrupted transactions from the database. See the restart/recovery information in the *Adabas Operations* documentation.

However, when a database data set (ASSO, DATA, or WORK) is destroyed, it is necessary to restore and regenerate the database to recover the lost data.

The Adabas Recovery Aid utility, ADARAI, can be used to automate and optimize *database* recovery. It records and reports all information needed to recover the database and builds the recovery job stream (JCL), which is the basis for reexecuting the jobs performed from the time of the last SAVE to the point of failure and error.

Concepts and Components

The Adabas Recovery Aid comprises two components:

- an interface (ADARAC) to collect information as relevant events occur against the database; and
- a utility (ADARAI) to list the information collected, generate jobs to recover the database or files on the database, or deactivate recovery logging.

This section covers the following topics:

- The Collection Interface
- Recovery Log (RLOG)
- Generation: The Unit of Recovery
- Retaining Noncurrent Generations

The Collection Interface

The collection interface is called by the nucleus and by all utilities to record information about each event that occurs; for example, a nucleus stop/start, a utility execution, or an event generated by the Adabas Online System.

Recovery Log (RLOG)

The interface records all event information into a Recovery log file (RLOG) for use by the utility component. The RLOG stores the information about data sets, utility parameters, and protection logs needed to build the recovery job control. The RLOG data set is DDRLOGR1.

In a nucleus cluster environment, all nuclei use the same RLOG. Concurrent updates to the RLOG are controlled by a lock.



Notes:

1. Sequential data sets used by the utilities whose runs are logged on the RLOG must be kept and available for any recovery operation; for example, the DDEBAND input to an ADALOD LOAD operation.
2. ADADBS file changes are now recorded on the RLOG data set.
3. Information recorded in the RLOG generally exceeds that required for recovery; it can also be used as a record of events that have occurred on a database over a period of time.

Generation: The Unit of Recovery

Information is stored on the RLOG by generation, the logical unit used for recovery.

A *generation* includes all activity between consecutive operations of:

- ADASAV SAVE/RESTORE (database),
- RESTORE GCB, or
- SAVE DELTA/RESTORE DELTA (database).

The first generation includes the first operation and extends to (but excludes) the second. A new generation is started when a database can be recovered in full after the previous operation.

Generations may be normal, restricted, or erroneous:

- A generation is labeled "normal" if a full save was available when it started and no unusual events occurred while activities were being logged in it.
- A generation is labeled "restricted" when certain events occur during the logging cycle that make it impossible for ADARAI to rebuild the database without user intervention. ADARAI generates a job, but the job will not run without help from the user. For example, if the Work data set is decreased in size, the user must create a Work data set with the original size so that the recovery job can run correctly up to the point where the Work data set size was decreased.
- A generation is labeled "erroneous" when errors occur during the logging cycle, for whatever reason. ADARAI generates a job, but the job will not run without changes.



Note: When a generation becomes restricted or erroneous, we recommend that you start a new generation as soon as possible by performing an on- or off-line save of the database. If the Adabas Delta Save Facility is installed, a SAVE DELTA will start a new generation.

Retaining Noncurrent Generations

Noncurrent generations provide a history of operations that have affected the database for use in problem resolution or for audit purposes.

Access to noncurrent generations is essential if an attempt to recover a database fails after the RESTORE step in the recovery job is executed. At this point, the generation being recovered becomes the current generation. If it then becomes necessary to rebuild the recovery job, the generation being recovered will be an older generation.

The RLOG retains the number of generations specified by the MINGENS parameter during the ADARAI PREPARE step. ADARAI recycles generations when the number stored on the RLOG reaches the number specified by the MINGENS parameter.

When a new generation plus those already stored exceed the available RLOG space, one of two events will occur:

- if the minimum number of generations as specified by MINGENS can be maintained, the oldest generation is overwritten; otherwise
- the RLOG is placed *out of service* by setting a flag in the RLOG control block. In this case, data is no longer logged.

138 CHKDB: Check the Database Recovery Status

```
ADARAI  CHKDB  [{ACTIVE | INACTIVE}]
```

The ADARAI CHKDB function checks that the recovery status of the nucleus is the status specified by the CHKDB function (active or inactive). To do this CHKDB issues a command to the nucleus and tests the nucleus response code.

If the command does not provide the expected response code, CHKDB reissues another command after ten seconds. Up to ten commands are issued. If the desired nucleus status (active/inactive) does not occur after ten tries, ADARAI terminates with error 158.

Example:

```
ADARAI CHKDB
```

Tests the recovery nucleus for active status.

139 DISABLE: Disable Recovery Logging



The ADARAI DISABLE function disables recovery logging by setting the RLOG table (control block) to inactive status.

Note: ADARAI DISABLE must be executed with the database inactive.

Following DISABLE, information is no longer recorded in the RLOG and the current generation is ended. The content of the RLOG before DISABLE is maintained and can still be listed or otherwise used for recovery purposes.

Recovery logging can be started again by starting a new generation. See **Generation: The Unit of Recovery** .

Example:

```
ADARAI DISABLE
```

Deactivates all Recovery Aid logging.

140 LIST: Display Current RLOG Generations

- Syntax 983
- Optional Parameters 983
- Examples 984

The ADARAI LIST function is used to view the RLOG contents in table form:

- generations are listed in numerical order;
- RLOG block ranges are listed for each generation; and
- the stop/start dates and times covered by each generation are listed.

The following information is provided for each entry on the RLOG including utility executions and nucleus session start and session stop entries:

- name of the event for which the RLOG entry was written;
- date and time the information was written to the RLOG;
- PLOG number associated with the event (if any);
- PLOG block containing an associated checkpoint (if any);
- parameters specified for the logged event to the DDCARD and DDKARTE statements; and
- details of any files written or read during the logged event.

In a nucleus cluster environment, the PLOG data sets are also listed on nucleus session start entries. The cluster nucleus ID (NUCID) is also listed.

Example:

```
*** 2001-08-21  11:37:08  NUCLEUS PLOG NUMBER=4
*** START NUCLEUS SESSION NUCID 40002

SYNC PLOG BLOCK NUMBER = 1
ACTIVE PLOG DATA SET NAMES: EXAMPLE.DBddddd.PLOGR21
                            EXAMPLE.DBddddd.PLOGR22
```

Syntax

```
ADARAI   LIST   [GENS = { NO | YES } ]
                [RELGEN = { gen-number | gen-number - gen-number } ]
                [RLOGDEV = device ]
```

Optional Parameters

GENS: Generation Print Control

GENS determines whether generation information is listed. GENS=NO lists only the RLOG control information. GENS=YES (the default) lists generation information also.

RELGEN: Relative Recovery Generation Number

RELGEN specifies the *relative* generation number (or range of generation numbers) to be listed. The current generation is always coupled with relative generation "0" (zero). The last completed generation is coupled with relative generation "1"; *two generations ago*, the generation before the last completed generation, is specified as relative generation "2".

Example:

To list the generations ranging from three generations ago to the last complete generation (inclusive), specify RELGEN=3-1.

If the first generation number specified is lower than the second generation number, ADARAI reduces the second generation number to match the first.

Example:

If you specify RELGEN=2-3, ADARAI changes it to RELGEN=2-2.

If RELGEN is not specified, all generations are printed.

The specified generation must currently be in the RLOG. Note, however, that instead of a relative number, each listed generation has an ascending order number, beginning with 1 (the first generation following the start of RLOG operation).

Example:

RELGEN=0 is equivalent to generation number 690; RELGEN=3-2 is equivalent to the generation numbers 687 and 688.

```
I GEN-   I      BLOCK      I                 DATE /TIME                 I
I NUMBER I   FROM       TO I        FROM                       TO       I
I--------I-----------------I-------------------------------------------I
I    690 I    715      715 I 2001-08-20  02:07:13   2001-08-20 08:51:19 I
I    689 I    714      714 I 2001-08-17  18:24:49   2001-08-20 02:03:21 I
I    688 I    713      713 I 2001-08-16  18:24:26   2001-08-17 16:48:16 I
I    687 I    712      712 I 2001-08-15  18:29:09   2001-08-16 12:54:28 I
I    686 I    711      711 I 2001-08-14  18:24:30   2001-08-15 17:45:44 I
I    685 I    710      710 I 2001-08-13  18:32:07   2001-08-14 15:46:25 I
I    684 I    709      709 I 2001-08-13  02:07:15   2001-08-13 18:00:18 I
I    683 I    708      708 I 2001-08-10  18:25:59   2001-08-13 02:03:23 I
I    682 I    707      707 I 2001-08-09  18:36:39   2001-08-10 10:24:14 I
```

RLOGDEV: RLOG Alternate Device

RLOGDEV specifies the device type containing the RLOG file. If the RLOG file is located on the device type specified by the ADARUN DEVICE parameter (the default device type), you do not need to specify RLOGDEV.

Examples

Input Examples

```
ADARAI LIST
```

This example lists all generations in the RLOG.

```
ADARAI LIST RELGEN=15-1
```

LIST displays the last 15 generations (if they are available in the RLOG), not including the current generation (0).

Output Examples

```
A D A R A I   Vv.v  SMv   DBID = 00203  STARTED          yyyy-mm-dd  hh:mm:ss

PARAMETERS:
-----------
   ADARAI LIST RELGEN=0
   RECOVERY LOG FILE FOR DATABASE 203

   START RABN FOR LOG DATA AREA IS     21
   HIGHEST LOG AREA RABN IS            633
   CURRENT VALUE FOR ROTATING RABN IS  23

I GEN-   I   I      BLOCK      I                 DATE /TIME                 I
I NUMBER I S I   FROM       TO I          FROM                       TO       I
I--------I---I-----------------I-------------------------------------------I
```

```
I      3 I N I     23       23 I yyyy-01-13  16:06:28   yyyy-01-13 16:11:35 I
I      2 I N I     22       22 I yyyy-01-09  16:07:10   yyyy-01-13 16:04:13 I
I      1 I N I     21       21 I yyyy-01-09  16:04:41   yyyy-01-09 16:06:16 I
I      0 I R I     20       20 I yyyy-01-09  16:04:07   yyyy-01-09 16:04:30 I
I--------I---I-----------------I-------------------------------------------I

*** yyyy-01-13  16:06:28
*** SAVE DATABASE OFFLINE

   DELTA SAVE ID IS AS FOLLOWS:
    FULL SAVE..............2
    LOW DELTA SAVE NUMBER...0
    HIGH DELTA SAVE NUMBER..0
    DATE WRITTEN............yyyy-01-13
    TIME WRITTEN............16:12:03

   FILES = 1,2,3,19


      ADARUN DBID=203,SVC=249,DEVICE=3390,PLOGRQ=YES
      ADARUN NCLOG=2,CLOGDEV=3390,CLOGSIZE=150
      ADARUN NPLOG=2,PLOGSIZE=1350
      ADARUN PLOGDEV=3390
      ADARUN DSF=YES
      ADARUN UEX2=USEREX2M
      ADARUN PROG=ADASAV

     ADASAV SAVE

    //DDSAVE1  DD DSN=EXAMPLE.ADASAV.FULL.G0058V00,
    //            UNIT=3390,SPACE=(TRK,(5,5)),DISP=NEW,
    //            DCB=(RECFM=VB,BLKSIZE=27998,LRECL=27994),
    //            VOL=SER=(SMS018)
```

DDSAVE1 VOLSER=SMS018 FROM BLOCK=1 (ASSO) TO BLOCK =1598 VOLUME IS ASSOCIATED WITH PLOG NO. 6

```
   DDSAVE1    VOLSER=SMS018    FROM BLOCK=1 (DATA)
                                TO BLOCK  =750
              VOLUME IS ASSOCIATED WITH PLOG NO. 6

*** yyyy-01-13  16:07:09  NUCLEUS PLOG NUMBER=7
*** START NUCLEUS SESSION [NUCID=nnnnn]

   SYNC PLOG BLOCK NUMBER = 5
   [ACTIVE PLOG DATASET NAMES: EXAMPLE.DBddddd.PLOGR21
                               EXAMPLE.DBddddd.PLOGR22]

      ADARUN DBID=203,SVC=249,DEVICE=3390,PLOGRQ=YES
      ADARUN NCLOG=2,CLOGSIZE=150,CLOGDEV=3390
```

```
      ADARUN NPLOG=2,PLOGSIZE=1350
      ADARUN PLOGDEV=3390
      ADARUN DSF=YES
      ADARUN UEX2=USEREX2M
      ADARUN PROG=ADANUC
      ADARUN MODE=MULTI
      ADARUN LOCAL=YES
      ADARUN SPT=NO
      ADARUN LWP=480000
      ADARUN LP=200
      ADARUN TT=1800
      ADARUN TNAE=1800
      ADARUN LBP=80000
      ADARUN NH=500
      ADARUN LFP=60000
      ADARUN LU=65525
      ADARUN NAB=45
      ADARUN LQ=12000
      ADARUN LI=20000
      ADARUN NT=10
      ADARUN NC=300
      ADARUN NU=300
      ADARUN LS=20000
      ADARUN TNAX=1800
      ADARUN CT=300
      ADARUN OPENRQ=NO
      ADARUN LOGGING=NO
      ADARUN LOGCB=NO
      ADARUN LOGSB=NO
      ADARUN LOGFB=NO

      ADARUN IGNDIB=NO
      ADARUN FORCE=NO

*** yyyy-01-13  16:07:18  NUCLEUS PLOG NUMBER=7
*** END NUCLEUS SESSION


   HIGHEST PLOG BLOCK WRITTEN = 7

*** yyyy-01-13  16:07:22
*** COPY MULTIPLE PROTECTION LOG DATASET FOR PLOG 7

      ADARUN DBID=203,SVC=249,DEVICE=3390,PLOGRQ=YES
      ADARUN NCLOG=2,CLOGSIZE=150,CLOGDEV=3390
      ADARUN NPLOG=2,PLOGSIZE=1350
      ADARUN PLOGDEV=3390
      ADARUN DSF=YES
      ADARUN UEX2=USEREX2M
      ADARUN PROG=ADARES,MODE=MULTI

      ADARES PLCOPY OPENOUT
```

```
      ADARES DSIMSIZE=5

    //DDSIAUS1 DD DSN=EXAMPLE.PLOG.G0243V00,UNIT=3390,
    //            SPACE=(TRK,(10,1)),DISP=NEW,DCB=(RECFM=VB,
    //            BLKSIZE=27998,LRECL=27994),
    //            VOL=SER=(SMS018)

    DDSIAUS1   VOLSER=SMS018    FROM BLOCK=1
                                TO BLOCK  =7
                                FROM DATE =yyyy-01-13  17:07:09
                                TO   DATE =yyyy-01-13  17:07:18
               VOLUME IS ASSOCIATED WITH PLOG NO. 7

*** yyyy-01-13  16:07:39  NUCLEUS PLOG NUMBER=8
*** START NUCLEUS SESSION [NUCID=nnnnn]

   SYNC PLOG BLOCK NUMBER = 3
   [ACTIVE PLOG DATASET NAMES: EXAMPLE.DBddddd.PLOGR21
                               EXAMPLE.DBddddd.PLOGR22]


      ADARUN DBID=203,SVC=249,DEVICE=3390,PLOGRQ=YES
      ADARUN NCLOG=2,CLOGSIZE=150,CLOGDEV=3390
      ADARUN NPLOG=2,PLOGSIZE=1350
      ADARUN PLOGDEV=3390
      ADARUN DSF=YES

      ADARUN UEX2=USEREX2M
      ADARUN PROG=ADANUC
      ADARUN MODE=MULTI
      ADARUN LOCAL=YES
      ADARUN SPT=NO
      ADARUN LWP=480000
      ADARUN LP=200
      ADARUN TT=1800
      ADARUN TNAE=1800
      ADARUN LBP=80000
      ADARUN NH=500
      ADARUN LFP=60000
      ADARUN LU=65525
      ADARUN NAB=45
      ADARUN LQ=12000
      ADARUN LI=20000
      ADARUN NT=10
      ADARUN NC=300
      ADARUN NU=300
      ADARUN LS=20000
      ADARUN TNAX=1800
      ADARUN CT=300
      ADARUN OPENRQ=NO
      ADARUN LOGGING=NO
      ADARUN LOGCB=NO
```

```
      ADARUN LOGSB=NO
      ADARUN LOGFB=NO
      ADARUN IGNDIB=NO
      ADARUN FORCE=NO

*** yyyy-01-13  16:09:16  NUCLEUS CHECKPOINT
ENCOUNTERED

    CHECKPOINT IS ON PLOG NUMBER 8 BLOCK NUMBER 4
    SYNS-CHECKPOINT IS 'DELETE FILE'
    FILES = 1

*** yyyy-01-13  16:09:16  NUCLEUS CHECKPOINT ENCOUNTERED

    CHECKPOINT IS ON PLOG NUMBER 8 BLOCK NUMBER 5
    SYNS-CHECKPOINT IS 'DELETE FILE'
    FILES = 2

*** yyyy-01-13  16:10:27  NUCLEUS PLOG NUMBER=8
*** ADABAS UTILITY RUN

    SYNP-CHECKPOINT ID IS 'ADALOD - LOAD'
    SYNP-CHECKPOINT IS FOUND ON PLOG 8 IN BLOCK NO. 6
    FILES = 1

      ADARUN DBID=203,SVC=249,DEVICE=3390,PLOGRQ=YES
      ADARUN NCLOG=2,CLOGSIZE=150,CLOGDEV=3390
      ADARUN NPLOG=2,PLOGSIZE=1350
      ADARUN PLOGDEV=3390
      ADARUN DSF=YES
      ADARUN UEX2=USEREX2M
      ADARUN PROG=ADALOD,MODE=MULTI

      ADALOD LOAD FILE=1
      ADALOD NAME='EMPLOYEES'
      ADALOD MAXISN=1500,DSSIZE=1
      ADALOD TEMPSIZE=15,SORTSIZE=15

    //DDEBAND  DD
DSN=ADABAS.ADAvrs.EMPL,UNIT=3390,DISP=OLD,
    //            VOL=SER=(ADA001)

*** yyyy-01-13  16:11:21  NUCLEUS PLOG NUMBER=8
*** ADABAS UTILITY RUN

   SYNP-CHECKPOINT ID IS 'ADALOD - LOAD'
    SYNP-CHECKPOINT IS FOUND ON PLOG 8 IN BLOCK NO. 7
    FILES = 2

     ADARUN
PROG=ADALOD,MODE=SINGLE,SVC=249,DEVICE=3390,DBID=203
```

```
      ADALOD LOAD FILE=2
      ADALOD NAME='VEHICLES'
      ADALOD MAXISN=1000,DSSIZE=1
      ADALOD TEMPSIZE=15,SORTSIZE=15


    //DDEBAND  DD
DSN=ADABAS.ADAvrs.VEHI,UNIT=3390,DISP=OLD,
    //            VOL=SER=(ADA001)

*** yyyy-01-13  16:11:31  NUCLEUS PLOG NUMBER=8
*** END NUCLEUS SESSION

    HIGHEST PLOG BLOCK WRITTEN = 9

*** yyyy-01-13  16:11:35
*** COPY MULTIPLE PROTECTION LOG DATASET FOR PLOG 8


      ADARUN DBID=203,SVC=249,DEVICE=3390,PLOGRQ=YES
      ADARUN NCLOG=2,CLOGSIZE=150,CLOGDEV=3390
      ADARUN NPLOG=2,PLOGSIZE=1350
      ADARUN PLOGDEV=3390
      ADARUN DSF=YES
      ADARUN UEX2=USEREX2M
      ADARUN PROG=ADARES,MODE=MULTI

      ADARES PLCOPY OPENOUT
      ADARES DSIMSIZE=5

    //DDSIAUS1 DD DSN=EXAMPLE.PLOG.G0244V00,UNIT=3390,
    //            SPACE=(TRK,(10,1)),DISP=NEW,DCB=(RECFM=VB,
    //            BLKSIZE=27998,LRECL=27994),
    //            VOL=SER=(SMS018)

    DDSIAUS1   VOLSER=SMS018    FROM BLOCK=1
                                TO BLOCK  =9
                                FROM DATE =yyyy-01-13  17:07:39
                                TO   DATE =yyyy-01-13  17:11:30
               VOLUME IS ASSOCIATED WITH PLOG NO. 8

A D A R A I  TERMINATED NORMALLY                          yyyy-01-13  16:12:03
```

141 PREPARE: Initialize and Start the RLOG

- Syntax 993
- Essential Parameter 993
- Optional Parameters 993
- Examples 994

The recovery log (RLOG) must be prepared before it can be used. The following steps are required to start the RLOG file:

Step 1. Format the RLOG file using the ADAFRM RLOGFRM function.

Before running ADARAI PREPARE, the RLOG data set must be formatted using the RLOGFRM function of the ADAFRM utility. If it is not, an error 159 is returned.

Step 2. Run the ADARAI PREPARE function to prepare the RLOG.

ADARAI PREPARE must be executed with the database inactive.

The ADARAI PREPARE function is used to define

- the size of the RLOG (the size must be the same as the value of the SIZE parameter of the ADAFRM RLOGFRM function);
- the minimum number of generations to retain (4 is the default); and
- the device type (the default is the device type specified by the ADARUN DEVICE parameter).

Step 3. Run the ADASAV SAVE (database) function to begin the first log generation.

After the PREPARE function executes, logging begins for the initial generation; however, this generation has a *restricted* status because it has not been started by a full database save or restore.

See the *Adabas Operations* documentation for more information about generation statuses.

Syntax

```
ADARAI   PREPARE   RLOGSIZE = size
                   [RLOGDEV = device ]
                   [MINGENS = { count  |  4 } ]
```

Essential Parameter

RLOGSIZE: RLOG Area Size

RLOGSIZE defines the size of the RLOG file in cylinders or blocks. This value must be the same as that defined by the SIZE parameter of the ADAFRM RLOGFRM function. RLOGSIZE *must* be specified; there is no default.



Note: The RLOG data set is limited to 16777215 (x'FFFFFF') blocks/RABNs.

Optional Parameters

MINGENS: RLOG Generation Count

MINGENS specifies the number of logging generations to hold in the RLOG. The RLOG numbers the generations in ascending order starting with "0". The minimum is 4 generations (the default); the maximum is 32.

RLOGDEV: RLOG Device Type

RLOGDEV specifies the device type containing the RLOG file. If the RLOG file is located on the device type specified by the ADARUN DEVICE parameter (the default device type), you do not need to specify RLOGDEV.



Important: If you choose a device type for the RLOG data set that is different from the default, you must specify the RLOGDEV parameter for all ADARES PLCOPY and COPY executions as well.

Examples

Example 1:

```
ADARAI PREPARE MINGENS=4,RLOGSIZE=5
```

This ADARAI PREPARE function defines and initializes the RLOG to hold the minimum of four generations in a log size of five cylinders. The RLOG device defaults to that specified by the ADARUN DEVICE parameter.

Example 2:

```
ADARAI PREPARE
RLOGSIZE=20,MINGENS=20,RLOGDEV=3390
```

This example defines a larger RLOG size (20 cylinders) to hold as many as 20 generations on a 3390 device type.

142 RECOVER: Build a Recovery Job Stream

- Recovery Processing 996
- Optimized Recovery Processing 998
- Requirements 998
- Restrictions 999
- Input Needed for Recovery 1000
- Output from the Recovery Operation 1000
- Executing the RECOVER Function 1001
- File-Level Recovery 1002
- Syntax 1003
- Optional Parameters and Subparameters 1003
- Examples 1121
- Skeleton Job Control 1006
- User Exit to Change JCL 1009
- Prerecovery Checking 1009
- Restarting the RECOVER Function or Recovery Job Stream 1010

The ADARAI RECOVER function builds the job control information (recovery job stream) for recovering the Adabas database or selected database files. The RECOVER function

- reads the PLOG information to determine if a PLCOPY is needed; and
- reads the RLOG to build the recovery job stream from the skeleton job control.

ADARAI RECOVER builds the job stream necessary to restore the database or files to the condition before the RECOVER function was run. The completed job stream is sent to the DDJCLOUT data set.

Where appropriate, ADARAI includes error or information messages in the generated job stream. You must then manually correct the errors before submitting the job. The existence of messages in the job stream is indicated by a nonzero return code from ADARAI RECOVER.



Note: When such errors occur, the job control must be corrected manually.

Recovery Processing

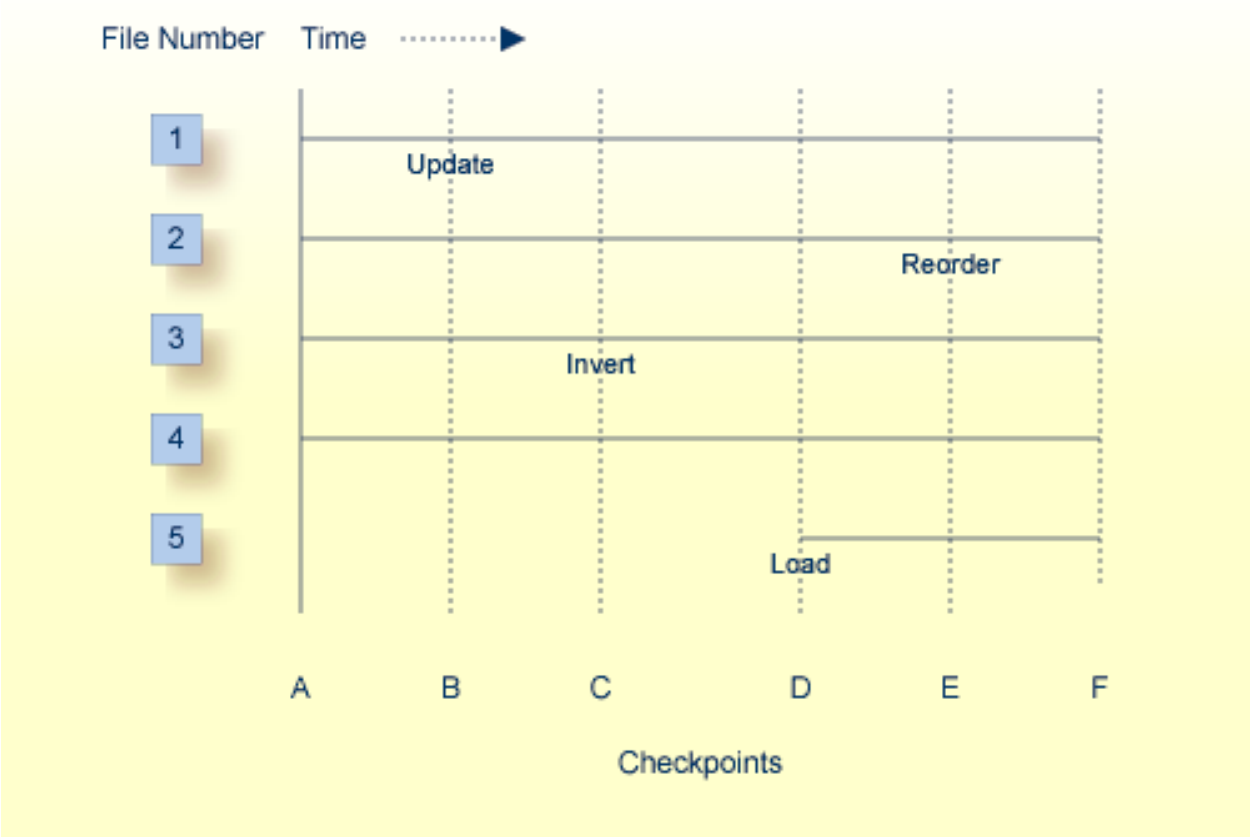
The ADARAI RECOVER function builds a job based on the exact sequence it finds in the generation to be recovered:

- it restores the database from the data sets created by the operation that started the generation;
- it regenerates PLOGs up to the next utility checkpoint found;
- it generates a job step to reexecute the utility and start the regeneration after that checkpoint.

This sequence continues until all utilities have been replayed and the last PLOG block in the generation has been regenerated.

The following diagram illustrates the functioning of ADARAI where

- a database is saved to start a new generation at A.
- the database runs and at various times during the generation
- - an update is run against file 1;
  - a reorder is run against file 2;
  - an invert is run against file 3; and
  - a load is run against file 5.



Given the above, the order of the recovery is as follows:

1. A full save or full save plus delta saves are restored to return the database to the status at A.
2. A database regenerate runs from the first checkpoint at A up to the update checkpoint at B. The regenerate job then terminates.
3. The update utility runs for file 1 and a database regenerate runs between the checkpoint at B and the invert checkpoint at C.
4. The invert utility runs for file 3 and a database regenerate runs between checkpoint C and the load checkpoint at D.
5. The load utility runs for file 5 and a database regenerate runs between checkpoint D and the reorder checkpoint at E.
6. The reorder runs for file 3 and a database regenerate runs between checkpoint E and the most up-to-date level of the database at F.

Optimized Recovery Processing

The OPT parameter for the RECOVER function of ADARAI is used to identify operations or sequences that would minimize the time required to recover a large database.

For example, if a file with 10,000 updates is deleted or reloaded, it should be possible to avoid restoring the file from the start, replaying the 10,000 updates, and then throwing it all away when the delete or load operation occurs.

When optimization is selected, ADARAI does not restore the example file in the main restore for the job. Regeneration for the file occurs only after the file has been deleted or created by the load:

- a deleted file has no more updates;
- for a file created by a load, only updates subsequent to the load are important.

When an optimized job stream is used, the recovered database is rebuilt in a way that is different from the original build. Because optimized recovery jobs do not replay in exactly the same way as the original jobs, problems may occur in the recovery; for example, insufficient space may be available on the database. In most cases, however, the risk is minimal compared with the potential benefits of optimizing the database recovery. Each situation must be examined for potential problems.

Requirements

To generate a recovery job that will run successfully, ADARAI imposes the following conditions:

- the database must be run with dual or multiple protection logging active.
- sequential data sets input to utility functions that update files in the database must be retained.
- sequential output data sets created by SAVE or MERGE functions must be retained. This applies to SAVE FILE functions only if RESTFILE=YES is used for ADARAI RECOVER.
- retained data sets must keep their original names; ADARAI cannot track copies with different names.

we recommend that retained data sets be cataloged.

Restrictions

Shadow Databases

If *shadow* databases, or copies of normal production databases, are built by restoring the delta save output and DSIM data set for a save of the original database, ADARAI has no knowledge of the PLOG activity that occurred during the delta save on the original database and therefore cannot rebuild the DSIM data set if a restore operation becomes necessary on the shadow database.

If, however, the DSIM data set and the delta save data set are merged to create a new *offline* delta save data set and the new merged data set is restored to the shadow database, ADARAI has all the information needed to recover the shadow database since the PLOG is not necessary in this case.

Restoring Delta Saves with a DSIM Data Set

In general, ADARAI handles RESTORE DELTA processing without problems. However, if the RESTORE DELTA uses a DSIM data set (which is essentially a *working* data set), the DSIM data set may not be intact if an ADARAI RECOVER becomes necessary. ADARAI therefore records the COPY or PLCOPY requests used to create a DSIM data set and emits a job step to rebuild the data set before attempting to replay such a RESTORE DELTA during RECOVER processing.

ADARAI searches the entire RLOG for appropriate entries. If the entries cannot be found, ADARAI cannot rebuild the DSIM data set prior to the RESTORE step and therefore cannot replay the RESTORE DELTA.

DDFILEA File

In a generated recovery job, ADARAI writes the DDFILEA file of the ADAORD utility. This cannot be avoided because the REORDER functions must be replayed and they require that the DDFILEA file be written.

In this case, the following restrictions apply:

- ADAORD STORE processing simply reads the same DDFILEA read when the utility was originally run as part of the generation being recovered.
- A temporary file (DISP=(NEW,DELETE), which is deleted after the step is executed) can be used for DDFILEA because the recovery job creates and deletes the file again when it is executed.
- An existing file (DISP=OLD) can be used for DDFILEA. If it still exists when the recovery job is run, ADARAI simply allocates the file with the disposition it had when the original job was run.
- If a new file (DISP=NEW,CATLG) is allocated for DDFILEA and retained in the original ADAORD REORDER step, and if it still exists when the recovery job comes to the REORDER

step (which is normal), ADARAI attempts to create the same file again, which causes the job to fail.

- If a GDG is used, the ADARAI recovery job sees only the name of the actual data set created by the generation. If the data set already exists (which is normal), ADARAI attempts to create the same file again, which causes the job to fail.

Input Needed for Recovery

The following data sets are input to the ADARAI RECOVER function:

- DDRLOGR1, the recovery log.
- DDPLOGR1 and DDPLOGRn, the multiple protection logs, which are required when the ADARAI RECOVER parameter FEOFPL=YES (the default) is used.
- DDJCLIN, which provides site-dependent skeleton job control statements. The RECOVER operation merges these statements with the RLOG information to create a complete database recovery job stream.

The DDJCLIN data set must be defined with RECFM=FB, LRECL=80, and a BLKSIZE that is a multiple of 80 bytes.

Output from the Recovery Operation

The ADARAI RECOVER output is an execution-ready job stream for recovering the database. This recovery job stream is written to the DDJCLOUT file. If a possible error condition is detected during the RECOVER operation, ADARAI issues a warning message and ends with a condition code of 4. See the section **Prerecovery Checking**.

The DDJCLOUT DD statement must point to a data set defined with RECFM=FB, LRECL=80, and a BLKSIZE that is a multiple of 80 bytes.

The recovery job stream includes job steps to start the nucleus

- before the first regenerate job step; and
- after any utility operation that causes the nucleus to terminate automatically.

ADARAI RECOVER jobs replay all utilities with the database active, whether the utility was originally run in single-user mode or not. Utilities originally run in single-user mode are replayed in multiuser mode. These job steps are described in the sections **Building the Recovery Job Stream** and **Skeleton Job Control**.

Executing the RECOVER Function

The RECOVER function is run a generation at a time under control of the RELGEN parameter. If RELGEN is not specified, the default is the current generation.

RECOVER can be executed with the nucleus active or inactive. It can be executed more than once for the same generation because it does not change the RLOG information for that generation.

However, if RECOVER is rerun after a failure while running a DDJCLOUT recovery job stream, the new recovery job stream produced may be different from the original recovery job stream. The reason is that the original recovery job stream may execute utilities against the database that updates the RLOG. The new RECOVER operation then builds a recovery job stream for the utilities that ran as part of the failed recovery job stream.

Also, if the recovery job stream failed after executing an ADASAV RESTORE, a new generation is created. In this case, execute RECOVER using the RELGEN=1 parameter setting to obtain the original generation.

Processing the PLOG

At the start of execution, if ADARAI RECOVER FEOFPL=YES, RECOVER reads the PLOGs, looking for information that must be copied. If necessary, it calls the nucleus to force a PLOG switch. If the nucleus is inactive, it invokes user exit 2.

Reading the Recovery Log

Next, RECOVER reads the skeleton job control into storage and reads the RLOG in chronological order, starting at the beginning of the generation specified by the RELGEN parameter. See **Generation: The Unit of Recovery** for a definition of generation.

If the entire database is being recovered, RECOVER uses the ADASAV SAVE or RESTORE information to create a new RESTORE/RESTONL database operation. For file-level recovery, it uses the SAVE/RESTORE database information to create a RESTORE FILE=... function.

Building the Recovery Job Stream

After creating a job stream for restoring the database or file, RECOVER creates a job step for starting the nucleus, using the %%JCL-STARTNUC statement.

RECOVER then creates the first regenerate job step. This job step does not contain a FROM checkpoint (FROMCP) unless an online SAVE (or DELTA SAVE) was the basis for starting the generation. In that case, the regenerate starts at the end checkpoint (SYN2) of the online save.

All PLOGs up to and including the next utility checkpoint (at which the REGENERATE must stop) are included and appropriate parameters are provided to the ADARES REGENERATE function. If more than 99 PLOGs are to be regenerated, ADARAI generates multiple REGENERATE job steps, each one processing up to 99 input PLOG data sets.

Once the PLOGs are regenerated up to the next utility execution, the utility job step is generated into the output recovery job. ADARAI then inserts another REGENERATE job step that includes all PLOGs up to and including the next utility checkpoint.

The recovery job continues inserting REGENERATE steps and utility steps until it detects the end of the generation specified by the RELGEN parameter. At this point, the completed job stream is sent to the DDJCLOUT file.

File-Level Recovery

Recovery can be made on a file level by specifying the RECOVER function's FILE parameter. The file-level recovery process is essentially the same as the database-level recovery process, but is restricted to the files specified using the FILE parameter.

ADARAI produces a file-specific result in DDJCLOUT by adding parameters to utility execution statements. For example, assume that the following statement was in the original ADASAV RESTORE statement:

```
ADASAV RESTORE FMOVE=2,3,NIRABN=100,1000,DSSIZE=550B,20
```

In this case, RECOVER FILE=3 produces the following DDJCLOUT statement:

```
ADASAV RESTORE FMOVE=2,3,NIRABN=100,1000,DSSIZE=550B,20
ADASAV EXCLUDE=2
```

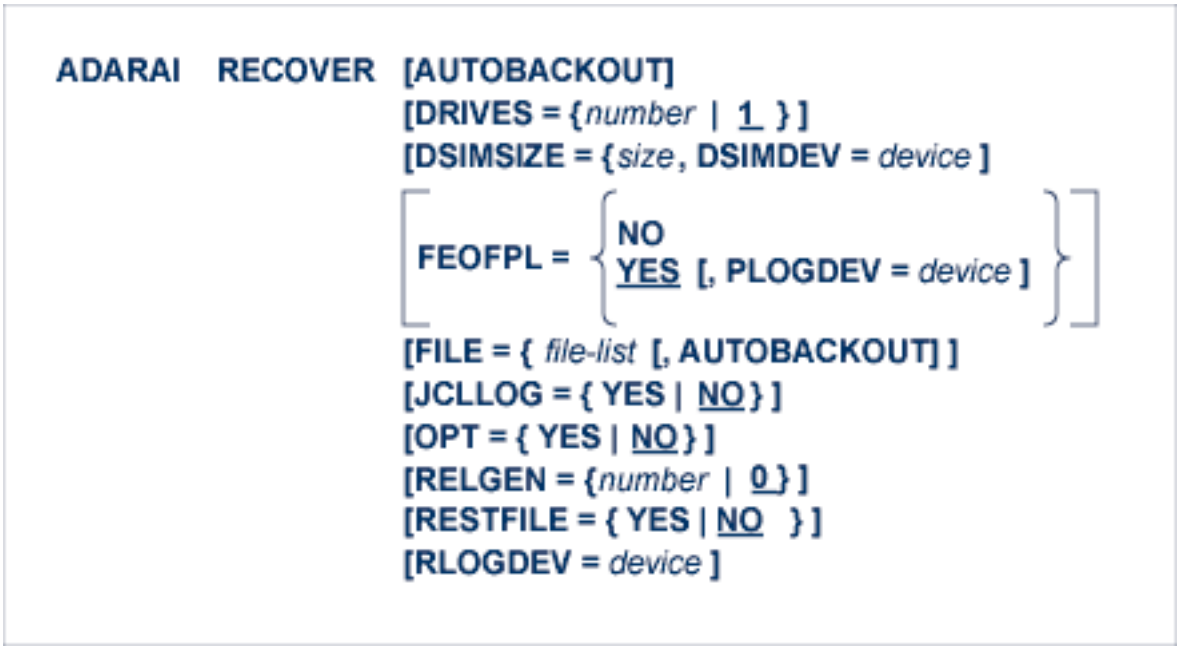


Note: If a file to be recovered is part of an expanded file chain or is coupled, all files in the chain or the coupled list must be recovered together. If all coupled files or expanded file chains are not recovered together, ADARAI detects this and the ADARAI RECOVER function fails.

The Adabas nucleus *must* be active before executing a file-level recovery job. This is different from the database-level recovery job, which starts the database itself.

A file-level RECOVER operation does not create job control for utilities that were executed on the whole database (for example, ADADEF NEWWORK). The exceptions to this are utilities that can be reexecuted for individual files as well as the complete database. An example is ADASAV RESTORE (database), which provides a DDSAVE input data set that can be used to create ADASAV RESTORE FILE=... job control.

Syntax



Optional Parameters and Subparameters

AUTOBACKOUT: Back Out Uncompleted Transactions

AUTOBACKOUT may only be specified for file-level recovery.

If AUTOBACKOUT is specified, transactions that were not complete at the end of the last REGENERATE function in the recovery job are backed out. Only completed transactions are left on the database.

If AUTOBACKOUT is *not* specified, incomplete transactions are left on the database.

For database-level recovery, incomplete transactions at the end of the last REGENERATE function are always backed out.

DRIVES: ADASAV Restore Input Drive Volumes

DRIVES is the number of input data sets to be used as input to the RESTORE step of the recovery job being generated.

The specified DRIVES parameter must be equal to or less than the DRIVES parameter on the job that started the generation. For example, if the generation was started with a database save with DRIVE=4, the RECOVER DRIVES parameter may only be specified as 1, 2, 3, or 4.

When you specify a lower number of DRIVES for the RESTORE step, ADARAI RECOVER allocates only the DDRESTn DDs required and allocates an equal number of input data sets for each DDRESTn DD.

DSIMDEV: DSIM Data Set Device Type

DSIMDEV specifies the DSIM data set device type if different from that specified by the ADARUN DEVICE parameter, which is the default.

DSIMSIZE: Size of the DSIM Data Set

The size is specified in cylinders.

When the Adabas Delta Save Facility Facility is active on the database being recovered, this parameter *must* be specified so that ADARAI can specify the DSIMSIZE parameter for any ADARES COPY operations it may have to generate.

FEOFPL: Synchronize Multiple PLOGs

If FEOFPL=YES (the default), ADARAI ensures that protection log (PLOG) data from all of the multiple PLOG data sets has been copied:

- If the nucleus is active, ADARAI forces a protection log switch. The nucleus then calls user exit 12, which copies the log data; ADARAI waits until the copying is completed. Note that the ADARUN parameter UEX12 must therefore be specified whenever FEOFPL=YES is specified.
- If the nucleus is not active, ADARAI itself calls user exit 12, which in turn copies the log data.

In a nucleus cluster environment, FEOFPL=YES functions differently:

- If at least one Adabas nucleus is available, ADARAI calls the nucleus to switch the PLOGs.
- If no Adabas nucleus is available, ADARAI generates a job that must be executed manually.

In either case, ADARAI must be restarted with FEOFPL=NO.

FILE: File Number

FILE specifies one or more database files to be included when the recovery job stream is built. Specifying FILE causes file- rather than database-level recovery; only those files specified are involved in the RECOVER operation. If FILE is not specified, all database files are included (the default).

JCLLOG: User-Supplied Job Control

JCLLOG controls listing of the user-supplied input job control (the JCL in DDJCLIN). If JCLLOG=YES is specified, the user-supplied input job control elements are printed in the utility log. The default is no listing of input job control statements (NO).

OPT: Optimize Recovery Job for a Generation

When OPT=YES is specified, ADARAI attempts to optimize the recovery job it produces for a given generation; that is, it attempts to leave out steps that are not required to bring the database or file back to its original logical state.

When OPT=NO is specified, the recovery job is not optimized.



Note: When space on the database is limited, an optimized recovery job may fail due to the fact that the database is not built in exactly the same way as it originally was. If this occurs, a recovery job generated without optimization should be used or the size of the database increased before recovery is attempted.

PLOGDEV: Multiple PLOG Device Type

The PLOGDEV value is only used when FEOFPL=YES is specified.

PLOGDEV specifies a PLOG device type different from that specified by the ADARUN DEVICE parameter, which is the default.

RELGEN: Relative Recovery Generation Number

RELGEN specifies the *relative* generation number to be used for recovery. The current generation is always coupled with relative generation "0" (zero), which is also the default. *Two generations ago*, or the generation before the last completed generation, is specified as relative generation "2".

The generation specified must currently be in the RLOG. Use the ADARAI LIST function to see the current RLOG generations available. Note, however, that the listed generations are numbered in ascending order, beginning with generation "1", the first generation following the start of RLOG operation.

RESTFILE: Create Restore File Jobstep

When RESTFILE=NO (the default), the DDJCLOUT recovery job stream does not include ADASAV RESTORE FILE=... job steps for logged ADASAV SAVE FILE= runs. Such job steps are not included because ADARES REGENERATE does not stop at ADASAV SAVE FILE=... checkpoints.

When RESTFILE=YES, ADARAI RECOVER creates an ADASAV RESTORE FILE=... job step in the recovery job stream for every ADASAV SAVE FILE=... utility execution logged.



Note: When using RESTFILE=YES, you must retain the file save data sets that are created in the generation.

When both RESTFILE=YES and OPT=YES are specified, the created RESTORE FILE= steps can speed the recovery process because restored files up to the RESTORE step are ignored.

When RESTFILE=YES and OPT=NO are specified, an unnecessary RESTORE step is included in the recovery job. You may wish to generate the recovery job in this way and then manually remove all steps prior to the RESTORE steps for the file(s) that are of interest.

RLOGDEV: RLOG Alternate Device

RLOGDEV specifies the device type containing the RLOG file. If the RLOG file is located on the device type specified by the ADARUN DEVICE parameter (the default device type), you do not need to specify RLOGDEV.

Examples

Example 1:

```
ADARAI RECOVER,DRIVES=3
```

The RECOVER function builds a recovery job stream based on the current generation (0, the default). The SAVE RESTORE portion of the job stream includes statements for three input data sets: DDREST1, DDREST2, and DDREST3.

Example 2:

```
ADARAI RECOVER  FILE=3,4,7,8,11
ADARAI          RELGEN=2, JCLLOG=YES
```

The recovery job stream is based on the third oldest generation; it includes activity for database files 3, 4, 7, 8, and 11 only; and creates a file-level job control. RECOVER also adds the user-supplied job control from data set DDJCLIN to the utility log.

Example3:

```
ADARAI RECOVER,RELGEN=1,OPT=Y
```

The RECOVER function builds a recovery job stream based on the last generation (i.e. the one preceding the current generation). ADARAI removes any unnecessary processing in order to speed up the recovery job.

Skeleton Job Control

Skeleton job control is contained in the DDJCLIN file and is read as input to the RECOVER function. RECOVER merges it with the RLOG information to create the recovery job stream. Skeleton job control usually remains stable and is specific to your operating environment.

Each function in the skeleton job control is identified by a statement with the following format:

```
%%name
```

The name is specific to the function, such as %%JCL-ADASAV or %%JCL-STARTNUC. The job control statements follow the %% *name* statement; they are ended by the next %%JCL statement. Each skeleton section can contain any valid job control statement, including comments or program execution. This ability provides flexibility for the recovery process.

ADARAI does not check the validity of the statements in the skeleton job control. Invalid statements are first apparent when a job control error occurs during execution of the recovery job stream.

Job Header: %%JCL-JOB-HEADER

Job header statements are placed at the beginning of the recovery job stream before any other job control statements.

This job control relates to the complete recovery job and includes statements such as JOB and JOBLIB statements for z/OS.

Job Trailer: %%JCL-JOB-TRAILER

Job trailer statements are placed at the end of the recovery job stream.

If the nucleus was started with the ADARUN UTIONLY=YES parameter as recommended in the %%JCL-STARTNUC section, you may want to provide a statement to execute an ADADBS OPERCOM UTIONLY=NO function in this section to make the database available after the recovery operation (see the skeleton job control examples later in this document).

Step Trailer: %%JCL-STEP-TRAILER

Step trailer statements are placed after each step in the recovery job stream.

DDKARTE Job Control: %%JCL-DDKARTE

The operating-system-dependent DDKARTE statements are included in each job step before DDKARTE parameters generated by ADARAI from the RLOG.

These statements should indicate that the DDKARTE parameters are contained in the job stream.

DDFILEA Job Control: %%JCL-DDFILEA

This (optional) JCL card is provided to avoid problems with ADAORD REORDER processing. As a placeholder, it may be specified to provide a different DD statement to the original DDFILEA statement in the job. If specified, it will be inserted instead of the original DDFILEA statement when an ADAORD REORDER is subsequently encountered.

Utility Job Control: %%JCL-*utility*

These skeleton sections are used to create utility job steps in the recovery job stream. The following utility jobs should be available in DDJCLIN:

%%JCL-ADADEF	%%JCL-ADAORD
%%JCL-ADAINV	%%JCL-ADARES
%%JCL-ADALOD	%%JCL-ADASAV

Each of the sections should contain the following:

- The database files; for example, DDASSOR1, DDDATAR1, DDWORKR1, DDSORTR1, DDTEMPR1, and so on, as needed for the utility execution;
- DDFILEA for ADALOD if used as a DDTEMPR1 overflow file;
- A DDPRINT and DDDRUCK statement or assignment;
- A DDCARD statement or assignment and all required ADARUN parameters; for example, DBID, DEVICE, PROG, SVC, and so on;

- Information needed about the Adabas library or other library.

It is possible to use a partitioned data set (PDS) member for the DDCARD parameters, database files, or libraries.

Job Control to Start the Nucleus: %%JCL-STARTNUC

This job control comprises all the statements needed to start the Adabas nucleus. The RECOVER function uses this job control to create a job step for starting the nucleus before the first regenerate job step and, if the nucleus is not already active, before each call to a utility that requires an active nucleus.

The entire nucleus job must be included in this job control, including

- job statements;
- program execution statements;
- library definitions;
- database file definitions; and
- DDCARD information, including the ADARUN parameters.

This section also requires a method for submitting the nucleus job control to the appropriate job entry system, such as IEBGENER. For examples of this job control, see the %%JCL-STARTNUC sections in the examples of skeleton job control later in this document.

It is also important that this job control contain a way to stop execution of the recovery job stream until the nucleus is actually active. For example, a program can be created to issue a CL (close) command to the database; if a response code 148 (ADARSP148) indicates that the database is not active, the program can wait a specified time and reissue the CL command. The program continues until response code 0 (ADARSP000) occurs, and then ends to allow the next recover step to be performed. You can use the ADARAI CHKDB ACTIVE function for this purpose.

Job Control to Stop the Nucleus: %%JCL-ENDNUC

Whenever it detects a utility that requires an inactive nucleus, RECOVER inserts the %%JCL-ENDNUC job control in the job stream to ADAEND the Adabas nucleus. The ADADBS OPERCOM ADAEND function can also be used to stop the nucleus. If ADADBS OPERCOM is used, these job control statements must contain all necessary statements for running the ADADBS OPERCOM function. Like the Start Nucleus skeleton job control, a method to stop execution of the recovery job stream until the nucleus becomes inactive is also needed; the ADARAI CHKDB INACTIVE function can be used for this purpose.

Special Job Control Statements

The following special keywords/statements are used in the DDJCLIN skeleton job control to control the generation of the DDJCLOUT recovery job stream:

%STEP	When the (optional) %STEP keyword is included on the program execution statement, it generates a step number in the job stream for each job step that also includes the %STEP keyword. The step numbers run in ascending sequence, beginning with 1.
%SEQUENTIAL	*Must* be included in each %% skeleton section that generates a sequential file job control statement. ADARAI creates the necessary sequential job control statement in place of the %SEQUENTIAL statement. If this statement is not included, an error occurs during processing.
%KARTE	*Must* be included in each %% skeleton section where Adabas DDKARTE parameters are generated. ADARAI creates the necessary DDKARTE parameters in place of the %KARTE statement. If this statement is not included, an error occurs during processing.
%DBID	When the (optional) %DBID keyword is included on the program execution statement, it generates the five-digit database ID number. If the database number has less than five digits, the number is padded with leading zeros.

User Exit to Change JCL

ADARAI provides the user exit UEXRAI so that users may change an automatically generated recovery job before submitting it. Changes required might include the device type or the volume name.

UEXRAI obtains control of a JCL record immediately before it is written to DDJCLOUT.

The user exit is called with the following registers set:

R1	JCL record line that is about to be written to DDJCLOUT.
R13	standard 72-byte register save area
R14	return address
R15	entry point

Prerecovery Checking

Check the status of the recovery database and the recovery job stream before starting the recovery job stream.

For database-level recovery, check that

- the existing nucleus session has ended.
- the session entry has been deleted from the ID table.

Note: Any remaining DIB entry or pending nucleus session autorestart can be ignored; it is handled automatically by the initial RESTORE step.

- all required database components (ASSO, DATA, etc.) have been formatted at least once.
- Allocate and format any components changed during the generation to be recovered to the sizes and device types valid at the *beginning* of the generation.
  - Allocate and format any components that have changed size to the largest size used during the generation to be recovered.

For file-level recovery, check that

the nucleus is active. The recovery job created by ADARAI does not start the nucleus automatically.

Restarting the RECOVER Function or Recovery Job Stream

If the ADARAI RECOVER function is interrupted, it can be restarted from the beginning, since the RECOVER function only reads the RLOG and does not change it.

The DDJCLOUT recovery job stream created by the RECOVER function can be restarted as in a normal restore/regenerate process. However, the job stream may need to be edited to remove steps for the utility operations that were successfully completed. Following this, the recovery process can continue (providing the cause of the interruption has been removed), beginning with the failed utility operation.

It is always possible to restart an interrupted recovery job from the beginning. It may also be possible to restart the recovery job at the job step that failed or a few steps earlier, depending on the cause of the error and the job step that contained the error.

143 REMOVE: Remove the Recovery Aid

- Example 1012

ADARAI REMOVE is functionally the same as the old ADARAI NORAI function; either REMOVE or NORAI can be specified.

Note: ADARAI REMOVE/NORAI must be executed with the database inactive.



The ADARAI REMOVE function disables recovery logging by updating the Associator GCBs to indicate that recovery logging (that is, the Recovery Aid) is no longer active in the database, and that information will no longer be recorded in the recover log (RLOG).

Existing RLOG information is maintained and available for listing or recovery operation following REMOVE, up until the next PREPARE operation is performed. Once the ADARAI PREPARE function is executed, all existing RLOG data is lost.

To restart recovery logging after using the REMOVE function, execute the ADARAI PREPARE function followed by an ADASAV SAVE/RESTORE database, RESTORE GCB, and/or SAVE DELTA/RESTORE DELTA (database) function to start a new generation. See the discussion of **ADARAI PREPARE** for information about preparing the RLOG.

Example

```
ADARAI REMOVE
```

Stops all Recovery Aid logging.

144 JCL Requirements and Examples

- Requirements to Run ADARAI JCL 1014

This section describes the job control information required to run ADARAI on the z/OS operating systems and shows examples of job streams.

Requirements to Run ADARAI JCL

Data Set	DD Name	Storage	More Information
Associator	DDASSORn	disk	
Data protection log	DDPLOGRn	tape/ disk	Required for RECOVER only if FEOFPL=YES is specified.
Recovery log (RLOG)	DDRLOGR1	disk	
Job stream input	DDJCLIN	disk	Required only for RECOVER
Recovery job output	DDJCLOUT	disk	Required only for RECOVER
ADARAI messages	DDDRUCK	printer	*Messages and Codes*
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADARUN parameters	DDCARD	reader	*Operations*
ADARAI parameters	DDKARTE	reader	

JCL Examples

Prepare for Recovery Logging (ADARAI PREPARE):

```
//RAIPREP   EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOADLIB
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.ADAyyyyy.ASSOR1
//DDRLOGR1  DD   DISP=SHR,DSN=EXAMPLE.ADAyyyyy.RLOGR1
//DDDRUCK   DD   SYSOUT=A
//DDPRINT   DD   SYSOUT=A
//DDCARD    DD   *
ADARUN PROGRAM=ADARAI,SVC=xxx,DEVICE=dddd,DBID=yyyyy,MODE=MULTI
//DDKARTE   DD   *
ADARAI PREPARE RLOGSIZE=5,MINGENS=5
```

List the RLOG (ADARAI LIST)

```
//RAILIST  EXEC   PGM=ADARUN
//STEPLIB  DD DSN=ADABAS.ADAvrs.LOADLIB,DISP=SHR
//DDASSOR1 DD DSN=EXAMPLE.ADAyyyyy.ASSOR1,DISP=SHR
//DDRLOGR1 DD DSN=EXAMPLE.ADAyyyyy.RLOGR1,DISP=SHR
//DDDRUCK  DD      SYSOUT=A
//DDPRINT  DD      SYSOUT=A
//DDCARD   DD      *
ADARUN PROGRAM=ADARAI,SVC=xxx,DEVICE=dddd,DBID=yyyyy,MODE=MULTI
//DDKARTE  DD      *
ADARAI LIST
/*
```

Create Recovery JCL (ADARAI RECOVER)

```
//ADARAI   EXEC   PGM=ADARUN
//STEPLIB  DD DSN=ADABAS.ADAvrs.LOADLIB,DISP=SHR
//*
//DDASSOR1 DD DSN=EXAMPLE.ADAyyyyy.ASSOR1,DISP=SHR
//*
//DDRLOGR1 DD DSN=EXAMPLE.ADAyyyyy.RLOGR1,DISP=SHR
//DDJCLIN  DD DSN=EXAMPLE.ADAyyyyy.RAIJCL(JCLIN),DISP=SHR
//DDJCLOUT DD      SYSOUT=A
//*
//DDPLOGR1 DD DSN=EXAMPLE.ADAyyyyy.PLOGR1,DISP=SHR
//DDPLOGR2 DD DSN=EXAMPLE.ADAyyyyy.PLOGR2,DISP=SHR
//*
//DDDRUCK  DD      SYSOUT=A
//DDPRINT  DD      SYSOUT=A
//DDCARD   DD      *
ADARUN PROGRAM=ADARAI,SVC=xxx,DEVICE=dddd,DBID=yyyyy,MODE=MULTI
ADARUN UEX2=UEX2
//DDKARTE  DD      *
ADARAI RECOVER JCLLOG=YES,RELGEN=0,DRIVES=2
```

Skeleton Job Control Example

This example can be found in member ADARAIIN of the JOBS data set.

```
%%JCL-JOB-HEADER
  //ADARECOV JOB 5,'ADA-USER',MSGCLASS=X,CLASS=A,REGION=4096K
  //*
  //JOBLIB DD DSN=ADABAS.ADAvrs.LOAD,DISP=SHR
  //*
%%JCL-JOB-TRAILER
  //*
  //* END OF RECOVERY
  //*
  //
%%JCL-STEP-TRAILER
  //*
  //* END OF STEP
  //*
%%JCL-DDKARTE
  //DDKARTE  DD  *
%%JCL-ADADEF
  //*
  //* DEFINE NEWWORK
  //*
  //%STEP   EXEC PGM=ADARUN          ADADEF JOB STEP
  //DDASSOR1 DD  DSN=ADABAS.ADAvrs.ASSOR1,DISP=SHR
  //DDDATAR1 DD  DSN=ADABAS.ADAvrs.DATAR1,DISP=SHR
  //DDWORKR1 DD  DSN=ADABAS.ADAvrs.WORKR1,DISP=SHR
  //DDTEMPR1 DD  DSN=ADABAS.ADAvrs.TEMPR1,DISP=OLD
  //DDSORTR1 DD  DSN=ADABAS.ADAvrs.SORTR1,DISP=OLD
  //DDRLOGR1 DD  DSN=ADABAS.ADAvrs.RLOGR1,DISP=SHR
  //SYSUDUMP DD  SYSOUT=*
  //DDDRUCK  DD  SYSOUT=*
  //DDPRINT  DD  SYSOUT=*
  //DDCARD   DD  *
  ADARUN MODE=MULTI,PROG=ADADEF,DBID=xxxxx,DE=yyyy,SVC=zzz
  //DDKARTE  DD  *
  %KARTE
%%JCL-ADAINV
  //*
  //* INVERT / COUPLE
  //*
  //%STEP   EXEC PGM=ADARUN          ADAINV JOB STEP
  //DDASSOR1 DD  DSN=ADABAS.ADAvrs.ASSOR1,DISP=SHR
  //DDRLOGR1 DD  DSN=ADABAS.ADAvrs.RLOGR1,DISP=SHR
  //DDSORTR1 DD  DSN=ADABAS.ADAvrs.SORTR1,DISP=OLD
  //DDTEMPR1 DD  DSN=ADABAS.ADAvrs.TEMPR1,DISP=OLD
  //SYSUDUMP DD  SYSOUT=*
  //DDDRUCK  DD  SYSOUT=*
  //DDPRINT  DD  SYSOUT=*
  //DDCARD   DD  *
  ADARUN MODE=MULTI,PROG=ADAINV,DBID=xxxxx,DE=yyyy,SVC=zzz
  //DDKARTE  DD  *
  %KARTE
%%JCL-ADALOD
  //*
```

```
  //* LOAD A FILE / MASS UPDATE
  //*
  //%STEP   EXEC PGM=ADARUN          ADALOD JOB STEP
  %SEQUENTIAL
  //DDASSOR1 DD  DSN=ADABAS.ADAvrs.ASSOR1,DISP=SHR
  //DDDATAR1 DD  DSN=ADABAS.ADAvrs.DATAR1,DISP=SHR
  //DDWORKR1 DD  DSN=ADABAS.ADAvrs.WORKR1,DISP=SHR
  //DDTEMPR1 DD  DSN=ADABAS.ADAvrs.TEMPR1,DISP=OLD
  //DDSORTR1 DD  DSN=ADABAS.ADAvrs.SORTR1,DISP=OLD
  //DDRLOGR1 DD  DSN=ADABAS.ADAvrs.RLOGR1,DISP=SHR
  //SYSUDUMP DD  SYSOUT=*
  //DDDRUCK  DD  SYSOUT=*
  //DDPRINT  DD  SYSOUT=*
  //DDCARD   DD  *
  ADARUN MODE=MULTI,PROG=ADALOD,DBID=xxxxx,DE=yyyy,SVC=zzz
  //DDKARTE  DD  *
  %KARTE
%%JCL-ADAORD
  //*
  //* REORDER
  //*
  //%STEP   EXEC PGM=ADARUN          ADAORD JOB STEP
  %SEQUENTIAL
  //DDASSOR1 DD  DSN=ADABAS.ADAvrs.ASSOR1,DISP=SHR
  //DDDATAR1 DD  DSN=ADABAS.ADAvrs.DATAR1,DISP=SHR
  //DDWORKR1 DD  DSN=ADABAS.ADAvrs.WORKR1,DISP=SHR
  //DDRLOGR1 DD  DSN=ADABAS.ADAvrs.RLOGR1,DISP=SHR
  //SYSUDUMP DD  SYSOUT=*
  //DDDRUCK  DD  SYSOUT=*
  //DDPRINT  DD  SYSOUT=*
  //DDCARD   DD  *
  ADARUN MODE=MULTI,PROG=ADAORD,DBID=xxxxx,DE=yyyy,SVC=zzz
  //DDKARTE  DD  *
  %KARTE
%%JCL-ADARES
  //*
  //* PLCOPY / REGENERATE / BACKOUT
  //*
  //%STEP   EXEC PGM=ADARUN
  %SEQUENTIAL
  //DDASSOR1 DD  DSN=ADABAS.ADAvrs.ASSOR1,DISP=SHR
  //DDRLOGR1 DD  DSN=ADABAS.ADAvrs.RLOGR1,DISP=SHR
  //*  Omit DDPLOGRx for Parallel and Cluster Services
  //DDPLOGR1 DD  DSN=ADABAS.ADAvrs.PLOGR1,DISP=SHR
  //DDPLOGR2 DD  DSN=ADABAS.ADAvrs.PLOGR2,DISP=SHR
  //*  Include MERGIN1 and MERGIN2 for Parallel and Cluster Services
  //MERGIN1 DD  DSN=ADABAS.ADAvrs.INTERI,DISP=SHR
  //MERGIN2 DD  DSN=ADABAS.ADAvrs.INTERO,DISP=SHR
  //SYSUDUMP DD  SYSOUT=*
  //DDDRUCK  DD  SYSOUT=*
  //DDPRINT  DD  SYSOUT=*
  //DDCARD   DD  *
```

```
  ADARUN MODE=MULTI,PROG=ADARES,DBID=xxxxx,DE=yyyy,SVC=zzz
  //DDKARTE  DD  *
  %KARTE
%%JCL-DDSIAUS1
  //DDSIAUS1 DD  DSN=ADABAS.ADAvrs.PLCOPY(+1),
  //            DISP=(NEW,CATLG),UNIT=TAPE
%%JCL-DDSIAUS2
  //DDSIAUS2 DD  DSN=ADABAS.ADAvrs.PLCOPY2(+1),     Optional second copy
  //            DISP=(NEW,CATLG),UNIT=TAPE
%%JCL-ADASAV
  //*
  //* RESTORE FILE(S)/DATABASE
  //*
  //%STEP   EXEC PGM=ADARUN          ADASAV JOB STEP
  %SEQUENTIAL
  //DDRLOGR1 DD  DSN=ADABAS.ADAvrs.RLOGR1,DISP=SHR
  //DDASSOR1 DD  DSN=ADABAS.ADAvrs.ASSOR1,DISP=SHR
  //DDDATAR1 DD  DSN=ADABAS.ADAvrs.DATAR1,DISP=SHR
  //DDWORKR1 DD  DSN=ADABAS.ADAvrs.WORKR1,DISP=SHR
  //DDTEMPR1 DD  DSN=ADABAS.ADAvrs.TEMPR1,DISP=OLD
  //SYSUDUMP DD  SYSOUT=*
  //DDDRUCK  DD  SYSOUT=*
  //DDPRINT  DD  SYSOUT=*
  //DDCARD   DD  *
  ADARUN MODE=MULTI,PROG=ADASAV,DBID=xxxxx,DE=yyyy,SVC=zzz
  //DDKARTE  DD  *
  %KARTE
%%JCL-STARTNUC
  //*
  //* START NUCLEUS
  //*
  //%STEP   EXEC PGM=IEBGENER
  //SYSPRINT DD  SYSOUT=*
  //SYSIN    DD  DUMMY
  //SYSUT2   DD  SYSOUT=(*,INTRDR)
  //SYSUT1   DD  DATA,DLM='$$'
  //ADANUC   JOB 5,'ADANUC',CLASS=A,MSGCLASS=X,REGION=6M,TIME=1440
  //NUCxxxx EXEC PGM=ADARUN
  //STEPLIB  DD  DSN=ADABAS.ADAvrs.LOAD,DISP=SHR
  //*
  //DDASSOR1 DD  DSN=ADABAS.ADAvrs.ASSOR1,DISP=SHR
  //DDDATAR1 DD  DSN=ADABAS.ADAvrs.DATAR1,DISP=SHR
  //DDWORKR1 DD  DSN=ADABAS.ADAvrs.WORKR1,DISP=SHR
  //DDRLOGR1 DD  DSN=ADABAS.ADAvrs.RLOGR1,DISP=SHR
  //DDPLOGR1 DD  DSN=ADABAS.ADAvrs.PLOGR1,DISP=SHR
  //DDPLOGR2 DD  DSN=ADABAS.ADAvrs.PLOGR2,DISP=SHR
  //INTRDR2  DD  SYSOUT=(*,INTRDR)
  //SYSUDUMP DD  SYSOUT=*
  //DDPRINT  DD  SYSOUT=*
  //DDCARD   DD  *
  ADARUN PROG=ADANUC
  ADARUN UEX2=UEX2
```

```
  ADARUN DUALPLS=nnn            dual PLOG size
  ADARUN DUALPLD=mmmm           dual PLOG device type
  ADARUN MODE=MULTI
  ADARUN DATABASE=xxxxx         DATA BASE ID
  ADARUN DEVICE=yyyy            ASSOCIATOR DEVICE TYPE
  ADARUN SVC=zzz                SVC NUMBER
  $$
  //*
  //* End of input for internal reader. Check whether nucleus is active.
  //*
  //ADARAI  EXEC PGM=ADARUN
  //STEPLIB  DD  DSN=ADABAS.ADAvrs.LOAD,DISP=SHR
  //SYSOUT   DD  SYSOUT=*
  //DDPRINT  DD  SYSOUT=*
  //DDDRUCK  DD  SYSOUT=*
  //DDCARD   DD  *
  ADARUN MODE=MULTI,PROGRAM=ADARAI,DBID=xxxxx,SVC=zzz,DE=yyyy
  //DDKARTE  DD  *
  ADARAI CHKDB ACTIVE
  //*
%%JCL-ENDNUC
  //*
  //* ADADBS END NUCLEUS
  //*
  //%STEP   EXEC PGM=ADARUN
  //DDASSOR1 DD  DSN=ADABAS.ADAvrs.ASSOR1,DISP=SHR
  //DDDATAR1 DD  DSN=ADABAS.ADAvrs.DATAR1,DISP=SHR
  //DDWORKR1 DD  DSN=ADABAS.ADAvrs.WORKR1,DISP=SHR
  //DDTEMPR1 DD  DSN=ADABAS.ADAvrs.TEMPR1,DISP=OLD
  //DDSORTR1 DD  DSN=ADABAS.ADAvrs.SORTR1,DISP=OLD
  //DDRLOGR1 DD  DSN=ADABAS.ADAvrs.RLOGR1,DISP=SHR
  //DDDRUCK  DD  SYSOUT=*
  //DDPRINT  DD  SYSOUT=*
  //DDCARD   DD  *
  ADARUN MODE=MULTI,PROG=ADADBS,DBID=xxxxx,DE=yyyy,SVC=zzz
  //DDKARTE  DD  *
  ADADBS OPERCOM ADAEND
 //*
 //* Check whether nucleus is inactive.
 //*
 //ADARAI  EXEC PGM=ADARUN
 //STEPLIB  DD  DSN=ADABAS.ADAvrs.LOAD,DISP=SHR
 //SYSOUT   DD  SYSOUT=*
 //DDPRINT  DD  SYSOUT=*
 //DDDRUCK  DD  SYSOUT=*
 //DDCARD   DD  *
 ADARUN MODE=MULTI,PROGRAM=ADARAI,DBID=xxxxx,SVC=zzz,DE=yyyy
 //DDKARTE  DD  *
 ADARAI CHKDB INACTIVE
 //* ↩
```

XVI ADAREP Utility: Database Status Report

This chapter covers the following topics:

Functional Overview
Report Syntax
Processing Save Tape Input
Report Description
JCL Requirements and Examples

145 Functional Overview

The ADAREP utility produces the database status report, which provides information concerning the current physical layout and logical contents of a database or a save tape (if the SAVETAPE parameter is specified).

The information provided in the database status report includes

- database name, number, creation date/time, file status, and current log number.
- in cylinders and blocks, the amount and locations of Associator, Data, and Work space currently used, and allocated but unused.
- Associator and Data Storage RABN information including device type, VOLSER number, file number (if appropriate), and usage (AC, NI/UI, Data Storage, DSST, alternate (only from a save tape), or unused).
- alternate RABN block amounts and locations (only from a save tape).
- by file, a summary of ISN, extent, padding factor, used/unused Associator and Data Storage space, and file options.
- PPT information.
- detailed information (optionally by file) that includes all summary information plus MIN-ISN/MAXISN settings, MUPEX settings, detailed space information, creation and last use date/time, field definition table (FDT) contents, and general or extended checkpoint file information.
- information on logically deleted fields in the database files and identifying logically deleted fields in each database file.

You can also produce an XML document containing the database status report in XML format. This XML document is structured as defined by an XML schema definition file (*adabas.xsd*) provided in the online Community Discussion Forum for Adabas. A stylesheet (*adabas.xsl*) is also provided for your use at the same location. You can transfer these XML files from the Adabas Forum to any system (probably Windows or UNIX) using automated tools or user-written programs in the

programming language of your choice (Natural, Java, .Net, C, Perl, or Python). In addition, the code page of the XML output can be controlled.

To access the Adabas Forum:

1 Access our Community Discussion Forum at *https://techcommunity.softwareag.com/signup/*.

2 Log into the Community Discussion Forum.

3 Expand **Forums - Overview** on the left side of the web page and select **Adabas**.

The Adabas Forum appears in the central section of the web page.

4 Select **Adabas Code Samples** in the Adabas Forum.

All Adabas code samples, including the schema definition file and stylesheet are available for download in this section.

146 Report Syntax

- Optional Parameters 1026
- Examples 1033

```
ADAREP  [REPORT]
        [ {CPEXLIST [, OFFSET = {column-number | 63} ]  |  CPLIST }
             [FROMDATE = yyyymmdd ] [TODATE = yyyymmdd ]
             [FROMSESSION = session ] [TOSESSION = session ]
             [CP=' keyword,... ']]
        [ {FILE = file-list [, LAYOUT = 1 ] | NOFILE } ]
        [LIMCOUNT | NOCOUNT]
        [NOFDT]
        [NOPPT]
        [ { NOLGLIST | NOPHLIST | NOSTD } ]
        [NOUSERABEND] }
        [PRINT={ON|OFF}]
        [SAVETAPE
             [PLOGNUM = protection-log-number  [ { SYN1 | SYN4 } = PLOG-block-number ] ] ]
        [OUTPUT='XML[,UTF8]']
             [ACODE= '{ GCB | dbcodepg } [ ,{ GCB | FCB | fncodepg } ] [ ,{ GCB | cpcodepg } ] ']
```

This chapter describes the syntax and parameters of the ADAREP utility.

Optional Parameters

ADAREP can be specified alone to retrieve a database status report. You can optionally customize the report by added parameter values.

ACODE: Identify Code Page for XML UTF-8 Output

Use the ACODE parameter only if the UTF8 option is specified on the OUTPUT parameter. This parameter identifies the code page to which the database name, file names, and checkpoint names should be converted. This parameter is specified in three parts, in a specific order, and with each part separated by a comma from the others:

```
ACODE='database-name-codepage,file-name-codepage,checkpoint-name-codepage'
```

The *database-name-codepage* can either be the literal "GCB" or a valid code page number; the *file-name-codepage* can either be the literal "GCB", the literal "FCB", or a valid code page number; the *checkpoint-name-codepage* can either be the literal "GCB" or a valid code page number:

- A setting of "GCB" for any part indicates that the code page for that part should be acquired from the code page in the GCB (the GCB ACODE setting).
- A setting of "FCB" for the *file-name-codepage* indicates that the code page for converting file names in the XML document to UTF-8 should be acquired from the code page setting in the FCB (the FCB ACODE setting).

If code pages for all three parts of the ACODE parameter are being specified, they must be specified in the order shown above. However, if you only need to specify the code page of the database name conversion, you only need to specify the first part. If you only need to specify the code page of the database name and file name conversions, you only need to specify the first two parts. If you don't need to specify the code page for an earlier part in the sequence,

simply specify a comma for that part instead. For example, if you only need to specify the code page setting for the checkpoint name UTF-8 conversion, you might specify:

```
ACODE=',,37'.
```

Note the two commas specified prior to the checkpoint code page number, 37. These identify the placement of the blank database name and file name code page specifications.

If the ACODE parameter is specified but the OUTPUT parameter is omitted, a setting of OUTPUT='XML,UTF8' is assumed.

CPEXLIST: Print Checkpoint List in Extended Format / CPLIST: Print Checkpoint List in Normal Format

These parameters are used to print the checkpoint list in normal (CPLIST) or extended (CPEXLIST) format. Either CPEXLIST or CPLIST must be specified to display checkpoint information. CPEXLIST adds the following information to the normal CPLIST information, depending on the checkpoint origin:

- (Utility or Adabas Online System/Basic Services) function name;
- Checkpoint-specific data.

If the CPEXLIST report is to be displayed, OFFSET can also be specified for a more readable display.

The FROMDATE, TODATE, FROMSESSION, and TOSESSION parameters may be used to indicate the range of checkpoints to be printed.

FILE or NOFILE: File Information to be Displayed or Suppressed

FILE defines the list of files for which status information is to be printed or displayed. If this parameter is omitted, status information for all files will be included.

If NOFILE is specified, the printing of all file and field description information is suppressed.

FROMDATE / TODATE: Start / End Checkpoint Dates for Report

When CPLIST or CPEXLIST is specified, specific start and/or end dates for checkpoint information can be specified. Examples of valid *yyyymmdd* date specification are:

```
ADAREP FROMDATE=19960101,TODATE=19960228    January 1-February 28, 1996
ADAREP FROMDATE=19951111              November 11, 1995 to checkpoint file end
ADAREP TODATE=19951223             From checkpoint file begin to (and including)
                                       December 23, 1995
```

If FROMDATE is not specified, the report begins with the earliest checkpoint information in the system (or with the first on the FROMSESSION session, if later); if TODATE is not specified, the report continues up to the most recent checkpoint (or ends with the last on the TOSESSION session, if earlier).

FROMSESSION / TOSESSION: Start / End Session for Report

Specify a start and/or end session number. Sessions before FROMSESSION and/or after TOSESSION session numbers are not included in the report information. If FROMSESSION

is not specified, the report begins with the earliest checkpoint information in the system (or with the first on the FROMDATE date, if later); if TOSESSION is not specified, the report continues up to the last checkpoint (or ends with the last on the TODATE date, if earlier).

CP: Checkpoint Classes to be Reported

With the CP parameter it is possible to limit the number of reported checkpoints to a certain class or classes (when a list of keywords is specified). Valid keywords are:

- **ADADBS**
  which limits the checkpoints to those where the originator is ADADBS.
- **ADADEF**
  which limits the checkpoints to those where the originator is ADADEF.
- **ADAINV**
  which limits the checkpoints to those where the originator is ADAINV.
- **ADALOD**
  which limits the checkpoints to those where the originator is ADALOD.
- **ADAORD**
  which limits the checkpoints to those where the originator is ADAORD.
- **ADARES**
  which limits the checkpoints to those where the originator is ADARES.
- **ADASCR**
  which limits the checkpoints to those where the originator is ADASCR.
- **ADASAV**
  which limits the checkpoints to those where the originator is ADASAV.
- **ADAZAP**
  which limits the checkpoints to those where the originator is ADAZAP.
- **ADAULD**
  which limits the checkpoints to those where the originator is ADAULD.
- **EXF**
  which limits the checkpoints to those where the originator is an user/utility processing file(s) in exclusive mode.
- **EXU**
  which limits the checkpoints to those where the originator is a user.
- **NUCLEUS**
  which limits the checkpoints to those where the originator is ADANUC.
- **SESSION**
  which limits the checkpoints to type x'5B' created by ADANUC (statistics at end of nucleus session).

- **SYSAOS**
  which limits the checkpoints to those where the originator is SYSAOS.

The originator of a checkpoint can be determined from the table at the end of the section *Checkpoint Information*, elsewhere in this section.

Note: If the checkpoint of type x'5B' was created with Adabas versions prior to Version 8.3, it is not possible to distinguish if the checkpoint creation originated from ADANUC or ADADBS; therefore it will be reported for CP='SESSION' and CP='NUCLEUS' but never for CP='ADADBS'. When the checkpoint of type x'5B' was created with an Adabas Version 8.3 or later, it will be correctly reported for CP='SESSION' and CP='NUCLEUS' and CP='ADADBS'.

LAYOUT: Format Output for Printing

LAYOUT=1 specifies that the "Contents of Database" table should be printed in a single 120-character column format. Normally, the "Contents of Database" and "File Space Allocations" information are presented in two separate sections in the report. However, when LAYOUT=1 is specified, they are merged together into the "Contents of Database" section. In addition, the padding factor is added as well.

The following is an example of how the "Contents of Database" section of the report would appear if LAYOUT=1 is specified. You compare this sample to the "Contents of Database" and "File Space Allocations" sections described in *Contents of the Database: General File Status* and in *File Space Allocations*, elsewhere in this section.

```
************************************************                              ↩
* Contents of Database    68 (EXAMP68         ) *         yyyy-mm-dd  hh:mm:ss ↩
************************************************                              ↩
                                                                              ↩
 File     Name           Loaded      TOP-ISN    MAX-ISN   PADD              ↩
Blocks Allocated      (Blocks Unused)
                                                          A% D%          NI    ↩
   UI         AC       DATA/CYL                                               ↩
                                                                              ↩
    1 EMPLOYEES        2001-12-28       1107       1695  10 10        200    ↩
   15          2        150/1
    1   Extents: NI    1  UI    1  AC    1  DS    1                  138    ↩
    2                   104/0
    2 MISCELLANEOUS    2001-12-28       1779       2543  10 10        200    ↩
   15          3        150/1
    2   Extents: NI    1  UI    1  AC    1  DS    1                  174    ↩
   10                    97/0
    3 VEHICLES         2001-12-28        773       1695  10 10        200    ↩
   15          2        150/1
    3   Extents: NI    1  UI    1  AC    1  DS    1                  183    ↩
    8                   138/0
```

LIMCOUNT or NOCOUNT: Counting of Number of Records Loaded

ADAREP reads the address converter to determine the value for RECORDS LOADED for a file. For very large files, this can result in a large amount of I/O activity. If LIMCOUNT is specified, ADAREP checks the value for TOPISN for the file. If TOPISN is greater than 1000, "NOT COUNTED" appears under RECORDS LOADED.

If NOCOUNT is specified, no value is printed for RECORDS LOADED for any file. If neither LIMCOUNT nor NOCOUNT are specified, ADAREP compiles the exact value for RECORDS LOADED for each file.

NOFDT: Suppress Printing of Field Definitions

The printing of the field definition table (FDT) information for each file is to be omitted. The FDT of the Adabas checkpoint and security files are not printed by ADAREP.

NOLGLIST, NOPHLIST, or NOSTD: Suppress Database Layout Printing

If NOLGLIST is specified, the logical database layout information is to be omitted. If NOPHLIST is specified, physical database layout information is omitted. Specifying NOSTD suppresses all database layout information, and is equivalent to specifying NOLGLIST and NOPHLIST.

NOPPT: Suppress PPT Information

Specify the NOPPT parameter if you do not want PPT information included in the report.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PRINT: Controls the Amount of Output Written to DDPRINT (Default Value is ON)

When the parameter PRINT is set to OFF the output written to DDPRINT is minimized. DDPRINT will contain only start and termination messages and possible errors and warnings.



Note: When PRINT=OFF is specified, the parameter OUTPUT must be utilized.

OFFSET: Defines Extended Report Line Alignment

The OFFSET parameter aligns the beginning of the extended information with the end of the normal checkpoint information for printing on a single line. The default is 63. To display extended information, OFFSET must be reduced or the extended portion may not be displayable within 80 columns.

OUTPUT: Create XML Report

This optional parameter should be used when you want to create an XML version of the database report in addition to the normal one. The format of the XML report can be either EBCDIC or converted to UTF-8:

If you specify:	The XML report format will be:
`OUTPUT='XML'`	EBCDIC
`OUTPUT='XML,UTF8'`	Converted to UTF-8 (including all tags). When UTF-8 format is requested, you may also use the ACODE parameter to specify the specific code page to which the database base name, file names, and checkpoint names should be converted.

When the OUTPUT parameter is specified, additional DD statements (DDXMLR1 and, optionally, DDXMLR2) must be specified in the ADAREP job. DDXMLR1 is required and identifies the data set to which the XML version of the base database report should be written. DDXMLR2 is used to store the XML version of any checkpoint information in the report; it is therefore only required if checkpoint information is requested for the report via the CPLIST or CPEXLIST parameters. The data sets specified by these DD statements must be defined with DCB parameters RECFM=VB, LRECL=512.

The resulting XML documents exactly match the information produced in the regular report, including any filtering done for the report (for example if NOFDT or NOPPT are requested).

Once the XML document is created, it can be transferred to a Windows or UNIX machine and you can use your own code or other tools to transform it.

The XML document is structured as defined by an XML schema definition file (*adabas.xsd*) provided in the online Community Discussion Forum for Adabas. A stylesheet (*adabas.xsl*) is also provided for your use at the same location. You can transfer these XML files from the Adabas Forum to any system (probably Windows or UNIX) using automated tools or user-written programs in the programming language of your choice (Natural, Java, .Net, C, Perl, or Python).

PLOGNUM: Protection Log Number

PLOGNUM specifies the number of the nucleus protection log used during the ADASAV save operation. The number of the nucleus protection log is supplied on the tape, so specifying a value for the PLOGNUM parameter *overrides* the information on the tape.

SAVETAPE: Print Save Tape Report

If SAVETAPE is specified, the report is printed from a save tape. The purpose of the save tape report is to determine what the save tape contains. For more information, see the section **Processing Save Tape Input**.

Some considerations that you should keep in mind:

- You must not specify CPLIST or CPEXLIST with SAVETAPE.
- You must specify NOCOUNT with SAVETAPE, because ADAREP does not count the number of records in a file on a save tape.
- You must specify NOPPT with SAVETAPE.

If you have CPLIST or CPEXLIST specified, ADAREP prints a warning message, changes this option internally, and continues processing. If you do not specify NOCOUNT and NOPPT with SAVETAPE, ADAREP prints a warning message, changes these options internally, and continues processing.

SYN1/SYN4: Beginning Block Number

The SYN1 and SYN4 parameters are mutually exclusive and specify the block number containing the SYN1 or SYN4 checkpoint at which the corresponding save operation began. These block numbers are supplied on the tape, as appropriate, so specifying a value for the SYN1 or SYN4 parameters *overrides* the information on the tape.

Examples

Example 1:

```
ADAREP REPORT
```

All database, file, and field information is to be printed. Checkpoint data is not to be printed.

Example 2:

```
ADAREP REPORT NOFDT
```

Database and file information is to be printed. Checkpoint data and field definitions for each file are not to be printed.

Example 3:

```
ADAREP CPEXLIST,FROMDATE=19980701,TODATE=19980715
ADAREP NOSTD,NOFILE
```

A checkpoint list (extended format) is to be printed for all checkpoints taken between July 1, 1998 and July 15, 1998. No other information is to be printed.

Example 4:

```
ADAREP REPORT SAVETAPE,NOCOUNT
```

All database, file, and field information is to be printed from a save tape. NOCOUNT is required because ADAREP does not count the number of records in a file on a save tape.

Example 5:

```
ADAREP CPEXLIST,OUTPUT='XML'
```

All database, file, field information, and checkpoint data in extended format will be printed in the ADAREP report and a CML version of the report will be produced.

Example 6:

```
ADAREP CPLIST,OUTPUT='XML,UTF8'
```

All database, file, field information, and checkpoint data will be printed in the ADAREP report and an XML version of the report will be produced and converted to UTF-8.

Example 7:

```
ADAREP REPORT OUTPUT='XML,UTF8',ACODE=',37'
```

All database, file, and field information will be printed in the ADAREP report and an XML version of the report will be produced and converted to UTF-8. Code page 37 will be used for the UTF-8 conversion of file names.

147 Processing Save Tape Input

- Supplying Protection Log Input 1036
- Checking Input Tapes 1037
- Concurrent Parameters 1037
- Reports for Delta Save Tapes 1037
- Report Layout 1038

If the SAVETAPE keyword is specified, the report is printed from a save tape.

The save tape

- must have been created using ADASAV version 7.1 or above;
- may have been created online or offline;
- may be a database save, file save, or delta save tape; and
- must be supplied as a DDSAVE sequential input file.

ADAREP does not scan the complete save tape: normally, it is sufficient to supply only the first cassette/tape reel.

ADAREP reads through the save tape to pick up the general control blocks (GCBs), the block of unreadable blocks (BUB), the mirror table, the mirror BUB, the free space table (FST), and all the file control blocks (FCBs). Once these are in main memory, ADAREP continues as for a normal database report. The file definition tables (FDTs) are read from the save tape as they are needed: they are not buffered in main memory.



Note: Adabas versions 7.2 and above do not support and therefore do not save BUB or mirror information. However, BUB and mirror information on save tapes from Adabas 7.1 is still reported.

Supplying Protection Log Input

If an online save tape is used, the corresponding protection log may optionally be specified as a DDPLOG sequential input file:

- If DDPLOG is supplied, ADAREP scans the protection log for FCB and FST blocks to ensure that it has the most recent versions.
- If DDPLOG is not supplied, ADAREP prints a warning message and continues. It displays the database status as of the beginning of the online save operation (time of SYN1/SYN4 checkpoint). Any secondary extents allocated during the online save operation are not reflected in the report because they are only visible on the protection log. In addition, the physical layout section may report phantom errors due to inconsistency in the FCB and FST blocks on the save tape. This happens only if a secondary extent was allocated during the short phase when ADASAV was saving the FCB and FST blocks.

Parameters PLOGNUM and SYN1 or SYN4 identify the protection log number and block number of the SYN1 or SYN4 checkpoint. The information provided by these parameters is supplied on the tape, so specifying PLOGNUM or the SYN1 or SYN4 parameters *overrides* the information on the tape.

When DDPLOG is supplied, two tape units are needed in parallel: it is not possible to concatenate the save tape and the protection log as for ADASAV RESTONL.

Checking Input Tapes

After opening the DDSAVE and DDPLOG input data sets, ADAREP cross-checks to ensure that the input tapes are correct:

- If an invalid save tape is supplied, ADAREP terminates and displays error-128 (invalid save tape supplied).
- If an invalid protection log is supplied, ADAREP displays an appropriate warning message, sets the condition code to 4, and continues.

Concurrent Parameters

CPLIST/ CPEXLIST information and the number of records loaded for a file cannot be printed from the save tape. If the CPLIST/ CPEXLIST parameter is specified or the NOCOUNT parameter is not specified with SAVETAPE, ADAREP prints a warning message, changes these options internally, and continues processing.

The VOLSER number is printed on the report. When the VOLSER is not identifiable, asterisks are printed as VOLSER numbers.

Reports for Delta Save Tapes

For delta save tapes, much of the information is either inaccessible or must be reconstructed:

- The delta save status is always *enabled*; the DLOG area usage is only one block (the header) which is displayed as "n%".
- The last full save number, last delta save number, and the date/time of the last delta save are taken from the DSID.
- The estimated number of changed blocks is MAXFILES times 5 plus 30 rounded to the next multiple of 100.
- The DLOG area location is derived from the GCBs.
- The date/time of last full save cannot be reconstructed and is always displayed as "unknown".

Report Layout

The purpose of the save tape report is to determine what the save tape contains.

The save tape report is preceded by a short header indicating the kind of save tape supplied, whether it was created online or offline, when it was created, the version of ADASAV used to create it, the database ID on the save tape, and possibly the delta save identifier of the save tape. For online save tapes, the session number of the corresponding protection log and the block number of the SYN1/SYN4 checkpoint (either supplied or derived from the tape) is displayed.

```
A D A R E P    Vv.r   SMs     DBID = nnnnn  STARTED           yyyy-mm-dd   hh:mm:ss ↩


PARAMETERS:
-----------

ADAREP REPORT SAVETAPE

************************************************************
*                                                          *
*  REPORT FROM            ONLINE DATABASE SAVE             *
*  CREATED AT             yyyy-mm-dd  hh:mm:ss             *
*  BY ADASAV VERSION      V vr                             *
*  DBIB                   nnnnn                            *
*  DSID                   1 / 0 /   yyyy-mm-dd  hh:mm:ss   *
*  PLOG SESSION NR        17                               *
*  SYN1 BLOCK NR          137                              *
*                                                          *
************************************************************

**********************
*                    *
*  DATA BASE REPORT  *                                     yyyy-mm-dd   hh:mm:ss ↩

*                    *
**********************
```

The database ID printed in the first line of the report is taken from the ADARUN DBID parameter. This DBID is *not* cross-checked with the database ID on the save tape. Instead, the save tape DBID is used throughout the report once the save tape is opened and the GCBs read.

The physical layout for file save reports is a table of RABN ranges indicating how each RABN in the database is used. Because a file save tape contains only the FCBs of the saved files, gaps exist in the physical layout table and are reported as "unknown" ranges rather than errors.

148 Report Description

- General Database Information 1040
- File Information 1054
- Checkpoint Information 1067

The ADAREP database status report contains general database information followed by information about the status, allocation, and definition of each file in the database. Although the report is designed for printing from the DDDRUCK data set, the following figures show examples of the report output displayed at a terminal. The examples display sections in the order they appear in the report; a description of each part is provided with them.

Note: Individual Adabas add-on products may supplement the information displayed on the ADAREP report. For example, if the database supports replication (via the Event Replicator for Adabas), additional statistics appear in various areas of this report. For complete information on the impact of the add-on products to this report, refer to the documentation for the Adabas add-on product.

General Database Information

The first section contains general information about the database and its physical layout:

```
********************
* *
* Data Base Report * yyyy-mm-dd hh:mm:ss
* *
********************
Data Base Name = EXAMPLE-DB
Data Base Number = 238
Date Loaded = yyyy-mm-dd
Time Loaded = hh:mm:ss
Checkpoint File = 5
Security File = 4
Maximum number of files = 255
Number of files loaded = 130
Current Log Tape Number = 184
RABNSIZE = 3
Recovery Aid = No
Universal Encoding Sup. = No
Replication = Yes
```

Additionally, if one of the following features is enabled, the corresponding information is displayed.

If universal encoding support (UES) is enabled (Universal Encoding Sup. = Yes), the following information is displayed:

```
Universal Encoding Sup. = Yes
 ALPHA FILE ENCODING     =       37
 WIDE FILE ENCODING      =     4095
 ALPHA ASCII ENCODING    =      437
 WIDE USER ENCODING      =     4095
 Replication             = No
```

If UES=NO, this information is suppressed.

If a default time zone is defined for the database, the following information is displayed:

```
Default Timezone        = AMERICA/DENVER
```

If Default Timezone=' ', this information is suppressed.



Note: When the default time zone is defined for the database the ADAREP utility job must include the TZINFO data definition statement (DD) identifying the TZ library.

If the database is defined as Replicator, the following information is displayed:

```
Replicator              = Yes
```

If Replicator=NO, this information is suppressed.

Field	Explanation
ALPHA ASCII ENCODING	Current user encoding set for alphanumeric (A) format fields in the database. Must be ASCII-compatible.
ALPHA FILE ENCODING	Current file encoding set for alphanumeric (A) format fields in the database. Must be EBCDIC-compatible.
CURRENT LOG TAPE NUMBER	Number of the most recent data protection log for the database.
DATABASE NAME	Name assigned to the database. See the ADADEF utility, DBNAME parameter.
DATABASE NUMBER	Number (ID) assigned to the database. See the ADADEF utility, DBIDENT parameter.
DATE LOADED	Date the database was initially defined.
MAXIMUM NUMBER OF FILES	Maximum number of files permitted for the database. See the ADADEF utility, MAXFILES parameter.
NUMBER OF FILES LOADED	Number of files currently in the database.
RABNSIZE	Length of the blocks in the database. RABNSIZE=3 indicates 24-bit blocks; RABNSIZE=4 indicates 31-bit blocks.
RECOVERY AID	Whether the Adabas Recovery Aid (ADARAI) is active for the database.
Replication	Whether or not replication (using Event Replicator for Adabas) is active for the database.
Replicator	Whether the database is defined as Replicator.

Field	Explanation
Replicator File	Identifies the file number of the Replicator system file if one is loaded on an Event Replicator Server.
Reptor SLOG File	Identifies the file number of the SLOG system file if one is loaded on an Event Replicator Server.
SYSTEM FILES	File numbers of Adabas system files.
TIME LOADED	Time of day when the database was initially defined.
Timezone	The default time zone of the database.
TRIGGER FILE	If the database contains a trigger file, this entry displays the file number. If no trigger file exists in the database, this line does not print.
UNIVERSAL ENCODING SUPPORT	Whether universal encoding support (UES) is active for the database.
WIDE FILE ENCODING	Current file encoding set for wide-character (W) format fields in the database.
WIDE USER ENCODING	Current user encoding set for wide-character (W) format fields in the database.

Space Allocated to Database Components

The "physical layout" table lists the space allocations for the major components of the database (Associator, Data Storage, and Work).

The "unused storage" table lists the unused space in the Associator and Data Storage areas. This space is not assigned to any file in the database.

```
P H Y S I C A L  L A Y O U T

       DD- I DEV   I NMBR OF I NMBR OF EXTENTS IN BLK. I BLOCK I NMBR OF I
     NAMES I TYPE  I CYLS    I BLOCKS  FROM TO         I LNGTH I M-BYTE  I
 ----------I-------I---------I-------------------------I-------I---------I
           I       I         I                         I       I         I
    ASSOR1 I 3380  I 100     I 28481    1      28481   I 2004  I 54      I
           I       I         I                         I       I         I
    DATAR1 I 3380  I 200     I 26991    1      26991   I 4820  I 124     I
           I       I         I                         I       I         I
    WORKR1 I 3380  I  40     I  5391    1      5391    I 4820  I 24      I
           I       I         I                         I       I         I
 ------------------------------------------------------------------------

  U N U S E D S T O R A G E

       DD- I DEV   I NMBR OF I NMBR OF EXTENTS IN BLK. I BLOCK  I NMBR OF I
     NAMES I TYPE  I CYLS    I BLOCKS  FROM TO         I LNGTH  I M-BYTE  I
 ----------I-------I---------I-------------------------I--------I---------I
           I       I         I                         I        I         I
    ASSOR1 I 3380  I  98     I 28134   328       28461 I 2004   I 54      I
           I       I         I                         I        I         I
    DATAR1 I 3380  I 198     I 26811   131       26941 I 4820   I 124     I
```

The columns in these tables provide the following information:

Column	Explanation
DDNAMES	The job/task control name (without the DD prefix) that defines the Associator, Data Storage, or Work component of the database.
DEV TYPE	The physical device containing the Associator, Data Storage, or Work component.
NMBR OF CYLS	The DASD cylinders allocated to the Associator, Data Storage, and Work components. If less than one full cylinder has been allocated, "0" is shown in this column.
NMBR OF BLOCKS	The total number of blocks assigned to the Associator, Data Storage, or Work component. Please note that for Data Storage, Associator, and Work, the first track is not used. ADAREP only shows the number of blocks that are used by Adabas, and not the blocks that are allocated and formatted for use.
EXTENTS IN BLK	The extents, listed by block range.
BLOCK LNGTH	The block size. The block size depends on the component and the device type.
NUMBER OF M-BYTES	The component storage size, in megabytes.

When the ADARUN parameter ENCRYPTION=YES is specified, and encryption is supported by the database, an additional encryption summary report will be provided, detailing which of the datasets comprising Asso, Data and Work are encrypted.

This report will appear between the 'Physical Layout' and 'Unused Storage' sections.

```
E N C R Y P T I O N   S T A T U S

      DD- I   Encrypted   I
    NAMES I               I
          I               I
----------I---------------I------------
          I               I
   ASSOR1 I       N       I
   ASSOR2 I       Y       I
          I               I
   DATAR1 I       N       I
   DATAR2 I       Y       I
   DATAR3 I       Y       I
          I               I
   WORKR1 I       N       I
          I               I
---------------------------------------
```

Column	Explanation
DDNAMES	The job/task control name (without the DD prefix) that defines the Associator, Data Storage, or Work component of the database.
ENCRYPTED	'Y' if the dataset is encrypted, 'N' if the dataset is not encrypted, 'unknown' if the encryption status cannot be determined.

Note that, in order to determine the encryption status of the WORK dataset, a DD statement for DDWORKR1 needs to be supplied in the ADAREP JCL. If this is not done, the DDNAME will be printed as 'WORK' and the ENCRYPTED column will contain 'unknown'. If no DD statement for DDDATAR1 is supplied, the DDNAME will be printed as 'DATA' and the ENCRYPTED column will contain 'unknown'.

Contents of PPT Table

When the parallel participant table (PPT) information is included in the report, it appears in the General Database Information section of the report, immediately following the subsection "Space Allocated to Database Components". The PPT information appears as follows:

```
******************
*                *
* Contents of PPT *                                    2009-04-20  2
*                *
******************



PPT RABN Range               =      2413 to     2444
PPT RABN                     =      2413
    NUCID                    =      0000
    Session Status           =  NUCLEUS ACTIVE OR FAILED (WORK NONEMPTY
                                PLOG(S) NOT YET COPIED
                                CLOG(S) NOT YET COPIED
    Number of entries        =         5
    Last Session number      =      0002
    Last PLOG block written  =         0  NOT INITIALIZED
    Next block number        =         0

    PPT Entry number         =   1
       Dataset               =  /USATSM/TSMATA/WORKR1/
       Dataset Type          =  DDWORK1


    PPT Entry number         =   2
       Dataset               =  /USATSM/TSMATA/PLOGR1/
       Dataset Type          =  DDPLOGR1


    PPT Entry number         =   3
       Dataset               =  /USATSM/TSMATA/PLOGR2/
       Dataset Type          =  DDPLOGR2


    PPT Entry number         =   4
       Dataset               =  /USATSM/TSMATA/CLOGR1/
       Dataset Type          =  DDCLOGR1


    PPT Entry number         =   5
       Dataset               =  /USATSM/TSMATA/CLOGR2/
       Dataset Type          =  DDCLOGR2                                    ↩
```

The columns in these tables provide the following information:

Statistic	Explanation
PPT RABN Range	The range of RABNs in the PPT.
PPT RABN	The current RABN.
NUCID	The nucleus DBID.
Session Status	The status of the session.
Number of entries	The number of entries in the PPT.
Last Session number	The number of the last session.
Last PLOG block written	The last PLOG block written.
Next block number	The block number of the next block in the PPT.
PPT Entry number	The number of a PPT entry.
Dataset	The data set name associated with the PPT entry.
Dataset Type	The type of data set associated with the PPT entry.

Contents of the Database: General File Status

The next section contains information on the status of each file in the database. Here is an example:

```
************************************************
* Contents of Database    99 (EXAMPLE-DB)       *         yyyy-mm-dd  hh:mm:ss
************************************************

  File    Name            Loaded      TOP-ISN    MAX-ISN          EXTENTS
                                                                N    U    A    D

     1 EMPLOYEES        2001-12-28       1107       1695     1    1    1    1
     2 MISCELLANEOUS    2001-12-28       1779       2543     1    1    1    1
     3 VEHICLES         2001-12-28        773       1695     1    1    1    1
```

Here is another example showing a database that uses a LOB file:

```
************************************************
* Contents of Database    99 (EXAMPLE-DB)       *         yyyy-mm-dd  hh:mm:ss
************************************************

File    Name            Loaded      TOP-ISN    MAX-ISN          EXTENTS
                                                              N    U    A    D

   1 SQLNC            2006-02-17          0       1377     1    1    1  301 *
   2 BASEFILE         2006-09-01         45       1377     1    1    1    1
   3 LOBFILE          2006-09-01          3       1377     1    1    1    1
   6 abcdefghijklmnop 2006-07-17          0       4133     1    1    1  300

Warning: * indicates less than 10 file extents remain for use
```

When LAYOUT=1 is specified for the ADAREP utility run, this section includes padding factor information merged with all the information in the File Space Allocation section of the report.

The columns in this table provide the following information:

Column	Explanation
FILE	Adabas file number.
NAME	File name (see the ADALOD utility, NAME parameter). If the file cannot build at least ten further extents, it is marked with an asterisk (*) to the right of the name.
LOADED	Date the file was loaded.
TOP-ISN	Highest ISN currently used in the file.
MAX-ISN	Highest ISN that can be assigned to a record in the file (see the ADALOD utility, MAXISN parameter).
EXTENTS	Number of logical extents currently assigned to the normal index (N), upper index (U), address converter (A), and Data Storage (D). The maximum number of logical file extents that you can now define is derived from the block size of the first Associator data set (DDASSOR1). The extent information is stored in a variable section of the FCB. New extents can be added now until the used FCB size reaches the block size of the Associator data set. If the extent limit has been reached, reorder the file (using ADAORD REORFILE or the ADAULD, ADADBS DELETE, ADALOD LOAD utility sequence) before the last extent fills, or Adabas will lock the file.
PADD	The block padding factor defined for the Associator (A%) and Data Storage (D%) (read about the `ASSOPFAC` and `DATAPFAC` parameters of the ADALOD LOAD utility for more information). **Note:** This column appears only when LAYOUT=1 is specified.

File Options

The next section lists the file options that are active for each file in the database. Here is an example:

```
****************
* File Options *
****************


                        ADAM File
                        .  Coupled File
                        .  .  ISNREUSE
                        .  .  .  DSREUSE
                        .  .  .  .  Ciphered File
                        .  .  .  .  .  Expanded File
                        .  .  .  .  .  .  USERISN
                        .  .  .  .  .  .  .  NOACEXTENSION
                        .  .  .  .  .  .  .  .  MIXDSDEV
                        .  .  .  .  .  .  .  .  .  PGMREFRESH
                        .  .  .  .  .  .  .  .  .  .  Multi-Client File
                        .  .  .  .  .  .  .  .  .  .  .  Index Compressed
                        .  .  .  .  .  .  .  .  .  .  .  .  2-Byte MU/PE Index
                        .  .  .  .  .  .  .  .  .  .  .  .  .  Spanned Record
                        .  .  .  .  .  .  .  .  .  .  .  .  .  .  Replicated
                        .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  Priv Use
 File  Name             .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
-------------------------------------------------------------------------------
    1  EMPLOYEES         .  .  I  D  .  .  .  .  .  .  .  .  .  .  .  .
    2  64BIT-1           .  .  .  D  .  .  .  .  .  .  .  .  .  .  .  .
    3  MISC              .  .  I  D  .  .  .  .  .  .  .  .  .  .  .  .
   19  CHECKPOINT-FILE   .  .  .  D  .  .  .  .  .  .  .  .  .  .  .  .
```

Here is another example showing a database that uses a LOB file:

```
****************
* File Options *
****************


                        ADAM File
                        . Coupled File
                        . . ISNREUSE
                        . . . DSREUSE
                        . . . . Ciphered File
                        . . . . . Expanded File
                        . . . . . . USERISN
                        . . . . . . . NOACEXTENSION
                        . . . . . . . . MIXDSDEV
                        . . . . . . . . . PGMREFRESH
                        . . . . . . . . . . Multi-Client File
                        . . . . . . . . . . . Index Compressed
                        . . . . . . . . . . . . 2-Byte MU/PE Index
                        . . . . . . . . . . . . . Spanned Record
                        . . . . . . . . . . . . . . Replicated
                        . . . . . . . . . . . . . . . Contains LOB Fields
                        . . . . . . . . . . . . . . . . Privileged Use
                        . . . . . . . . . . . . . . . . . LOB File
 File  Name             . . . . . . . . . . . . . . . . . .
-------------------------------------------------------------------------------
    2  BASEFILE         . . I D . . . . . C . . . L . .
    3  LOBFILE          . . I D . . . . . C . . . . . L
   11  UES-EMPLOYEES    . . . D . . . . . . . . . . . .
   13  UES-TEST         . . . D . . . . . . . . . . . .
   14  DBCS3035         . . . D . . . . . . . . . . . .
   15  COLLATION1       . . I D . . . . . P . . . . . .
   18  SECURITY         . . . D . . . . . . . . . . . .
   19  CHECKPOINT       . . . D . . . . . . . . . . . .
   23  EMPL23-EXT       . . . D . . U . . P . . . . . .
   24  SAGABS_MYFILEXX  . . I D . . . . M . . C . . . . .
   50  EMPL50-EXT       . . . D . . U . . . . . . . . .
   88  BASEF-LBLA       . . . D . . . . . . . . . . L . .
   89  LOBF-LBLA        . . . D . . . . . C . . . . . L
  101  file101          . . I D . . . . M . . C . . . . .
  266  UES-EMPLOYEES    . . . D . . . . . . . . . . . .
  267  UES-EMPLOYEES    . . . D . . . . . . . . . . . .
```

Options that are active for a file are indicated by the following codes in the row containing the file name:

Code	Explanation
A	ADAM file. The file was loaded with the ADAM option.
C	Coupling, ciphering, or index compression. The file is coupled to one or more files, and/or the file data is ciphered, and/or the file index is compressed.
D	Space reuse. Space which has been released within a block as a result of a record deletion may be used for a new record.
I	ISN reuse. ISNs of deleted records may be reassigned to new records.
L	If this appears in the "Contains LOB Fields" column, the file contains LB fields (it is a base file). If this appears in the "LOB File" column, the file is a LOB file, not a base file. **Note:** These two columns are mutually exclusive; an "L" will only appear in one of them, if it appears at all.
M	MIXDSDEV active (multiple Data Storage device types) and/or a multiclient file.
N	File is defined with the NOACEXTENSION option.
P	PGMREFRESH is active. "P" is also used in the Priv Use column to indicate that the file is locked by the nucleus for privileged use. The privileged use information is also available in the *File Information* section of the ADAREP report.
R	Replication (Event Replicator for Adabas processing) is active for the file.
S	Spanned record support is activated for the file.
T	Two-byte MU/PE indexes (when MU/PE occurrences exceed 191) are active for the file.
U	File was loaded with the USERISN option.
X	File is a component of an expanded file.

File Space Allocations

The next section shows the space allocated for each file in the database. Here is an example:

```
      **************************
      * FILE SPACE ALLOCATIONS *
      **************************

      FILE    NAME     ALLOC.: NI    UI   AC   DATA/CYL
                       UNUSED:

       1    EMPLOYEES         100     30   03     80/0
       1                       24     17          31/0
       2    VEHICLES           10     20   03     30/0
       2                       03     02          12/0
      10    CHECKPOINT         10     01   01     20/0
      10                       05      0          11/0

↩
```

Here is another example showing a database that uses a LOB file:

```
*************************
* File Space Allocations *
*************************

  File   Name          Alloc.:   NI          UI          AC        Data/Cyl
                       Unused:

     2 BASEFILE                 10           5           1          10/0
     2                          10           5                       9/0
     3 LOBFILE                  10           5           1         100/1
     3                           9           3                      99/1
    11 UES-EMPLOYEES            47          20           1          75/1
    11                           0           4                      53/0
    13 UES-TEST                 30          20           1          10/0
    13                          25          14                       9/0
    14 DBCS3035                 60          22           1          30/0
    14                          22           9                       8/0
    15 COLLATION1                6           3           1           1/0
    15                           4           0                       0/0
    18 SECURITY                  1           1           1           1/0
    18                           1           0                       1/0
    19 CHECKPOINT               30           3           4          90/1
    19                          30           2                      53/0
    23 EMPL23-EXT               60          24           2          30/0
    23                          52          19                      27/0
    24 SAGABS_MYFILEXX          24           4           1           5/0
    24                          24           0                       5/0
    50 EMPL50-EXT               80          49           2         100/1
    50                          25          28                      75/1
    88 BASEF-LBLA               10           5           1          10/0
    88                          10           4                      10/0
    89 LOBF-LBLA                10           5           8         411/5
    89                          10           4                     411/5
   101 file101                   7          14           2          72/0
   101                           6          12                      71/0
   266 UES-EMPLOYEES            30          20           1          75/1
   266                          30          18                      75/1
   267 UES-EMPLOYEES            63          20           1          75/1
   267                          16           4                      53/0
```

When LAYOUT=1 is specified, this section is merged into the Contents of Database section of the report.

Each file listed has two rows in the file space allocations table. The first row shows the number of blocks and cylinders *allocated*. The second row shows the number of blocks and cylinders currently *unused*.

The first two columns give the number and logical name of the file. The remaining columns provide the following information:

Column	The number of . . .
NI	blocks for the normal index.
UI	blocks for the upper index.
AC	blocks for the address converter.
DATA/CYL	blocks and cylinders for Data Storage.

LOB File

If a database includes a LOB file, an additional section describing the LOB file is included in the report. Here is an example:

```
*************
*           *
* LOB Files *                                          yyyy-mm-dd  hh:mm:ss
*           *
*************



File with  I Associated I
LOB fields I LOB file   I
--------------------------
         2 I          3 I
-----------I------------I
        88 I         89 I
-----------I------------I
```

The columns provide the following information:

Column	Lists:
File with LOB fields	The file numbers of files containing LB fields.
Associated LOB file	The number of the LOB file in which the actual LB field values are stored.

Physical Layout of the Database

The next section lists all space allocations for the database in RABN sequence. RABNs allocated to the Associator are listed first, followed by RABNs allocated to Data Storage.

```
**********************************
*                                *
* Physical Layout of the Database *                      yyyy-mm-dd  hh:mm:ss
*                                *
**********************************



      From          To     Number  Dev   Table File VOLSER
       Blk         Blk    of Blks Type   Type       Number

       131     -   162          32 3390    PPT     0 ADA001
       163     -   163           1 3390   DSST     0 ADA001
       164     -   164           1 3390    AC     19 ADA001
       165     -   174          10 3390    UI     19 ADA001
       175     -   224          50 3390    NI     19 ADA001
       225     -   242          18 3390    AC      2 ADA001
       243     -   243           1 3390    AC2     2 ADA001
       244     -   254          11 3390  UNUSED    0 ADA001
       255     -   259           5 3390    AC      1 ADA001
```

Note: Normally, a gap in the physical layout table is accompanied by an error message pointing to the gap. However, this is not the case for the physical layout of a file save. Since the file save contains only the FCBs of the saved files, there will be gaps in the physical layout table and these are reported as 'unknown' ranges.

The columns in this table provide the following information:

Column	Explanation	
FROM BLK	The RABN of the first block in the logical extent.	
TO BLK	The RABN of the last block in the logical extent.	
NUMBER OF BLKS	The number of blocks contained within the extent.	
DEV TYPE	The physical device type.	
TABLE TYPE	The element for which the allocation was made:	
	AC	address converter
	NI	normal index
	UI	upper index
	DS	Data Storage
	DSF	Delta Save logging area
	DSST	Data Storage Space Table
	UNUSED	available space
FILE	The file for which the allocation was made. Zero indicates that the extent is not related to a particular file.	

Column	Explanation
VOLSER NUMBER	The serial number of the volume on which the extent is contained. This is shown for Data Storage only if the Data Storage data sets are present in the JCL.

File Information

General Characteristics

Detailed information on each file in the database is provided after the database information. This information can be limited to certain files or omitted altogether. The first part of this section displays information about the file's characteristics. Here is an example of a file containing spanned records:

```
*********************************
*                               *
* File     1 (BASE-FILE       ) *                          2016-02-25  00:55:03
*                               *
*********************************



TOP-ISN            =       198,972     Highest Index Level =  3
MAX-ISN Expected   =       229,807     Padding Factor ASSO = 50%
Records Loaded     =           100     Padding Factor DATA =  0%
MIN-ISN            =             1     Length of Client NR =  0
TOP AC2 ISN        =            30
MAX AC2 ISN Exp.   =           847
MIN AC2 ISN        =             1
MAX-ISN formatted  =       229,807
MAX-2nd-ISN form.  =           847
Number of Updates  =             0     ISNSIZE             =  4

MAX COMP REC LEN   =           N/A     Date Loaded         = 2009-08-25
BLK/ADD DS  EXT    =            10     Time Loaded         = 00:53:51
BLK/ADD UI  EXT    =             6
BLK/ADD NI  EXT    =             3

ADAM File          No
Ciphered File      No
ISN Reusage        Yes
Space Reusage      Yes
Coupled Files      None
Expanded File      No
USERISN            No
NOACEXTENSION      No
MIXDSDEV           No
PGMREFRESH         No
Multi Client File  No
Privileged usage   No
Online INVERT      None
Index Compressed   Yes
Spanned Rec Supp   Yes
Two Byte MU/PE     Yes
LOB file           No
Contain LOB fields Yes
RPLUPDATEONLY      No
READONLY-MODE      No
Index structure    Advanced
```

Here is an example of the general characteristics of the *base file* of a base file-LOB file pair:

```
*********************************
*                               *
* File     2 (BASEFILE        ) *                          yyyy-mm-dd  hh:mm:ss
*                               *
*********************************



TOP-ISN            =            45     Highest Index Level =  0
MAX-ISN Expected   =         1,377     Padding Factor ASSO = 10%
Records Loaded     =            44     Padding Factor DATA = 10%
MIN-ISN            =             1     Length of Client NR =  0
Number of Updates  =        55,937     ISNSIZE             =  3

MAX COMP REC LEN   =        10,792     Date Loaded         = 2006-09-01
BLK/ADD DS  EXT    =             0     Time Loaded         = 14:16:57
BLK/ADD UI  EXT    =             0     Date of last update = 2007-01-23
BLK/ADD NI  EXT    =             0     Time of last update = 07:00:52

FILE ALPHA CODE    =            37
FILE WIDE CODE     =         4,095
USER WIDE CODE     =         4,095
```

Here is an example of the general characteristics of the *LOB file* of a base file-LOB file pair:

```
*********************************
*                               *
* File     3 (LOBFILE         ) *                          yyyy-mm-dd  hh:mm:ss
*                               *
*********************************



TOP-ISN            =             3     Highest Index Level =  3
MAX-ISN Expected   =         1,377     Padding Factor ASSO = 10%
Records Loaded     =             0     Padding Factor DATA = 10%
MIN-ISN            =             1     Length of Client NR =  0
Number of Updates  =             0     ISNSIZE             =  3

MAX COMP REC LEN   =        10,792     Date Loaded         = 2006-09-01
BLK/ADD DS  EXT    =             0     Time Loaded         = 14:16:58
BLK/ADD UI  EXT    =             0
BLK/ADD NI  EXT    =             0

FILE ALPHA CODE    =    DB DEFAULT
FILE WIDE CODE     =    DB DEFAULT
USER WIDE CODE     =    DB DEFAULT
```

The following information can be provided on this report (although all of these fields may not appear on the sample above):

Field	Explanation
BLK/ADD DS EXT	Maximum number of blocks which may be allocated for each Data Storage secondary extent. See the ADALOD utility, MAXDS parameter.
BLK/ADD NI EXT	Maximum number of blocks which may be allocated for each secondary normal index extent. See the ADALOD utility, MAXNI parameter.
BLK/ADD UI EXT	Maximum number of blocks which may be allocated for each secondary upper index extent. See the ADALOD utility, MAXUI parameter.
Collect before images of updates	Indicates whether or not before images of data storage are collected for replication (if replication is activated) during the update of a record on a file. This information is shown whether or not a file is replicated with a primary key defined.
DATE LOADED	Date the file was loaded.
DATE OF LAST UPDATE	Date the file was last changed.
FILE ALPHA CODE	Current file encoding set for alphanumeric fields in the file. This information is not displayed if UES=NO.
FILE WIDE CODE	Current file encoding set for wide-character fields in the file. This information is not displayed if UES=NO.
HIGHEST INDEX LEVEL	Highest index level currently active for the file.
ISNSIZE	Whether the file contains 3-byte or 4-byte ISNs.
LENGTH OF CLIENT NR	Length of the owner ID for a multiclient file.
MAX AC2 ISN EXP	The highest secondary ISN expected in the file. This statistic is given only if spanned records are activated for the file.
MAX COMP REC LEN	Maximum compressed record length permitted for the file. See the ADALOD utility, MAXRECL parameter. If spanned record support is enabled for the file, the value for this field is shown as "N/A".
MAX-2nd-ISN form.	The highest secondary ISN formatted for the file.
MAX-ISN EXPECTED	Highest ISN planned for the file. See the ADALOD utility, MAXISN parameter.
MAX-ISN FORMATTED	Highest ISN formatted for the file.
MIN AC2 ISN	The lowest secondary ISN in the file. This statistic is given only if spanned records are activated for the file.
MIN-ISN	Lowest ISN that can be assigned to a record in the file. See the ADALOD utility, MINISN parameter.
Number of Updates	Number of updates that have been applied to the file after it was loaded.
PADDING FACTOR ASSO	Associator padding factor. For more information, read about the ADALOD LOAD **ASSOPFAC** parameter or the ADAORD **REORASSO** and **REORFASSO** functions, elsewhere in this guide.

Field	Explanation
PADDING FACTOR DATA	Data Storage padding factor. For more information, read about the ADALOD LOAD **DATAPFAC** parameter or the ADAORD **REORDATA** and **REORFDATA** functions, elsewhere in this guide. **Note:** If spanned records are used, the padding factor is ignored, in an attempt to fully use the block. So it is frequently listed as zero in this report. The padding factor is only used in the last, short, segment of a spanned record.
Primary key of replicated records	Identifies the primary key for replication, if replication is activated. This information is shown only if a file is replicated with a primary key defined.
Records Loaded	Number of records currently contained in the file.
Replicator target ID	Identifies the database ID of the Event Replicator Server used for replication, if replication is activated. This information is shown whether or not a file is replicated with a primary key defined.
TIME LOADED	Time the file was loaded.
TIME OF LAST UPDATE	Time the file was last changed.
TOP AC2 ISN	The highest secondary ISN in use in the file. This statistic is given only if spanned records are activated for the file.
TOP-ISN	Highest ISN currently used in the file.
USER WIDE CODE	Current user encoding set for wide-character fields in the file. This information is not displayed if UES=NO.

Options

File option settings for the file are displayed next. Here is an example showing that spanned records are used:

```
ADAM File          No
Ciphered File      No
ISN Reusage        No
Space Reusage      Yes
Coupled Files      None
Expanded File      No
USERISN            No
NOACEXTENSION      No
MIXDSDEV           No
PGMREFRESH         No
Multi Client File  No
Privileged usage   No
Online INVERT      None
Index Compressed   No
Spanned Rec Supp   Yes
Two Byte MU/PE     No


ADABAS version needed for this file: V71 or later
```

Here is an example of the *base file* of a base file-LOB file pair, showing that the file contains LB fields:

```

ADAM File          No
Ciphered File      No
ISN Reusage        Yes
Space Reusage      Yes
Coupled Files      None
Expanded File      No
USERISN            No
NOACEXTENSION      No
MIXDSDEV           No
PGMREFRESH         No
Multi Client File  No
Privileged usage   No
Online INVERT      None
Index Compressed   Yes
Spanned Rec Supp   No
Two Byte MU/PE     No
LOB file           No
Contain LOB fields Yes
```

Here is an example of the *LOB file* of a base file-LOB file pair, showing that the file is itself a LOB file:

```
ADAM File          No
Ciphered File      No
ISN Reusage        Yes
Space Reusage      Yes
Coupled Files      None
Expanded File      No
USERISN            No
NOACEXTENSION      No
MIXDSDEV           No
PGMREFRESH         No
Multi Client File  No
Privileged usage   No
Online INVERT      None
Index Compressed   Yes
Spanned Rec Supp   No
Two Byte MU/PE     No
LOB file           Yes
Contain LOB fields No
```

Field	Indicates . . .
ADAM File	whether the file was loaded with the ADAM option.
Ciphered File	whether the file was loaded with the cipher option.
ISN Reusage	whether the file ISNs can be reused.
Space Reusage	whether the file Data Storage space can be reused.
Coupled Files	the file(s) to which this file is physically coupled.
Expanded File	whether the file is part of an expanded file; if so, the number of the expanded file is displayed.
USERISN	whether the file was loaded with the USERISN option.
NOACEXTENSION	whether the file permits increasing the MAXISN setting.
MIXDSDEV	whether the file Data Storage extents can be on different device types.
PGMREFRESH	whether the file can be refreshed using the E1 command.
Multiclient File	whether the file can contain records belonging to multiple owners/owner IDs.
Privileged usage	whether the file was locked by the nucleus for privileged usage; if so, only Adabas utilities are allowed to access the file.
Online INVERT	the descriptor(s) being inverted online.
Index Compressed	whether the file index is compressed.
Spanned Rec Supp	whether spanned record support is activated for the file.
Two Byte MU/PE	whether two-byte MU/PE indexes (when MU/PE occurrences exceed 191) are active for the file.
LOB file	whether the file is a LOB file.
Contain LOB fields	whether the file contains one or more LB fields (it is a base file).

Field	Indicates . . .
Index Structure	whether a file has the advanced index structure. "Original" indicates that the index of the file follows the original index structure rules of Adabas. "Advanced" indicates that the index of the file has been updated by a nucleus running with INDEXUPDATE=ADVANCED, and it follows the new, advanced index structure rules.

Delta Save Change Flags

If the Delta Save Facility is installed on the database and delta save logging is enabled, ADAREP shows the delta save change flags for each file:

```
     DELTA SAVE CHANGE FLAGS:
      SAVE ENTIRE INDEX     = [YES | NO]
      SAVE ENTIRE ADDR CONV = [YES | NO]
      SAVE ENTIRE DATA STOR = [YES | NO]
      TOTAL CHANGES BY UTILITIES = nnn BLOCKS

↩
```

Each flag indicates whether all of the index, address converter, or Data Storage, respectively, of the file have been changed by a utility and will be saved entirely in the next delta save operation.

The "TOTAL CHANGE BY UTILITIES" include the blocks within extents that will be saved entirely as well as the blocks changed by ADALOD UPDATE executions.

Space Allocation

The next section lists the space allocations for the file. Here is an example showing space allocations when spanned records are used:

```
List I Dev  Block I   Space Alloc.   I     From        To  I   Unused Space   I
Type I Type Lngth I  Blocks     Cyl  I     RABN       RABN I  Blocks     Cyl  I
     I            I                  I                     I                  I
-----I------------I------------------I---------------------I------------------I
     I            I                  I                     I                  I
 AC  I 3390  2544 I        18       0I       225        242I                  I
 AC2 I 3390  2544 I         1       0I       243        243I                  I
 NI  I 3390  2544 I       500       1I       524       1023I       499       1I
 UI  I 3390  2544 I       100       0I      1024       1123I        98       0I
DSST I 3390  2544 I         1       0I       163        163I                  I
     I            I                  I                     I                  I
 DS  I 3390  5064 I      1500      10I       473       1972I      1397       9I
     I            I                  I                     I                  I
------------------------------------------------------------------------------ ↩
```

Here is an example of the space allocation of a *base file* in a base file-LOB file pair:

```
List I Dev  Block I   Space Alloc.   I     From        To  I   Unused Space   I
Type I Type Lngth I  Blocks     Cyl  I     RABN       RABN I  Blocks     Cyl  I
     I            I                  I                    I                  I
-----I------------I------------------I----------------------I------------------I
     I            I                  I                    I                  I
 AC  I 8391  4136 I         1       0I      1717       1717I                  I
 NI  I 8391  4136 I        10       0I      1988       1997I        10        0I
 UI  I 8391  4136 I         5       0I      1718       1722I         5        0I
 FDT I 8391  4136 I         4       0I       335        338I                  I
DSST I 8391  4136 I         1       0I      1563       1563I                  I
     I            I                  I                    I                  I
 DS  I 8391 10796 I        10       0I       176        185I         9        0I
     I            I                  I                    I                  I
-------------------------------------------------------------------------------
```

Here is an example of the space allocation of a *LOB file* in a base file-LOB file pair:

```
List I Dev  Block I   Space Alloc.   I     From        To  I   Unused Space   I
Type I Type Lngth I  Blocks     Cyl  I     RABN       RABN I  Blocks     Cyl  I
     I            I                  I                    I                  I
-----I------------I------------------I----------------------I------------------I
     I            I                  I                    I                  I
 AC  I 8391  4136 I         1       0I      1998       1998I                  I
 NI  I 8391  4136 I        10       0I      1732       1741I         9        0I
 UI  I 8391  4136 I         5       0I      1999       2003I         3        0I
 FDT I 8391  4136 I         4       0I       339        342I                  I
DSST I 8391  4136 I         1       0I      1563       1563I                  I
     I            I                  I                    I                  I
 DS  I 8391 10796 I       100       1I       403        502I        99        1I
     I            I                  I                    I                  I
-------------------------------------------------------------------------------
```

The space allocations table provides the following information:

Column	Explanation	
LIST TYPE	The database component:	
	AC	address converter
	AC2	secondary address converter extents (for spanned records)
	NI	normal index
	UI	upper index
	DS	Data Storage
	DSF	File-specific delta save logging area
	DSST	Data Storage Space Table

Column	Explanation	
	UNUSED	Available space
DEV TYPE	Physical device containing the component.	
BLOCK LNGTH	Block length depends on the component and device type.	
SPACE ALLOC.	Total number of blocks and cylinders allocated to the component; "0" indicates less than one full cylinder.	
FROM RABN	RABN of the first block in the logical extent.	
TO RABN	RABN of the last block in the logical extent.	
UNUSED SPACE	Number of allocated blocks and cylinders but currently unused; "0" indicates less than one full cylinder.	

Field Definition Table

The Field Definition Table (FDT) is displayed next. This information can be omitted. Here is a general example of the FDT section of the report:

```
       FIELD DESCRIPTION TABLE

             I      I        I         I               I
       LEVEL I NAME I LENGTH I FORMAT I    OPTIONS    I           PARENT OF
             I      I        I         I               I
      ------I------I--------I---------I---------------I-----------------------------I
             I      I        I         I               I                             I
         1   I AA   I    8   I    A    I     DE,UQ     I                             I
         1   I AB   I        I         I               I                             I
         2   I AC   I   20   I    A    I     NU        I                             I
         2   I AE   I   20   I    A    I     DE        I             SUPERDE,PHONDE  I
         2   I AD   I   20   I    A    I     NU        I                             ↩
I
         1   I AF   I    1   I    A    I     FI        I                             I
         1   I AG   I    1   I    A    I     FI        I                             I
         1   I AH   I    6   I    U    I     DE        I                             I
         1   I A2   I        I         I               I                             I
         1   I AO   I    6   I    A    I     DE        I             SUBDE,SUPERDE   I
         1   I AQ   I        I         I     PE        I                             I
         2   I AR   I    3   I    A    I     NU        I             SUPERDE         I
         2   I AS   I    5   I    P    I     NU        I             SUPERDE         I
         1   I A3   I        I         I               I                             I
         2   I AU   I    2   I    U    I               I             SUPERDE         I
         2   I AV   I    2   I    U    I     NU        I             SUPERDE         I

 ↩
```

Here is an example of part of the FDT associated with the *base file* of a base file-LOB file pair, showing the LB fields in the base file.

```
Field Description Table

      I      I        I        I                       I                       I
Level I Name I Length I Format I  Options              I  Parent of            I
      I      I        I        I                       I                       I
------I------I--------I--------I-----------------------I-----------------------I
      I      I        I        I                       I                       I
  1   I  AA  I    6   I    A   I NU                    I                       I
  1   I  AP  I    2   I    P   I NU                    I                       I
  1   I  A1  I    0   I    A   I NU                    I                       I
  1   I  A2  I    0   I    A   I NU,NV                 I                       I
  1   I  A3  I    0   I    A   I MU,NU                 I                       I
  1   I  A4  I    0   I    A   I MU,NU,NV              I                       I
  1   I  A5  I    0   I    A   I NC                    I                       I
  1   I  A6  I    0   I    A   I NC,NV                 I                       I
  1   I  B1  I    0   I    A   I LA,NU                 I                       I
  1   I  B2  I    0   I    A   I LA,NB,NU              I                       I
  1   I  B3  I    0   I    A   I LA,NU,NV              I                       I
  1   I  B4  I    0   I    A   I LA,NB,NU,NV           I                       I
  1   I  B5  I    0   I    A   I LA,MU,NU              I                       I
  1   I  B6  I    0   I    A   I LA,NB,MU,NU           I                       I
  1   I  B7  I    0   I    A   I LA,MU,NU,NV           I                       I
  1   I  B8  I    0   I    A   I LA,NB,MU,NU,NV        I                       I
  1   I  C1  I    0   I    A   I LB,NU                 I                       I
  1   I  C2  I    0   I    A   I LB,NB,NU              I                       I
  1   I  C3  I    0   I    A   I LB,NU,NV              I                       I
  1   I  C4  I    0   I    A   I LB,NB,NU,NV           I                       I
  1   I  C5  I    0   I    A   I LB,MU,NU              I                       I
  1   I  C6  I    0   I    A   I LB,NB,MU,NU           I                       I
  1   I  C7  I    0   I    A   I LB,MU,NU,NV           I                       I
  1   I  C8  I    0   I    A   I LB,NB,MU,NU,NV        I                       I
  1   I  D1  I    0   I    W   I NU                    I                       I
  1   I  D2  I    0   I    W   I NU,NV                 I                       I
  1   I  D3  I    0   I    W   I MU,NU                 I                       I
  1   I  D4  I    0   I    W   I MU,NU,NV              I                       I
  1   I  D5  I    0   I    W   I NC                    I                       I
  1   I  D6  I    0   I    W   I NC,NV                 I                       I
  1   I ↩
```

Here is an example of an FDT report showing **logically deleted** fields (fields W4 and W9 have been logically deleted):

```
Field Description Table

      I      I        I        I                       I                        I
Level I Name I Length I Format I  Options              I  Parent of            I
      I      I        I        I                       I                        I
------I------I--------I--------I-----------------------I------------------------I
      I      I        I        I                       I                        I
  1   I  W1  I    4   I    B   I DT=E(DATE)            I                        I
  1   I  W2  I   10   I    A   I NU                    I                        I
  1   I  W3  I   20   I    A   I NU                    I                        I
  1   I  W4  I   20   I    A   I NU                    I                        I
      I      I        I        IDELETED FIELD         I                        I
  1   I  W5  I   20   I    A   I NU                    I                        I
  1   I  W6  I    2   I    A   I NU                    I                        I
  1   I  W7  I    9   I    U   I NU                    I                        I
  1   I  W8  I    4   I    U   I NU                    I                        I
  1   I  W9  I   12   I    U   I NU                    I                        I
      I      I        I        IDELETED FIELD         I                        I
      I      I        I        I                       I                        I
-------------------------------------------------------------------------------
```

FDT sections are not printed for *LOB files*.

Field	Explanation	
LEVEL	Field level.	
NAME	Field name.	
LENGTH	Field length, in bytes.	
FORMAT	Field's data type:	
	A	alphanumeric
	B	binary
	F	fixed point
	P	packed decimal
	G	floating point
	U	unpacked decimal
	W	wide-character
OPTIONS	DE	Descriptor
	DL	Logically deleted descriptor
	FI	Fixed storage
	LA	Long alphanumeric
	LB	Large object field
	MU	Multiple-value field
	NB	No blank compression
	NC	Null/not counted

Field	Explanation	
	NN	Null not allowed
	NU	Null value suppression
	NV	Not converted (alpha and wide-character fields)
	PE	A periodic group. The fields composing the periodic group are those which follow and have a higher level number.
	UQ	Unique descriptor
	XI	Index (occurrence) number excluded from UQ in PE
PARENT OF	Shows whether this field is a parent field for a collation descriptor, sub/superfield, sub/superdescriptor, hyperdescriptor, or phonetic descriptor.	

Special Descriptors

The next section displays information about any special descriptors (collation descriptors, sub-descriptors, subfields, superdescriptors, superfields, phonetic descriptors, and hyperdescriptors) in the file:

```
SPECIAL DESCRIPTOR TABLE
        I      I        I        I                       I                I
   TYPE I NAME I LENGTH I FORMAT I              OPTIONS I      STRUCTURE I
        I      I        I        I                       I                I
 -------I------I--------I--------I-----------------------I-----------------I
        I      I        I        I                       I                I
  SUPER I   H1 I      4 I      B I                 DE,NU I    AU ( 1 - 2) I
        I      I        I        I                       I    AV ( 1 - 2) I
    SUB I   S1 I      4 I      A I                    DE I    AO ( 1 - 4) I
  SUPER I   S2 I     26 I      A I                    DE I    AO ( 1 - 6) I
        I      I        I        I                       I   AE ( 1 - 20) I
  SUPER I   S3 I     12 I      A I             DE,NU,PE I    AR ( 1 - 3) I
        I      I        I        I                       I    AS ( 1 - 9) I
        I      I        I        I                       I                I
   PHON I   PH I        I        I                       I  PH = PHON(AE) I
        I      I        I        I                       I                I
    COL I   Y1 I     20 I      W I                    DE I       CDX 8,PA I
    COL I   Y2 I     12 I      A I             DE,NU,PE I       CDX 1,AR I
        I      I        I        I                       I                I
        I      I        I        I                       I                I
 ------------------------------------------------------------------------
```

Along with the name, length, and format of each special descriptor, this table provides the following information:

Column	Explanation	
TYPE	SUB	Subfield/subdescriptor
	SUPER	Superfield/superdescriptor
	PHON	Phonetic descriptor
	HYPER	Hyperdescriptor
	COL	Collation descriptor
OPTIONS	DE	Descriptor field
	FI	Fixed point
	LA	Long alphanumeric
	MU	Multiple-value field
	NC	Null not counted (SQL null representation)
	NN	Null not allowed
	NU	Null value suppression
	NV	Not converted (alpha and wide-character fields)
	PE	Periodic group
	UQ	Unique descriptor
	XI	Index (occurrence) number excluded from UQ in PE
STRUCTURE	The component fields and field bytes of the sub-, super-, or hyperdescriptor. Phonetic descriptors show the equivalent alphanumeric elementary fields. Collation descriptors show the associated collation descriptor user exit and the name of the parent field.	

Checkpoint Information

Checkpoint information is also provided if the CPLIST or CPEXLIST parameters are specified:

```
     ********************
     * CHECK-POINT-LIST *                                  yyyy-mm-dd hh:mm:ss
     ********************

     CP        CP        DATE        TIME      PLOG     BLOCK   JOBNAME
     NAME      TYPE                            NR       NR
                    USER TYPE                  VOLSER NR....

     SYNP       30       1995-06-03  14:07:38  47       1        DUAL GAOTB1
                    LOAD                       VOLSER = WRK001
     SYNC       01   ET   1995-06-03  14:08:16  48       2        DUAL GANUC70A
                    SESSION OPEN IGNDIB=N FORCE=N
     SYNP       1C   UTI  1995-06-03  14:08:36  48       3        DUAL GAOTB1
                    RESTRUCT

↩
```

The columns in this table provide the following information:

Column	Explanation
CP-NAME	CP-NAME is the checkpoint identifier. In the case of a user non-synchronized checkpoint, this is the checkpoint identifier supplied by the user program. Checkpoint names starting with "SYN"" are reserved for the Adabas nucleus and utilities, as described in: ■ SYNC -- A synchronized checkpoint made during nucleus initialization, including the status of the ADARUN IGNDIB and FORCE parameters. ■ SYNF -- A checkpoint taken by a user program or utility that requires exclusive (EXF) control of one or more files. ■ SYNP -- A checkpoint from a utility that requires privileged control. Such a utility can perform updating without using the Adabas nucleus. ■ SYNS (mainframe only) -- A checkpoint from Adabas Online System (SYSAOS), ADADBS, or the nucleus. The table below identifies the source. The function identified by this checkpoint is implemented without user intervention during regeneration. Exceptions include a second SYNS 5B recorded at the end of a nucleus session, SYNS 60 recorded at an interval specified by the ADARUN INTNAS parameter, and SYNS 61 recorded when more space is allocated for a file. ■ SYNV (mainframe only) -- Indicates that a volume ID changed during sequential write to a data set is being closed. ■ SYNX -- A checkpoint from a utility requiring exclusive control (EXU) of one or more files. ■ SYN1 (mainframe only) -- A checkpoint made at the beginning of online ADASAV execution (SAVE database function).

Column	Explanation
	■ SYN2 (mainframe only) -- A checkpoint made at the end of online ADASAV execution (SAVE database function). ■ SYN4 (mainframe only) -- A checkpoint made at the beginning of online ADASAV execution (SAVE files operation). ■ SYN5 (mainframe only) -- A checkpoint made at the end of online ADASAV execution (SAVE files operation).
CP TYPE	The checkpoint number. See the following table of checkpoints for the possible checkpoint numbers.
USER TYPE	The Adabas user type that set the checkpoint. The user types are: ■ ET -- An ET user. ■ EXF -- An exclusive-file-control user or utility (privileged user). ■ EXU -- An exclusive-file-update user or utility. ■ UTI -- A utility-update-control utility (privileged user). Exceptions include a second SYNS 5B recorded at the end of a nucleus session, SYNS 60 recorded at an interval specified by the ADARUN INTNAS parameter, and SYNS 61 recorded when more space is allocated for a file. ■ UTS -- An online ADASAV SAVE file (privileged user).
DATE TIME	The date and time the checkpoint was taken.
PLOG NR.	The number of the data protection log in use when the checkpoint was written to the checkpoint file.
BLOCK NR.	The block number of the data protection log in which the checkpoint was written.
VOLSER-NUMBER	The volume serial number of the sequential protection (DDSIBA) log. The volume serial number is "DUAL" if dual logging is used and "MULTI" if multiple logging is used.
JOBNAME	The name of the job that created the checkpoint.

The following table describes the checkpoints written by the Adabas nucleus or utilities:

Type	Name	Originator	Description
01	SYNC	ADANUC	Written by nucleus at start of nucleus session.
01	SYNF	User/Utility	User/utility session OPEN with files used in EXF (exclusive use) mode.
01	SYNS	ADANUC	Written by the starting of a second or subsequent Parallel Services or Cluster Services nucleus.
01	SYNX	EXU user	EXU user open.
02	SYNV	ADANUC	VOLSER entry. Written at volume switch on DDSIBA and at the end of the session if sequential logging is used.
03	SYNF	User/Utility	Close checkpoint for an EXF user.

Type	Name	Originator	Description
03	SYNX	EXU	Close checkpoint for an EXU user.
05	SYN4	ADASAV	SAVE file(s)-start of operation with nucleus active.
05	SYN5	ADASAV	SAVE file(s)-end of operation with nucleus active.
05	SYNP	ADASAV	SAVE file(s)-start of operation
06	SYN1	ADASAV	SAVE database-start of operation with nucleus active.
06	SYN2	ADASAV	SAVE database-end of operation with nucleus active.
06	SYNP	ADASAV	SAVE database-start of operation
07	SYNP	ADASAV	RESTORE file(s)-end of operation
08	SYNP	ADASAV	RESTPLOG-end of operation
09	SYNV	ADASAV	SAVE file(s), VOLSER entry. Written at volume change on DDSAVE and at SAVE-operation end.
0A	SYNV	ADASAV	SAVE database, VOLSER entry. Written at volume switch on DDSAVE and at SAVE-operation end.
0B	SYN1	ADASAV	SAVE DELTA-start of operation with nucleus active
0B	SYN2	ADASAV	SAVE DELTA-end of operation with nucleus active
0B	SYNP	ADASAV	SAVE DELTA-end of operation
0C	SYNP	ADASAV	RESTORE DELTA-end of operation
0D	SYNP	ADASAV	MERGE-end of operation
0E	SYNV	ADASAV	SAVE DELTA, VOLSER entry
0F	SYNV	ADASAV	MERGE, VOLSER entry
10	SYNP	ADAINV	COUPLE files
11	SYNP	ADAINV	INVERT field(s)
15	SYNP	ADAORD	REORDER Associator database
16	SYNP	ADAORD	REORDER Data Storage database
17	SYNP	ADAORD	REORDER database
18	SYNP	ADAORD	REORDER Associator file
19	SYNP	ADAORD	REORDER Data Storage file
1A	SYNP	ADAORD	REORDER file
1B	SYNP	ADAORD	STORE
1C	SYNP	ADAORD	RESTRUCTURE
1D	SYNP	ADADEF	DEFINE NEWWORK
1E	SYNP	ADADEF	MODIFY (default character encodings, timezone)
22	SYNX	ADARES	REGENERATE file
23	SYNX	ADARES	BACKOUT file
24	SYNX	ADARES	REGENERATE all; CPEXLIST lists excluded files
25	SYNX	ADARES	BACKOUT all; CPEXLIST lists excluded files
26	SYNP	ADARES	REPAIR Data Storage

Type	Name	Originator	Description
27	SYNP	ADARES	COPY function sucessfully completed
27	SYNV	ADARES	COPY sequential protection log
28	SYNP	ADARES	PLCOPY function successfully completed
28	SYNV	ADARES	PLCOPY dual or multiple protection log
29	SYNP	ADARES	ADARES CLCOPY function successfully completed.
29	SYNV	ADARES	CLCOPY dual or multiple command log
2A	SYNP	ADARES	PLCOPY MERGE function successfully completed
2A	SYNV	ADARES	PLCOPY MERGE dual or multiple protection log
2B	SYNP	ADARES	CLOG MERGE function successfully completed
2B	SYNV	ADARES	CLOG MERGE dual or multiple command log
30	SYNP	ADALOD	LOAD file
31	SYNP	ADALOD	Mass update
35	SYNX	ADAULD	Unload file
3F	SYNP	ADAZAP	Successful VERIFY - REPLACE
40	SYNS	SYSAOS	Add extent
41	SYNS	SYSAOS	CHANGE default field length
42	SYNS	SYSAOS	DECREASE database size
44	SYNS	SYSAOS	Delete file
45	SYNS	SYSAOS	INCREASE database size
47	SYNS	SYSAOS	RECOVER space
48	SYNS	SYSAOS	Refresh file
49	SYNS	SYSAOS	Remove component file from expanded-file chain
4A	SYNS	SYSAOS	Release descriptor
4B	SYNS	SYSAOS	RENAME file
4C	SYNS	SYSAOS	RENUMBER file
4D	SYNS	SYSAOS	RESET DIB
4E	SYNS	SYSAOS	Reuse ISN
4F	SYNS	SYSAOS	Reuse Data Storage
50	SYNS	SYSAOS	UNCOUPLE files
51	SYNS	SYSAOS	ALLOCATE file extent
52	SYNS	SYSAOS	DEALLOCATE file extent
53	SYNS	SYSAOS	Delete checkpoint
54	SYNS	SYSAOS	Set user priority
55	SYNS	SYSAOS	Modify FCB
57	SYNS	SYSAOS	DEFINE file
58	SYNS	SYSAOS	Write FDT

Type	Name	Originator	Description
59	SYNS	SYSAOS	DEFINE new field
5B	SYNS	ADADBS	Write refreshed statistics (some or all per user request)
5B	SYNS	ADANUC	Write (all) statistics at end of nucleus session
5C	SYNS	SYSAOS	CHANGE default field format
5D	SYNS	SYSAOS	Change file encoding
5E	SYNS	ADADBS	ADADBS REPTOR function (refer to your Event Replicator for Adabas documentation)
60	SYNS	ADANUC	Nucleus statistic checkpoint
61	SYNS	ADANUC	Allocate file space
64	SYNS	ADASCR	Protect files
65	SYNS	ADASCR	Protect fields
66	SYNS	SYSAOS	Link component file into expanded-file chain
67	SYNS	NUCLEUS	Session start with IGNDIB=YES
68	SYNS	SYSAOS	Set USERISN on/off
69	SYNS	SYSAOS	Set MIXDSDEV on/off
6A	SYNS	SYSAOS	Install Delta Save DLOG area
6B	SYNS	SYSAOS	Change Delta Save DLOG area
6C	SYNS	SYSAOS	Remove Delta Save DLOG area
6D	SYNS	ADANUC	NUC-ZAP was used
6E	SYNS	ADADBS	ADADBS REPLICATION function (refer to your Event Replicator for Adabas documentation)
6F	SYNS	SYSAOS	Online process initiated
70	SYNS	SYSAOS	Online invert process
71	SYNS	SYSAOS	Online reorder process
73	SYNC	ADANUC	Nucleus (nuclei) successfully quiesced.
74	SYNC	ADANUC	Nucleus (nuclei) have resumed normal processing.
75	SYNS	ADANUC	Delete heuri-user-entry after Response 72 was detected during nucleus startup.
76	SYNS	ADANUC	Delete heuri-user-entry after Response 72 was detected during nucleus session.
77	SYNS	ADADBS	Enable spanned record support.
78	SYNS	ADADBS	Enable or disable extended MU or PE fields
7A	SYNS	ADADBS	Delete field from the FDT
7B	SYNS	ADANUC	Logically delete a descriptor
7C	SYNS	SYSAOS	Set file status: READONLY mode
7D	SYNS	ADADBS	Add or delete CLOG
7E	SYNS	ADADBS	Add or delete PLOG
7F	SYNS	ADANUC	Change fields
81	SYNS	ADANUC	Modify FCB

Type	Name	Originator	Description
82	SYNS	ADANUC	Online add or increase of an ASSO or DATA data set
83	SYNS	ADANUC	Autoincrease of an ASSO or DATA data set
84	SYNS	SYSAOS	AOSLOG information with user ID
85	SYNS	ADADBS	ADADBS AUDITING function
86	SYNS	ADADBS	ADADBS AUDITSERVER function

149 JCL Requirements and Examples

- Requirements to Run ADAREP JCL 1076

This section describes the job control information required to run ADAREP on the z/OS operating system and shows examples of job streams.

Requirements to Run ADAREP JCL

Data Set	DD Name	Storage	More Information
Associator	DDASSORn	disk	
Data Storage	DDDATARn	disk	
Work Storage	DDWORKR1	disk	Optional, required only when ADARUN parameter ENCRYPTION=YES is in use
Save tape	DDSAVE	tape/ disk	Only with SAVETAPE
Protection log	DDPLOG	tape/ disk	Option with online save tape
ADARUN parameters	DDCARD	reader	*Operations*
ADAREP parameters	DDKARTE	reader	
ADARUN messages	DDPRINT	printer	*Messages and Codes*
Time zone file	TZINFO	disk	Required when database has default time zone
ADAREP messages	DDDRUCK	printer	*Messages and Codes*
ADAREP XML summary report	DDXMLR1	tape/disk	Only necessary when the OUTPUT or ACODE parameters are specified. This data must be defined with DCB parameters RECFM=VB,LRECL=512.
ADAREP XML detailed report	DDXMLR2	tape/disk	Only necessary when the OUTPUT or ACODE parameters are specified in the same run as the CPLIST or CPEXLIST parameters. This data must be defined with DCB parameters RECFM=VB,LRECL=512.

ADAREP JCL Example

All Functions

```
//ADAREP    JOB
//*
//*   ADAREP: ALL FUNCTIONS
//*
//REP      EXEC  PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD      <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDDRUCK   DD   SYSOUT=X
```

```
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADAREP,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADAREP    CPEXLIST
/*
```

Refer to ADAREP in the JOBS data set for this example.

Report from a Save Tape

```
//ADAREPS   JOB
//*
//*   ADAREP: REPORT FROM A SAVE TAPE
//*
//REP      EXEC  PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD      <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <=== WORK
//DDSAVE    DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.SAVE   <=== SAVE DATASET
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADAREP,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADAREP    REPORT SAVETAPE,NOCOUNT
/*
```

Refer to ADAREPS in the JOBS data set for this example.

Report with OUTPUT='XML,UTF8'

```
//ADAREPX   JOB
//*
//*   ADAREP: Report from a save tape, XML in UTF8 Format
//*
//REP      EXEC  PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD     <=== ADABAS LOAD
//          DD   DISP=SHR,DSN=SAG.ICSvrs.L0nn        <=== Code pages
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DBYYYYY.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DBYYYYY.DATAR1 <=== DATA
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DBYYYYY.WORKR1 <=== WORK
//DDSAVE    DD   DISP=SHR,DSN=EXAMPLE.DBYYYYY.SAVE   <=== SAVE DATASET
```

```
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//DDXMLR1   DD   SYSOUT=X,DCB=(LRECL=512,RECFM=VB)
//DDXMLR2   DD   SYSOUT=X,DCB=(LRECL=512,RECFM=VB)
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADAREP,MODE=MULTI,SVC=XXX,DEVICE=DDDD,DBID=YYYYY
/*
//DDKARTE   DD   *
ADAREP    REPORT CPEXLIST,OUTPUT='XML,UTF8'
/*
```

Refer to ADAREPX in the JOBS data set for this example.

XVII ADARES Utility: Database Recovery

This chapter covers the following topics:

Functional Overview
BACKOUT Functions
CLCOPY: Copy Dual Command Log
COPY: Copy a Sequential Protection Log or Save Tape
MERGE CLOG: Merge Nucleus Cluster Command Logs
PLCOPY: Copy Protection Log to Sequential Data Set
REGENERATE: Regenerate Updates
REPAIR: Repair Data Storage Blocks
Multithreaded Processing Statistics
JCL Requirements and Examples

150 Functional Overview

- Using ADARES in Adabas Nucleus Cluster Environments 1083

The ADARES utility performs functions related to database recovery.

1. The functions BACKOUT (except BACKOUT DPLOG or MPLOG), REGENERATE, and REPAIR require a sequential data set containing protection log data as input. A dual/multiple protection log cannot be used directly. To convert dual/multiple protection logs to sequential logs, use the ADARES PLCOPY function.
2. The REGENERATE, BACKOUT, and COPY functions can process several sessions within one job if the following is true:
   - The DDSIIN input file contains the sessions in ascending order by session number; gaps in the session number sequence exist only for those sessions representing save data set generations;
   - Each session on the file begins with block number 1, and there are no gaps in block numbering.
3. To select a *single* session only, specify the log number using the PLOGNUM parameter or specify the session number using only the FROMPLOG parameter (without the TOPLOG parameter); to specify a *range* of sessions, specify values for both the FROMPLOG and TOPLOG parameters.
4. The ADARES COPY function accepts ADASAV output save (DDSAVEn) files. No parameters indicating a save file can be specified; ADARES recognizes a save file by its structure. Only one save file can be copied during an ADARES COPY run. When copying a save file, specify the session number with PLOGNUM.
5. Adabas expanded files: the BACKOUT and REGENERATE (file) functions process complete expanded files only. If the file specified is either the anchor or component file of an expanded file, all other component files of the expanded file must also be specified.
6. To perform the additional Delta Save Facility operations of ADARES, ADARUN parameter DSF=YES must be specified in the DDCARD input.
7. Multithreaded BACKOUT, BACKOUT DPLOG or MPLOG, and REGENERATE require additional main memory, which can be estimated using the values for the corresponding nucleus ADARUN parameter NT and NU:

   ```
   (NT x 70,000) + (NU x 72)
   ```

   For example, if NT=28 and NU=1000, about 2MB of main memory is required.
8. For optimal processing when using the multithreaded backout/regenerate feature, we recommend that you set the nucleus ADARUN parameter NAB to at least

   ```
   NAB=NT x (32K + 108) / 4096
   ```

Using ADARES in Adabas Nucleus Cluster Environments

In an Adabas nucleus cluster environment, the protection logs (and optionally, the command logs) of all individual nuclei in the cluster are merged into single log files in chronological order for the cluster database shared by all the nuclei as a whole. The chronological order is determined by timestamps on all individual nucleus log records, which are synchronized across the cluster by the operating system.

- Merging Logs
- Intermediate Data Sets
- Uniquely Identifying Checkpoints

Merging Logs

For recovery processing, all protection log data sets (PLOGs) must be merged into a single log stream for each cluster database. PLOGs are merged automatically when an ADARES PLCOPY is executed. The PLCOPY process accesses the parallel participant table (PPT) to determine which PLOGs to copy and uses dynamic allocation to access the appropriate data sets.

An existing PLCOPY job must be modified to run in a cluster environment. The user exit 2 may also need to be modified. A sample PLCOPY job ADARESMP that illustrates the necessary addition of the intermediate data sets and a sample user exit 2 (USEREX2P) is provided. See *Automatically Copy/Merge Nucleus Cluster Protection Logs*. It is not necessary to remove the PLOG DD statements, however. If they remain, they are ignored.

Caution: Sample user exits and programs and are not supported under any maintenance contract agreement.

By default, dual/multiple command log data sets (CLOGs) can be copied to a sequential data set for each nucleus using the ADARES CLCOPY function, but the resulting data sets are not then automatically merged across the cluster into a single CLOG data set for the cluster database. You can choose to merge the CLCOPY output from each nucleus manually by using the ADARES MERGE CLOG function. By default, the CLOG data sets must be specified in the user exit 2 JCL; they are not dynamically allocated.

However, for accounting or other tracking purposes, you may want to automate the CLOG merge process the same way the PLOG merge process is automated. When you specify ADARUN CLOGMRG=YES, the CLOG merge process is invoked automatically when the ADARES CLCOPY job is submitted from UEX2 and executed. ADARUN LOGGING=YES must also be specified. As with the PLCOPY process, the CLCOPY process then accesses the parallel participant table (PPT) to determine which CLOGs to copy and uses dynamic allocation to access the appropriate data sets.

Existing CLCOPY jobs must be modified to include the intermediate data sets. A sample CLCOPY job ADARESMC is provided that illustrates the necessary addition of the intermediate data sets. See **Automatically Copy/Merge Nucleus Cluster Command Logs**. The sample user exit 2 (USEREX2P) includes both CLCOPY and PLCOPY functionality for the merge.

The automated PLCOPY and CLCOPY jobs copy/merge as much data as possible; if a nucleus is still writing to a log data set, the job 'partially' merges the data set.

Intermediate Data Sets

The merge begins with the lowest timestamp from all PLOGs and CLOGs being merged and ends with the lowest of the ending timestamps from all data sets. Records beyond this point are written to an 'intermediate' data set, which must be supplied as input to the subsequent merge. A cross-check ensures that the correct intermediate data set has been supplied.

ADARES expects that at least one of the PLOGs or CLOGs being merged is at 'completed' status. If this is not the case, ADARES reports that there is no data to be copied.

A sample user exit 2 (USEREX2P for both PLOGs and CLOGs) is provided that illustrates the necessary JCL for the intermediate data sets. When intermediate data sets are used for both CLCOPY and PLCOPY jobs, the data set names for each must be unique so that they are not overwritten.

- PLCOPY example:

```
//MERGIN1 DD DISP=SHR,DSN=EXAMPLE.PINTERI
//MERGIN2 DD DISP=SHR,DSN=EXAMPLE.PINTERO
```

- CLCOPY example:

```
//MERGIN1 DD DISP=SHR,DSN=EXAMPLE.CINTERI
//MERGIN2 DD DISP=SHR,DSN=EXAMPLE.CINTERO
```

Depending on whether it is a PLCOPY or a CLCOPY, the job submitted by user exit 2 must refer to the appropriate set of statements.

Once DD statements for the PLOG data sets have been supplied on the session startup JCL, you do not need to supply them again for ADARES as these are opened using dynamic allocation. If the DD statements are supplied, they are ignored.

It is not necessary to manually change the JCL after each execution. ADARES maintains control information in the parallel participant table (PPT) to determine which intermediate data set to expect as input. It checks the header information in both data sets to determine which to use for input and which for output.

The following checks are made to ensure that the intermediate data set has been supplied correctly:

1. The DBID is stored in the intermediate data set header and must match the DBID in the log.
2. The log number is stored in the intermediate data set header and must either match or be one less than the current number from the log data set.
3. The STCK in the intermediate data set header must match the STCK stored in the PPT.

If any of the checks fails, ADARES ERROR 157 is returned.

ADARES also ensures that the intermediate data set contains the number of records expected. If not, ADARES ERROR 164 is returned.

Uniquely Identifying Checkpoints

After the protection log (PLOG) merge process, the block number will not necessarily be the same. To uniquely identify the checkpoint in this situation, it is necessary to also specify the NUCID for all functions that can specify a TOBLK/ FROMBLK parameter; that is, BACKOUT and REGENERATE.

The merge process ensures that there is at most one checkpoint per block. It records the (old) block number prior to the merge and the NUCID that wrote the checkpoint. When you then specify the block number and NUCID as reported in ADAREP, ADARES is able to uniquely identify the block.



Note: In an Adabas nucleus cluster environment, ADAREP includes the NUCID when printing all checkpoint information.

The additional parameters that are required in an Adabas nucleus cluster environment are NUCID, TONUCID, and FROMNUCID. If the NUCID is the same for the starting and ending checkpoint, only the NUCID needs to be specified.



Note: ADASAV stores this information in the header so that it can uniquely identify the block for the RESTONL and RESTPLOG functions.

151 BACKOUT Functions

Data protection information in the form of *before* and *after* images of all updated records is written to the protection log during each Adabas session. This information is needed to remove or reapply updates.

The protection log may be assigned to a sequential data set or to a dual/multiple protection log data set (direct access) on disk. If the dual/multiple protection log is used, the ADARES PLCOPY function must be used to copy it to a sequential data set. This data set can be used as input to ADARES BACKOUT or REGENERATE.

We do not recommend the use of 3480/3490 tape cartridge compression (IDRC) for protection log files. The ADARES utility BACKOUT function runs at least twice as long under z/OS when processing compressed data.

The ADARES BACKOUT {DPLOG | MPLOG} function is not valid for a cluster database. This is not allowed because a merged PLOG is required in order to perform the BACKOUT.

BACKOUT	Back Out Updates Using the Sequential Protection Log (SIBA)
BACKOUT DPLOG or MPLOG	Back Out Updates Using the Dual or Multiple Protection Log

BACKOUT Function Statistics

File processing statistics from ADARES BACKOUT function processing are provided at the end of the run. These statistics include the number of data storage records backed out for each file as well as information about the PLOG blocks read and sent, the commands and transactions processed, the number of Adabas calls processed (including maximum and average calls processed in parallel), and the average record buffer size. The statistics are gathered via 4-byte counters that keep count of the total data storage updates for each file during backout processing.

Spanned records and records with large object (LB) fields can span more than one data storage block. Thus an update to a record of this type will increment the data record update counter by one for each data storage block. For example, suppose a spanned record is updated that is stored

across three data storage blocks. In this case, the record update counter is incremented three times, one for each data storage block.

Data storage records that were modified in a transaction that did not successfully terminate with an ET command are counted, even though they are backed out by the nucleus.

The following is a sample of an ADARES BACKOUT report (when MTR=YES):

Note: The number of records listed as "Data Record Updates" is the number of data storage records processed; the number of records listed as "PLOG Records sent to ADABAS" is the number of input records from the protection log (PLOG). The PLOG record count is usually much larger than the data record update count.

```
(BACKOUT) File Processing Statistics
--------------------------------------------
I   File Number   I   Data Record Updates   I
I-----------------I-------------------------I
I        10       I               127       I
--------------------------------------------


Multi-Threading Processing Statistic
------------------------------------

PLOG Blocks Read from Input           17
PLOG Records Sent to ADABAS          405
Commands Processed                   158
Transactions Processed                28
Number of ADABAS Calls                28
Maximum Calls in Parallel             23
Average Calls in Parallel             10
Average Record Buffer Size          1574
```

The following is a sample of an ADARES BACKOUT report when MTR=NO:

```
(BACKOUT) File Processing Statistics

--------------------------------------------
I   File Number   I   Data Record Updates   I
I-----------------I-------------------------I
I        10       I               127       I
--------------------------------------------


ADARES (BACKOUT) Normal end: 17 blocks / 405 records processed.
```

152 BACKOUT: Back Out Updates Using the Sequential Protection Log (SIBA)

- Essential Parameters 1091
- Optional Parameters and Subparameters 1092
- Examples 1096

The BACKOUT function removes all the updates applied between two specified checkpoints. Both checkpoints must be contained on the sequential protection log input data set.

The BACKOUT function requires that the read backward feature is supported by the tape drive to be used for sequential input.



Note: An interrupted BACKOUT run must be reexecuted from the beginning.

You can specify either the log number (PLOGNUM) or the session number (FROMPLOG) of the protection log as a starting point for BACKOUT processing. If you specify a session number, you can also specify a range of sessions to be processed using the TOPLOG parameter.

By default, ADARES processes the database specified by the ADARUN DBID parameter. If BACKOUT processing is required against a different database, use the PLOGDBID parameter to specify the database.

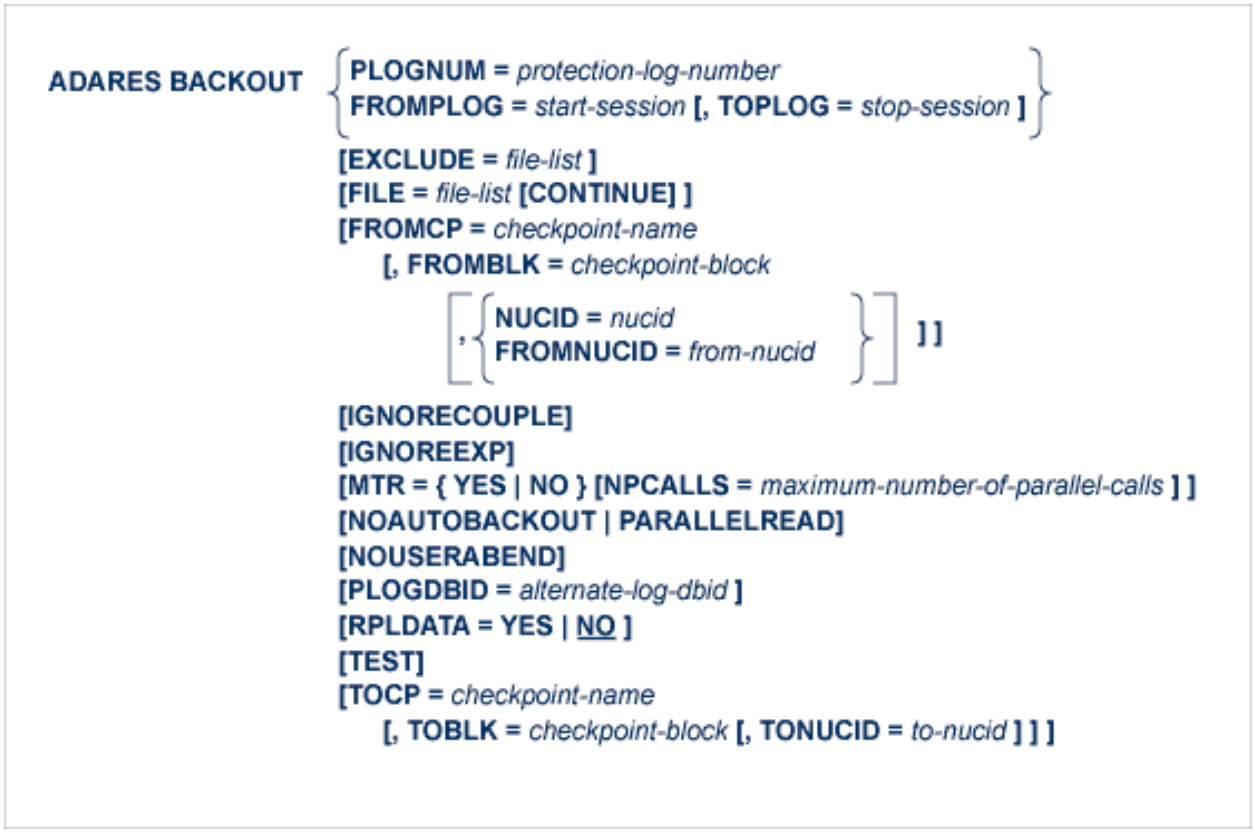
By default, BACKOUT processing continues until the end of the input data set is reached. You can limit the extent of BACKOUT processing using the TOCP parameter.

By default, all files in the specified input data set are included in the BACKOUT processing. You have the option to identify specific files to be included.

At the end of BACKOUT processing, ADARES automatically backs out all incomplete logical transactions when BACKOUT is specified for the entire database and continues until the end of the input data set is reached. This also occurs if

- the FILE parameter and the CONTINUE parameter are both specified; and
- the TOCP parameter is not specified.

You can override this process by specifying the NOAUTOBACKOUT parameter.



Essential Parameters

PLOGNUM: Protection Log Number

PLOGNUM specifies the log number of the sequential protection log to be used as input for BACKOUT processing. The log number may be obtained from the database status report.

FROMPLOG: Starting Session for BACKOUT

FROMPLOG specifies the session number at which BACKOUT processing is to start. ADARES searches the sequential PLOG input (DDSIIN) file for the correct starting session. To define the starting point more precisely, specify the FROMCP and FROMBLK parameters.

Optional Parameters and Subparameters

CONTINUE: Continue File Recovery with Autobackout

When FILE is specified, CONTINUE locks the complete database for exclusive use by the BACKOUT function.

It allows autobackout of incomplete transaction changes, if any, during file backout. If specified, all changes made by incomplete transactions are backed out of the database data sets specified by the FILE parameter.

If the file list contains coupled or expanded component files and CONTINUE is specified, the usual default checking of the list for all coupled and/or remaining component files does not occur; in this case, IGNORECOUPLE or IGNOREEXP does not have to be specified to stop the checking.

EXCLUDE: Exclude Specified Files from Backout

EXCLUDE lists the numbers of the files to be excluded from BACKOUT processing; that is, the files that are not to be backed out. Any protection records that pertain to these files are ignored.

The parameter is optional: if not specified, no files are excluded. A file number may be listed only once.

When the FILE parameter is specified, all files specified in the EXCLUDE parameter must also be specified in the FILE parameter.

The EXCLUDE parameter has no bearing on whether the BACKOUT is performed with or without transaction logic.

The EXCLUDE parameter is provided for use in recovery jobs built by the Adabas Recovery Aid (ADARAI).

Excluded files are listed in the extended CPLIST of the ADAREP report.

FILE: Files to Be Included

If all files are to be included in the BACKOUT processing, this parameter should not be specified.

If the specified file is a component file of an Adabas expanded file, then all other component files for the expanded file must also be specified. If a specified file is coupled to other files, these files must also be specified.

FROMBLK: Beginning Block for BACKOUT

FROMBLK specifies the block number containing the FROMCP checkpoint entry. This block number, which may be obtained from the database status report, refers to either PLOGNUM or FROMPLOG. FROMBLK can only be specified if FROMCP is specified.

FROMCP: Beginning Checkpoint for BACKOUT

FROMCP specifies the checkpoint before which the backout process is to begin. The checkpoint identification (name), which may be obtained from the database status report, refers to either PLOGNUM or FROMPLOG.

If backout processing is to begin at the end of the log, this parameter should be omitted.

FROMNUCID: Starting Nucleus ID

In a cluster environment, the NUCID parameter or a combination of the FROMNUCID and TONUCID parameters may be required to identify the nuclei associated with the checkpoints referenced in this backout run. If the starting checkpoint block (FROMBLK parameter) and ending checkpoint block (TOBLK parameter) are for the same nucleus, use the NUCID parameter; if the starting checkpoint block (FROMBLK) and ending checkpoint block (TOCP) are for different nuclei, use the FROMNUCID and TONUCID parameters.

The FROMNUCID parameter specifies the nucleus ID for the starting nucleus. If you specify the FROMNUCID parameter, a corresponding TONUCID parameter is expected.

IGNORECOUPLE: Ignore Unspecified Couple Files

IGNORECOUPLE (or CONTINUE) stops the BACKOUT function from checking the FILE list for complete coupled file pairs. If neither CONTINUE nor IGNORECOUPLE are specified and the FILE list specifies a coupled file without specifying its mate, ADARES terminates and issues an error message.

IGNOREEXP: Ignore Expanded Component Files

If the FILE list includes Adabas expanded component files, ADARES BACKOUT normally checks to ensure that all additional component files *related to the listed component files* are also in the list; if not, ADARES ends the BACKOUT operation and issues an error message. Specifying IGNOREEXP (or CONTINUE) stops the checking for related component files.

MTR: Multithreaded Regenerate Switch

MTR=YES activates the multithreaded regenerate feature; MTR=NO disables it.

When the multithreaded regenerate feature is active, multiple buffers containing PLOG information are sent to the Adabas nucleus in parallel to improve performance. When the feature is not active, only one buffer is sent to Adabas at a time.

If the nucleus ADARUN parameter MODE=SINGLE, MTR is automatically set to NO. Multiple threads are not available to Adabas running in single user mode.

If the FILE parameter is not specified, or is specified with CONTINUE, the default value for MTR is YES. In these cases, multithreaded regenerate has exclusive control of the whole database and is generally effective.

Otherwise, the default value is NO. If it only has exclusive control of some files, as is the case when FILE is specified *without* CONTINUE, multithreaded regenerate can run in parallel with normal applications accessing different files and has the potential to negatively impact the performance of production applications.

NOAUTOBACKOUT: Prevent Incomplete Transaction Backout

If several consecutive BACKOUT runs are necessary in order to process multiple protection logs resulting from a single Adabas session, an automatic backout should be performed only for the last input log. The NOAUTOBACKOUT parameter should therefore be specified for each BACKOUT run except the run in which the last input log is used.



Notes:

1. NOAUTOBACKOUT cannot be specified in single-user mode.
2. NOAUTOBACKOUT is mutually exclusive with PARALLELREAD; only one of these parameters may be specified in an ADARES BACKOUT run.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

NPCALLS: Maximum Number of Parallel Calls

When MTR=YES, the NPCALLS parameter may be specified to limit the number of parallel calls sent to the Adabas nucleus.

If the FILE parameter is not specified, or is specified with CONTINUE, the default value for NPCALLS is the nucleus ADARUN parameter NT+1 or NC, whichever is smaller.

If the FILE parameter is specified without CONTINUE, the default value is the nucleus ADARUN parameter NT+1 or NC/2, whichever is smaller.

NPCALLS is primarily used to reduce the number of parallel calls allowed by the default value. Fewer parallel calls mean a smaller nucleus workload produced by ADARES. This is especially useful for increasing the resources available to application programs running in parallel with BACKOUT FILE.

NUCID: Nucleus ID

In a cluster environment, the NUCID parameter or a combination of the FROMNUCID and TONUCID parameters may be required to identify the nuclei associated with the checkpoints referenced in this backout run. If the starting checkpoint block (FROMBLK parameter) and ending checkpoint block (TOBLK parameter) are for the same nucleus, use the NUCID parameter; if the starting checkpoint block (FROMBLK) and ending checkpoint block (TOCP) are for different nuclei, use the FROMNUCID and TONUCID parameters.

PARALLELREAD: Enable Read-Only File Usage for Other Users

The PARALLELREAD parameter provides for concurrent read-only access to the files being processed by ADARES BACKOUT both for database-wide and file-oriented functions:

- for file-oriented functions, specifying PARALLELREAD causes ADARES to issue an OPEN call with "EXU=file-list" in the record buffer. This allows read-only access to the files for other users while ADARES is active.
- when FILE is not specified or when CONTINUE is specified, the PARALLELREAD parameter is effective for a database-wide session backout. The parameter makes it possible for read-only users to access the database at the same time the database session is being backed out.

Update commands are rejected.

If parallel access users read records that were updated in the database session being backed out, they may see record images that are logically wrong in the sense of the application, or response codes such as 113 (ADARSP113) that indicate inconsistencies.



Notes:

1. During ADARES operation with PARALLELREAD, temporary differences between the Associator and Data Storage may cause nucleus responses 113 or 199 to occur.
2. NOAUTOBACKOUT is mutually exclusive with PARALLELREAD; only one of these parameters may be specified in an ADARES BACKOUT run.

PLOGDBID: Alternate Protection Log ID

When performing a backout operation using a protection log from a database other than that specified by the ADARUN statement's DBID parameter, PLOGDBID specifies the database ID of the alternate protection log. The default is the database ID from the ADARUN-specified database.

RPLDATA: Replicate protection log data

The RPLDATA parameter allows you to specify whether or not the nucleus should replicate the protection log data sent to it. Valid values are "YES" and "NO"; the default is "NO".

ADARES BACKOUT with RPLDATA=YES will not replicate data for a file with spanned DS record support.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

TOBLK: Ending TOCP Block

TOBLK specifies the block number containing the TOCP checkpoint entry. This block number, which can be obtained from the database status report, refers to either PLOGNUM or FROMPLOG, or to TOPLOG, if specified. TOBLK can only be specified if TOCP is specified.

TOCP: Ending Checkpoint Block for Backout

TOCP specifies the checkpoint at which the backout process is to be terminated. If backout processing is to continue until the beginning of the log, do not specify TOCP. The checkpoint identification (name), which can be obtained from the database status report, refers to either TOPLOG, if specified, or to PLOGNUM or FROMPLOG.

TONUCID: Ending Nucleus ID

In a cluster environment, the NUCID parameter or a combination of the FROMNUCID and TONUCID parameters may be required to identify the nuclei associated with the checkpoints referenced in this backout run. If the starting checkpoint block (FROMBLK parameter) and ending checkpoint block (TOBLK parameter) are for the same nucleus, use the NUCID parameter; if the starting checkpoint block (FROMBLK) and ending checkpoint block (TOCP) are for different nuclei, use the FROMNUCID and TONUCID parameters.

The TONUCID parameter specifies the nucleus ID for the ending nucleus. Only specify a TONUCID parameter if a corresponding FROMNUCID parameter is also specified.

TOPLOG: Ending PLOG Session for Backout

TOPLOG specifies the last session to be processed by the specified ADARES function. If ADARES finds a session on the PLOG input (DDSIIN) file whose session number is outside the inclusive range defined by FROMPLOG/TOPLOG, that session is excluded from ADARES processing. TOPLOG can only be specified if FROMPLOG is also specified. If TOPLOG is not specified, the FROMPLOG session becomes the default. To define the ending point more precisely, specify the TOCP and TOBLK parameters.

Examples

Example 1:

```
ADARES BACKOUT PLOGNUM=3
```

All files are to be included in backout processing. The protection log number is 3. Backout processing is to begin at the end of the log and is to end at the beginning of the log. At the end of the backout processing, an automatic backout (but moving forward) of incomplete transactions occurs.

Example 2:

```
ADARES BACKOUT
FILE=4,7,PLOGNUM=11,FROMCP=CH18,FROMBLK=1864,
ADARES                TOCP=CH01,TOBLK=1
```

The backout is to be limited to files 4 and 7. All updates applied to files 4 and 7 between the taking of checkpoints CH01 and CH18 are to be removed. CH01 is located in block 1 of data protection log 11. Checkpoint CH18 is located in block 1864 of data protection log 11. No automatic backout of incomplete transactions occurs.

153 BACKOUT DPLOG or MPLOG: Back Out Updates Using the Dual or Multiple Protection Log

- Executing the Function 1098
- Syntax 1099
- Essential Parameter 1100
- Optional Parameters 1100
- Example 1105

The BACKOUT {DPLOG | MPLOG} function removes all the updates applied between two checkpoints contained on the same Adabas dual or multiple protection log data set, respectively.

The BACKOUT {DPLOG | MPLOG} function is not valid for a cluster database. This is disallowed because a merged PLOG is required in order to perform the BACKOUT.

Executing the Function

The following sequence is recommended for executing the BACKOUT DPLOG or MPLOG function:

1. Issue the operator or Online System command FEOFPL.

   Force EOF on the current protection log data set and switch to a new one. The new protection log data set will contain all information required for BACKOUT DPLOG or MPLOG.
2. Run the user application.

   All protection log data written by the nucleus for this application must fit on a single protection log data set. No protection log switch may occur while the application is running. Here, you should assume that the application program has failed, and must be backed out.
3. Issue again the operator or Online System command FEOFPL.

   Close the protection log data set. The closed data set contains all information required for BACKOUT DPLOG or MPLOG.
4. Run ADARES PLCOPY.

   Copy the content of the protection log data set to a sequential data set. This can be done by running ADARES PLCOPY or by using user exit 2 for DPLOG or user exit 12 for MPLOG.
5. Run ADARES BACKOUT DPLOG or MPLOG.

   This backs out the session to the status of step 1.

   An interrupted BACKOUT DPLOG or MPLOG run must be reexecuted from the beginning. If the data on the protection log data set that is to be used is unavailable (the nucleus uses this data protection log again), a BACKOUT from the sequential copy must be done.

Syntax

In general, the parameters FROMCP/TOCP/FROMBLK/TOBLK should not be specified. We recommend that you back out using the entire contents of one protection log data set.

During backout, the nucleus writes new protection log information to the protection log data set currently available. This is the only data set that can be used by the nucleus. In case of a protection log switch during BACKOUT DPLOG or MPLOG, the nucleus waits until the complete data set has been copied with ADARES PLCOPY, and then resumes the backout run.

By default, all files in the specified input data set are included in the BACKOUT processing. You have the option to identify specific files to be included.

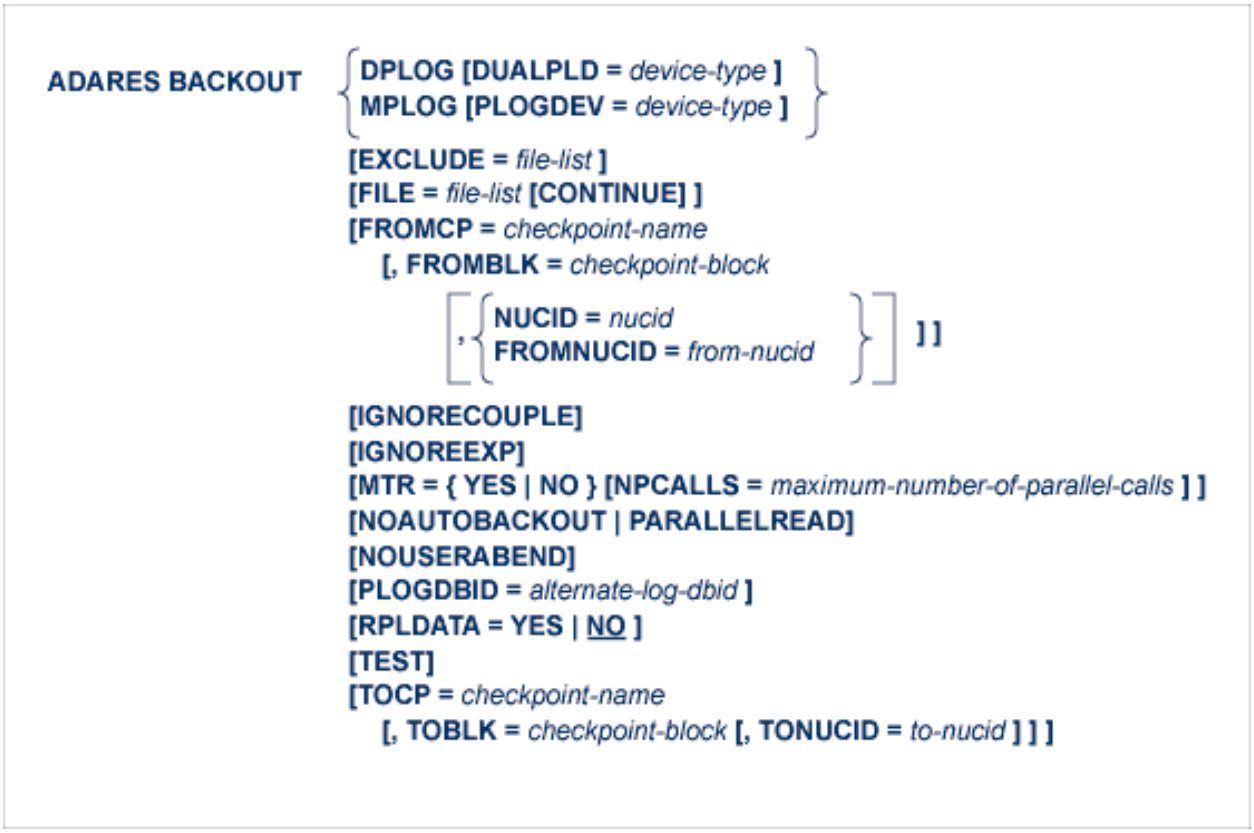
CONTINUE allows autobackout of incomplete transaction changes, if any, during file backout. If specified, all changes made by incomplete transactions are backed out of the database data sets specified by the FILE parameter. If the file list contains coupled or expanded component files and CONTINUE is specified, the usual default checking of the list for all coupled and/or remaining component files does not occur; in this case, IGNORECOUPLE or IGNOREEXP does not have to be specified to stop the checking.

Specifying CONTINUE locks the complete database for exclusive use by the BACKOUT function during file backout.

At the end of BACKOUT processing, ADARES automatically backs out all incomplete logical transactions when BACKOUT is specified for the entire database and continues until the end of the input data set is reached. This also occurs if

- the FILE parameter and the CONTINUE parameter are both specified; and
- the TOCP parameter is not specified.

You can override this process by specifying the NOAUTOBACKOUT parameter.



Essential Parameter

DPLOG | MPLOG: Dual or Multiple PLOG Source

DPLOG indicates that a dual protection log data set is to be used as input; MPLOG indicates that a multiple protection log data set is to be used as input.

Optional Parameters

CONTINUE: Continue File Recovery with Autobackout

When FILE is specified, CONTINUE locks the complete database for exclusive use by the BACKOUT function.

It allows autobackout of incomplete transaction changes, if any, during file backout. If specified, all changes made by incomplete transactions are backed out of the database data sets specified by the FILE parameter.

If the file list contains coupled or expanded component files and CONTINUE is specified, the usual default checking of the list for all coupled and/or remaining component files does not

occur; in this case, IGNORECOUPLE or IGNOREEXP does not have to be specified to stop the checking.

DUALPLD | PLOGDEV: PLOG Device Type

DUALPLD specifies the device type used for the dual protection log data sets; PLOGDEV specified the device type used for the multiple protection log data sets. The default is the device type specified by the ADARUN DEVICE parameter.

EXCLUDE: Exclude Specified Files from Backout

EXCLUDE lists the numbers of the files to be excluded from BACKOUT processing; that is, the files that are not to be backed out. Any protection records that pertain to these files are ignored.

The parameter is optional: if not specified, no files are excluded. A file number may be listed only once.

When the FILE parameter is specified, all files specified in the EXCLUDE parameter must also be specified in the FILE parameter.

The EXCLUDE parameter has no bearing on whether the BACKOUT is performed with or without transaction logic.

The EXCLUDE parameter is provided for use in recovery jobs built by the Adabas Recovery Aid (ADARAI).

Excluded files are listed in the extended CPLIST of the ADAREP report.

FILE: Files to be Included

FILE specifies the files to be included in the backout process. If all files are to be included, this parameter should be omitted. If the specified file is a component file of an Adabas expanded file, all other component files of the expanded file must also be specified here. If a specified file is coupled to other files, the coupled files must also be specified.



Note: Before beginning, ADARES locks all specified files for the duration of BACKOUT execution. If the FILE parameter is omitted, the entire database will be locked. Other users can have read-only access to the specified files if the UTYPE=EXU parameter is specified or to the database if the PARALLELREAD parameter is specified.

FROMBLK: Beginning FROMCP Block Number

FROMBLK is the block number containing the FROMCP checkpoint entry. This block number may be obtained from the database status report. FROMBLK can be specified only if FROMCP is specified.

FROMCP: Beginning Checkpoint for Backout

FROMCP specifies the checkpoint before which the backout process is to begin. The checkpoint ID may be obtained from the database status report. If backout processing is to begin at the end of the log, do not specify the FROMCP parameter.

FROMNUCID: Starting Nucleus ID

In a cluster environment, the NUCID parameter or a combination of the FROMNUCID and TONUCID parameters may be required to identify the nuclei associated with the checkpoints referenced in this backout run. If the starting checkpoint block (FROMBLK parameter) and ending checkpoint block (TOBLK parameter) are for the same nucleus, use the NUCID parameter; if the starting checkpoint block (FROMBLK) and ending checkpoint block (TOCP) are for different nuclei, use the FROMNUCID and TONUCID parameters.

The FROMNUCID parameter specifies the nucleus ID for the starting nucleus. If you specify the FROMNUCID parameter, a corresponding TONUCID parameter is expected.

IGNORECOUPLE: Ignore Unspecified Coupled Files

IGNORECOUPLE (or CONTINUE) stops the BACKOUT function from checking the FILE list for complete coupled file pairs. If neither CONTINUE nor IGNORECOUPLE are specified and the FILE list specifies a coupled file without specifying its mate, ADARES terminates and issues an error message.

IGNOREEXP: Ignore Expanded Component Files

If the FILE list includes any Adabas expanded component files, ADARES BACKOUT normally checks to ensure that all additional component files *related to the listed component files* are also in the list; if not, ADARES ends the BACKOUT operation and issues an error message. Specifying IGNOREEXP (or CONTINUE) stops the checking for related component files.

MTR: Multithreaded Backout Switch

MTR=YES activates the multithreaded backout feature; MTR=NO disables it.

When the multithreaded backout feature is active, multiple buffers containing PLOG information are sent to the Adabas nucleus in parallel to improve performance. When the feature is not active, only one buffer is sent to Adabas at a time.

If the nucleus ADARUN parameter MODE=SINGLE, MTR is automatically set to NO. Multiple threads are not available to Adabas running in single user mode.

If the FILE parameter is not specified, or is specified with CONTINUE, the default value for MTR is YES. In these cases, multithreaded backout has exclusive control of the whole database and is generally effective.

Otherwise, the default value is NO. If it only has exclusive control of some files, as is the case when FILE is specified *without* CONTINUE, multithreaded backout can run in parallel with normal applications accessing different files and has the potential to negatively impact the performance of production applications.

NOAUTOBACKOUT: Prevent Incomplete Transaction Backout

If several consecutive BACKOUT runs are necessary in order to process multiple protection logs resulting from a single Adabas session, an automatic backout should be performed only for the last input log. The NOAUTOBACKOUT parameter should therefore be specified for each BACKOUT run except the run in which the last input log is used.

Notes:

1. NOAUTOBACKOUT cannot be specified in single-user mode.
2. NOAUTOBACKOUT is mutually exclusive with PARALLELREAD; only one of these parameters may be specified in an ADARES BACKOUT run.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

NPCALLS: Maximum Number of Parallel Calls

When MTR=YES, the NPCALLS parameter may be specified to limit the number of parallel calls sent to the Adabas nucleus.

If the FILE parameter is not specified, or is specified with CONTINUE, the default value for NPCALLS is the nucleus ADARUN parameter NT+1 or NC, whichever is smaller.

If the FILE parameter is specified without CONTINUE, the default value is the nucleus ADARUN parameter NT+1 or NC/2, whichever is smaller.

NPCALLS is primarily used to reduce the number of parallel calls allowed by the default value. Fewer parallel calls mean a smaller nucleus workload produced by ADARES. This is especially useful for increasing the resources available to application programs running in parallel with BACKOUT DPLOG FILE.

NUCID: Nucleus ID

In a cluster environment, the NUCID parameter or a combination of the FROMNUCID and TONUCID parameters may be required to identify the nuclei associated with the checkpoints referenced in this backout run. If the starting checkpoint block (FROMBLK parameter) and ending checkpoint block (TOBLK parameter) are for the same nucleus, use the NUCID parameter; if the starting checkpoint block (FROMBLK) and ending checkpoint block (TOCP) are for different nuclei, use the FROMNUCID and TONUCID parameters.

PARALLELREAD: Enable Read-Only File Usage by Other Users

The PARALLELREAD parameter provides for concurrent read-only access to the files being processed by ADARES BACKOUT DPLOG both for database-wide and file-oriented functions:

- for file-oriented functions, specifying PARALLELREAD causes ADARES to issue an OPEN call with "EXU=file-list" in the record buffer. This allows read-only access to the files for other users while ADARES is active.

- when FILE is not specified or when CONTINUE is specified, the PARALLELREAD parameter is effective for a database-wide DPLOG backout. The parameter makes it possible for read-only users to access the database at the same time the database DPLOG is being backed out.

Update commands are rejected.

If parallel access users read records that were updated in the database DPLOG being backed out, they may see record images that are logically wrong in the sense of the application, or response codes such as 113 (ADARSP113) that indicate inconsistencies.



Notes:

1. During ADARES operation with PARALLELREAD, temporary differences between the Associator and Data Storage may cause nucleus responses 113 or 199 to occur.
2. NOAUTOBACKOUT is mutually exclusive with PARALLELREAD; only one of these parameters may be specified in an ADARES BACKOUT run.

PLOGDBID: Alternate Protection Log ID

When performing a backout operation using a protection log from a database other than that specified by the ADARUN statement's DBID parameter, PLOGDBID specifies the database ID of the alternate protection log. The default is the database ID from the ADARUN-specified database.

RPLDATA: Replicate protection log data

The RPLDATA parameter allows you to specify whether or not the nucleus should replicate the protection log data sent to it. Valid values are "YES" and "NO"; the default is "NO".

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

TOBLK: Ending TOCP Block Number

TOBLK specifies the block number containing the TOCP checkpoint entry. TOBLK can only be specified if TOCP is specified.

TOCP: Ending Checkpoint for Backout

TOCP specifies the checkpoint at which the backout process is to stop. Backout processing continues up to the specified checkpoint. If backout processing is to continue until the beginning of the log, do not specify TOCP or NOAUTOBACKOUT.

TONUCID: Ending Nucleus ID

In a cluster environment, the NUCID parameter or a combination of the FROMNUCID and TONUCID parameters may be required to identify the nuclei associated with the checkpoints referenced in this backout run. If the starting checkpoint block (FROMBLK parameter) and ending checkpoint block (TOBLK parameter) are for the same nucleus, use the NUCID parameter; if the starting checkpoint block (FROMBLK) and ending checkpoint block (TOCP) are for different nuclei, use the FROMNUCID and TONUCID parameters.

The TONUCID parameter specifies the nucleus ID for the ending nucleus. Only specify a TONUCID parameter if a corresponding FROMNUCID parameter is also specified.

Example

1. ADADBS OPERCOM FEOFPL
2. User application on files 20 and 21 fails
3. ADADBS OPERCOM FEOFPL
4. ADARES BACKOUT DPLOG,FILE=20,21

This example assumes that the PLCOPY function is performed with user exit 2. Whenever a protection log switch occurs, this user exit submits a job to copy the content of the dual protection log to a sequential data set.

1. Switch to a new PLOG.
2. Run the user session creating PLOG data on the new PLOG data set.
3. Close the PLOG data set. User exit 2 submits a job which copies the contents of the PLOG data set just closed.
4. Perform a BACKOUT from that PLOG for the files 20 and 21 up to the beginning of the PLOG.

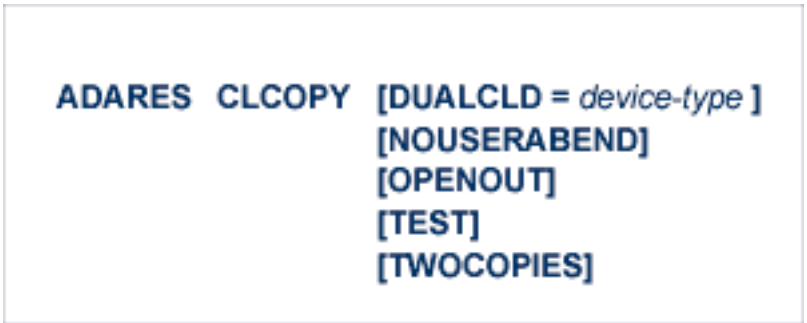
154 CLCOPY: Copy Dual Command Log

- Optional Parameters 1108
- Examples 1109

The CLCOPY function is used only if dual logging of command information was specified for the Adabas session. This function copies the data set that has the earlier time stamp to a sequential data set. Once the CLCOPY function is completed successfully, the copied data set is marked as empty. This function may, therefore, be used only once for any given data set.

Once the ADARES CLCOPY job has run for a CLOG data set, the ADARES utility checks the PPT to determine whether any additional CLOG data sets need to be copied. If so, it invokes user exit 2 or user exit 12, as appropriate, to accommodate the number of data sets that need copying. For example, if NCLOG=8, once the initial CLCOPY job completes, the ADARES utility will issue a call to the nucleus to invoke user exit 2 or 12 for each uncopied CLOG data set it detects. When it invokes user exit 12, it uses the flag "F" to identify this type of call; this flag is processed as a new EX12TYPE value in user exit 12. For more information about user exit 12, read *User Exit 12 (Multiple Data Set Log Processing)*, in the *Adabas User, Hyperdescriptor, Collation Descriptor, and SMF Exits Manual*.

The CLCOPY function is not allowed in single-user mode.



Optional Parameters

ADARES CLCOPY can be specified with no parameters.

DUALCLD: Dual Command Log Device Type

DUALCLD specifies the device type used for the dual command log data sets. This parameter is required if the device type used for the command log data set is different from that specified with the ADARUN DEVICE parameter.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

OPENOUT: Open DDSIAUS1/2 Output Data Sets

The OPENOUT parameter indicates that the DDSIAUS1/2 output data sets are to be opened by ADARES, even if no data is actually to be copied. Without OPENOUT, the sequential output data sets are not opened if ADARES detects an end-of-file condition while attempting to read the first input record; this may cause problems in some operating system environments. With OPENOUT, the output data sets are opened before the first input record is read.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

TWOCOPIES: Create Two Copies of Output

TWOCOPIES causes two copies of the output to be created.

Examples

Example 1:

```
ADARES CLCOPY
```

Dual command log is to be copied.

Example 2:

```
ADARES CLCOPY TWOCOPIES
```

Dual command log is to be copied. Two copies of the output are to be created.

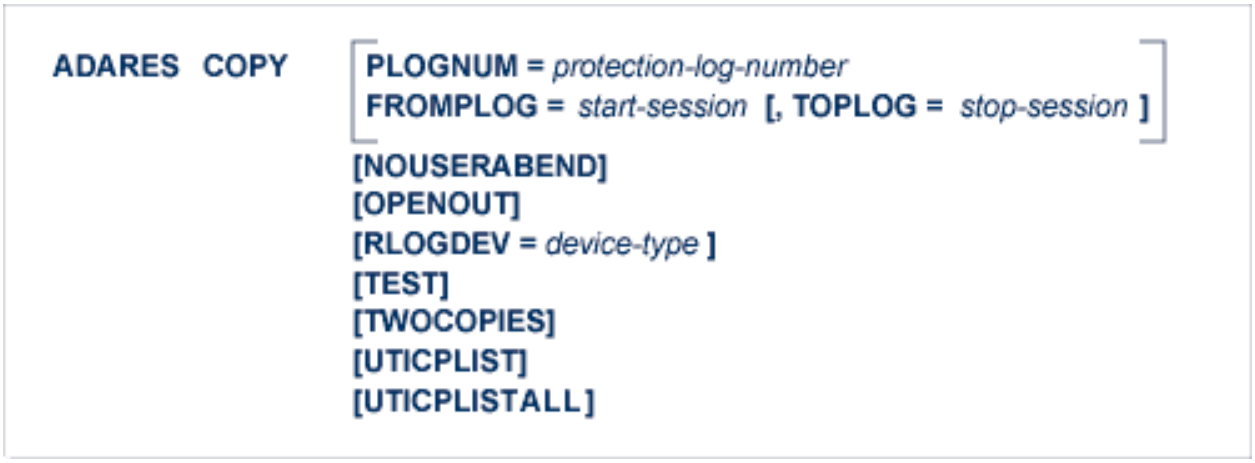
155 COPY: Copy a Sequential Protection Log or Save Tape

- Optional Parameters 1112
- Examples 1114

The COPY function copies an Adabas sequential protection log data set. If the Adabas session that created the sequential protection log data set was terminated abnormally, the COPY function must be executed before the data set can be used as input to any other ADARES function.

ADARES COPY

- must be used to copy a data protection log data set from disk to a tape data set before it can be used as input to the ADARES BACKOUT function.
- may be used even if subsequent Adabas sessions have created other data protection log data sets.
- also accepts ADASAV SAVE output (DDSAVEn) as input. Only one ADASAV SAVE input volume can be copied in a single ADARES COPY run. A SAVE output tape must be assigned to the DDSIIN job control file.
- may be executed any number of times for a given input data set.



The COPY function has special uses if you are using the Adabas Delta Save Facility Facility. Refer to the *Adabas Delta Save Facility Facility* documentation for more information.

ADARES COPY can be specified with no parameters. If ADARES COPY is specified without either PLOGNUM or FROMPLOG, the whole input protection log is copied.

Optional Parameters

FROMPLOG: Beginning Session for Copy

FROMPLOG specifies the session number at which the specified ADARES function is to start. ADARES searches the PLOG input (DDSIIN) file for the correct starting session.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump)

or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

OPENOUT: Open DDSIAUS1/2 Output Data Sets

The OPENOUT parameter indicates that the DDSIAUS1/2 output data sets are to be opened by ADARES, even if no data is actually to be copied. Without OPENOUT, the sequential output data sets are not opened if ADARES detects an end-of-file condition while attempting to read the first input record; this may cause problems in some operating system environments. With OPENOUT, the output data sets are opened before the first input record is read.

PLOGNUM: Protection Log Number

The Adabas protection log number of the data set to be copied. This number may be obtained from the database status report produced by the ADAREP utility. The output of the COPY function will be assigned the same log number.

RLOGDEV: Device Type for RLOG Data Set

The RLOGDEV parameter is used if the Adabas Recovery Aid (ADARAI) is active to specify a device-type for the recovery log (RLOG) data set.

If RLOGDEV is not specified (the default), the recovery log device-type is assumed to be the same as the ADARUN DEVICE parameter.

If the specified or default value for RLOGDEV is incorrect, ADARES COPY terminates with error 149, "missing or mismatching RLOGDEV parameter".

The RLOGDEV parameter makes it possible for ADARES to record its function for ADARAI, even if the GCBs of the database have been destroyed.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

TOPLOG: Ending PLOG Session for Backout

TOPLOG specifies the last session to be processed by the specified ADARES function. If ADARES finds a session on the PLOG input (DDSIIN) file that is greater than the specified TOPLOG session, that session is excluded from ADARES processing.

TWOCOPIES: Create Two Output Copies

TWOCOPIES causes two copies of the output to be created. If TWOCOPIES is not specified, the default is one copy.

UTICPLIST: Print Utility Checkpoints

The UTICPLIST parameter causes ADARES to select and print all SYNP, SYNV, and SYNS checkpoints found on the data protection log during the COPY function.

UTICPLISTALL: Print All Utility Checkpoints

The UTICPLISTALL parameter causes ADARES to select and print all SYNC, SYNF, SYNP, SYNS, SYNV, SYNX, SYN1, SYN2, SYN4, and SYN5 checkpoints found on the data protection log during the COPY function.

Examples

Example 1:

```
ADARES COPY PLOGNUM=6
```

Data protection log 6 is to be copied.

Example 2:

```
ADARES COPY PLOGNUM=8,TWOCOPIES
```

Data protection log 8 is to be copied. Two copies of the output are to be created.

156 MERGE CLOG: Merge Nucleus Cluster Command Logs

- Essential Parameter 1116

In an Adabas cluster environment, you can merge command logs (CLOGs) across a cluster in one of two ways:

- If your system is set up appropriately (CLOGMRG=YES and user exit 2), CLOGs are merged automatically.
- Otherwise, you can merge CLOGs manually using the ADARES MERGE CLOG utility function.

```
ADARES  MERGE  CLOG, NUMLOG = nn
```

Sequential data sets are expected as input to the MERGE CLOG function; therefore, the ADARES CLCOPY function must be executed prior to the ADARES MERGE function.

Essential Parameter

NUMLOG: Number of Command Log Data Sets

The NUMLOG parameter is required: it specifies the number of command log data sets to be included in the merge process. The maximum number is 32.

157 PLCOPY: Copy Protection Log to Sequential Data Set

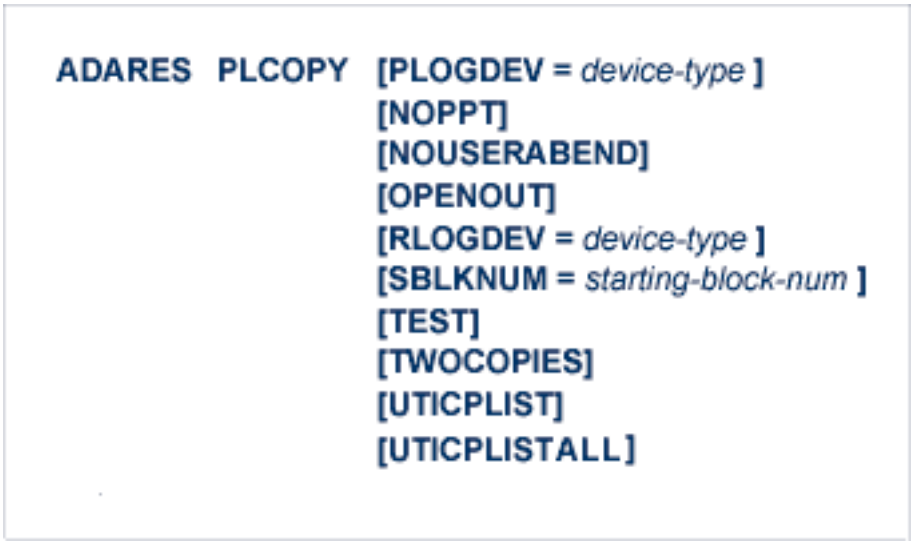
- Optional Parameters 1119
- Examples 1121

The PLCOPY function is used only if dual/multiple logging of protection information was specified for the Adabas session. This function copies the data set that has the earlier time stamp to a sequential data set. Once the PLCOPY function is successfully completed, the copied data set is marked as empty. This function may, therefore, be used only once in an Adabas session for any given data set.

Once the ADARES PLCOPY job has run for a PLOG data set, the ADARES utility checks the PPT to determine whether any additional PLOG data sets need to be copied. If so, it invokes user exit 2 or user exit 12, as appropriate, to accommodate the number of data sets that need copying. For example, if NPLOG=8, once the initial PLCOPY job completes, the ADARES utility will issue a call to the nucleus to invoke user exit 2 or 12 for each uncopied PLOG data set it detects. When it invokes user exit 12, it uses the flag "F" to identify this type of call; this flag is processed as a new EX12TYPE value in user exit 12. For more information about user exit 12, read *User Exit 12 (Multiple Data Set Log Processing)*, in the *Adabas User, Hyperdescriptor, Collation Descriptor, and SMF Exits Manual*.

The use of hardware compression (IDRC) is *not* recommended for protection log files. On z/OS systems, the BACKOUT function will take at least twice as long to run when processing compressed data.

The PLCOPY function is not allowed in single-user mode.



The PLCOPY function has special uses if you are using the Adabas Delta Save Facility. Refer to the *Adabas Delta Save Facility Facility* documentation for more information.

ADARES PLCOPY can be specified with no parameters.

Optional Parameters

PLOGDEV: PLOG Device Type

PLOGDEV specifies the device type used for dual/multiple protection log data sets. This parameter is required if the device type used for the dual/multiple protection log data set is different from that specified with the ADARUN DEVICE parameter.

NOPPT (Clustered Nucleus Environments Only)

The parallel participant table (PPT) tells ADARES PLCOPY which data sets to copy. If the PPT is destroyed, the ADARES NOPPT function allows the DBA to specify the PLOG data sets that are to be copied and merged.

If ADARAI is used, the PLOG data sets are written to the RLOG at nucleus initialization. In the event of a failure and a final PLCOPY is still needed, ADARAI can construct the PLCOPY NOPPT JCL from the PLOG data sets written to the RLOG.

NOPPT is intended only for emergency use when the PPT has been overwritten. It specifies that the PPT is to be ignored and DDPLOG data sets are to be supplied with JCL.

Caution: Use this parameter cautiously since it ignores the PPT and all control-type information typically provided by the PPT.

When you use this parameter, you must supply

- the correct intermediate data set; and
- the correct input protection logs from all nuclei in the form of DDPLOG01-nn.

Caution: Without the PPT, ADARES cannot perform any extensive validations on the input data sets.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

OPENOUT: Open DDSIAUS1/2 Output Data Sets

The OPENOUT parameter indicates that the DDSIAUS1/2 output data sets are to be opened by ADARES, even if no data is actually to be copied. Without OPENOUT, the sequential output data sets are not opened if ADARES detects an end-of-file condition while attempting to read

the first input record; this may cause problems in some operating system environments. With OPENOUT, the output data sets are opened before the first input record is read.

RLOGDEV: Device Type for RLOG Data Set

The RLOGDEV parameter is used if the Adabas Recovery Aid (ADARAI) is active to specify a device-type for the recovery log (RLOG) data set.

If RLOGDEV is not specified (the default), the recovery log device type is assumed to be the same as the ADARUN DEVICE parameter.

If the specified or default value for RLOGDEV is incorrect, ADARES PLCOPY terminates with error 149, "missing or mismatching RLOGDEV parameter".

The RLOGDEV parameter makes it possible for ADARES to record its function for ADARAI, even if the GCBs of the database have been destroyed.

SBLKNUM

The SBLKNUM parameter can only be specified in conjunction with the NOPPT parameter and only for the PLCOPY function.

SBLKNUM allows the user to specify the starting block number for the sequential merge output. If this parameter is omitted, an attempt will be made to read the PPT and obtain the block number from there. If this read fails, the output will start with block one.

To determine the value for this parameter, the user must look at the output from the previous PLCOPY and use the next block number in sequence.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

TWOCOPIES: Create Two Copies of Output

TWOCOPIES causes two copies of the output to be created. If TWOCOPIES is not specified, the default is one copy.

UTICPLIST: Print Utility Checkpoints

The UTICPLIST parameter causes ADARES to select and print all SYNP, SYNV, and SYNS checkpoints found on the data protection log during the PLCOPY function.

UTICPLISTALL: Print All Utility Checkpoints

The UTICPLISTALL parameter causes ADARES to select and print all SYNC, SYNF, SYNP, SYNS, SYNV, SYNX, SYN1, SYN2, SYN4, and SYN5 checkpoints found on the data protection log during the PLCOPY function.

Examples

Example 1:

Copy the dual/multiple protection log.

```
ADARES PLCOPY
```

Example 2:

Create two copies of the dual/multiple protection log.

```
ADARES PLCOPY TWOCOPIES
```

Example 3:

Copy the dual/multiple protection log. The Adabas Recovery Aid (ADARAI) is active. The recovery log (RLOG) device type is 8390.

```
ADARES PLCOPY RLOGDEV=8390
```

158 REGENERATE: Regenerate Updates

- Syntax 1125
- Essential Parameters 1125
- Optional Parameters and Subparameters 1126
- Examples 1132
- Output Statistics 1133

The REGENERATE function reapplies all the updates performed between two checkpoints.

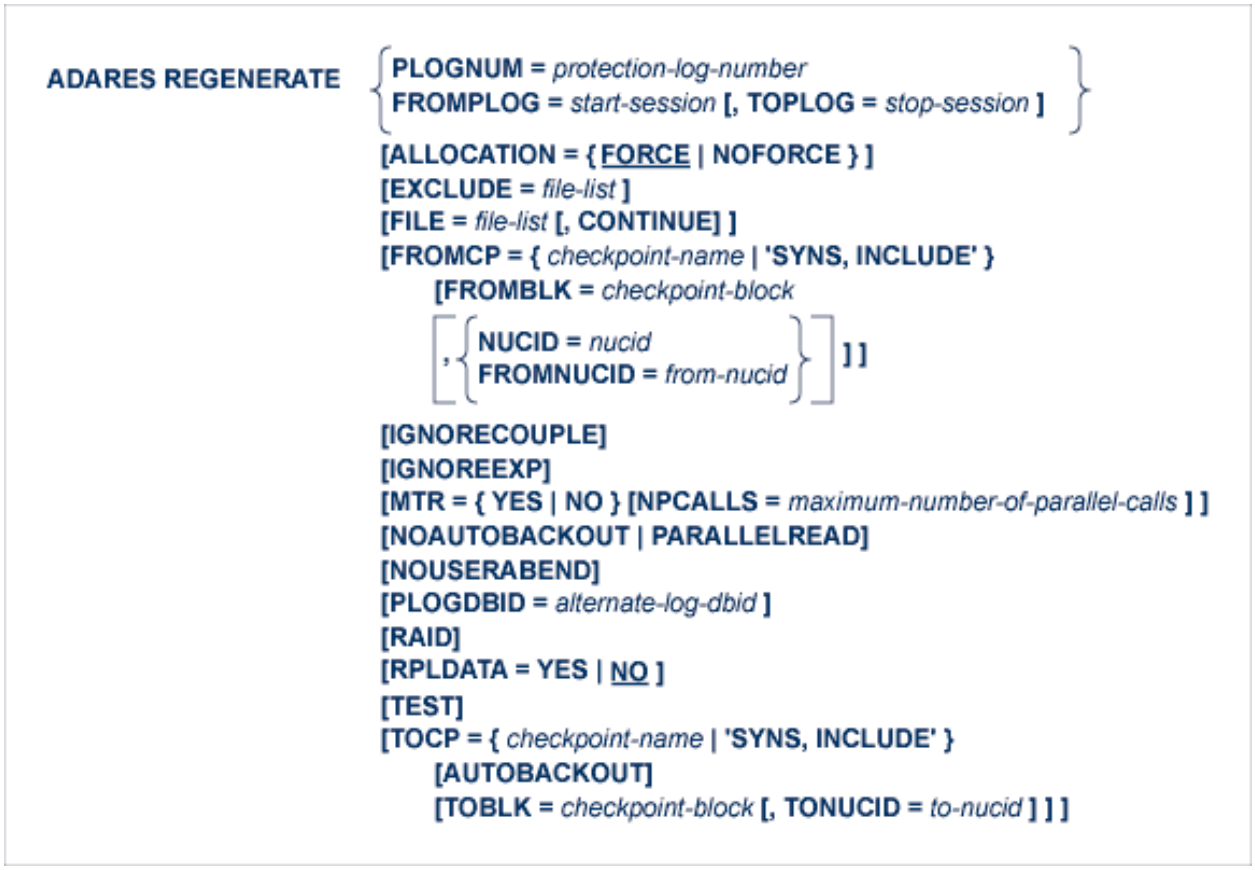
In addition to restoring normal updates, ADARES REGENERATE also restores any of the following ADADBS utility (or Adabas Online System) function updates that were performed between the specified checkpoints for the selected file or files:

ALLOCATE	DELETE	NEWFIELD	RELEASE
CHANGE	DSREUSE	PRIORITY	RENAME
DEALLOCATE	ISNREUSE	RECOVER	RENUMBER
DELCP	MODFCB	REFRESH	UNCOUPLE

For the database, all file-related operations listed above are performed, plus any of the following ADADBS (or Adabas Online System) database-related functions:

ADD	INCREASE (data set size)
DECREASE (data set size)	RECOVER

Syntax



Essential Parameters

You can specify either the log number (PLOGNUM) or the session number (FROMPLOG) of the protection log as a starting point for REGENERATE processing. If you specify a session number, you can also specify a range of sessions to be processed using the TOPLOG parameter.

FROMPLOG: Beginning Session for Regeneration

FROMPLOG specifies the session number at which the specified ADARES function is to start. ADARES searches the PLOG input file for the correct starting session. To define the starting point more precisely, specify the FROMCP and FROMBLK parameters.

Note: If only FROMPLOG is specified (without the TOPLOG parameter), only the session number specified by FROMPLOG is regenerated. If PLOGs with higher session numbers are concatenated, they are ignored.

PLOGNUM: Protection Log Number

PLOGNUM is the log number of the data protection log to be used as input for regenerate processing. The log number may be obtained from the database status report.

Optional Parameters and Subparameters

ALLOCATION: Action to Follow File Extent Allocation Failure

ALLOCATION specifies the action to be taken if file extent allocations cannot be obtained according to the placement parameters ACRABN, DSRABN, NIRABN, or UIRABN.

ALLOCATION concerns the following operations, which are replayed as part of the regeneration:

- ADADBS ALLOCATE
- Adabas Online System "Define File"
- Adabas Online System "Install/Change DLOG Area"

By default (that is, ALLOCATION=FORCE), the utility terminates with error if any file extent allocation cannot be met according to RABN placement parameters.

If ALLOCATION=NOFORCE is specified and any allocation with placement parameters fails, the utility retries the allocation without the placement parameter.

AUTOBACKOUT: Back Out Transactions from TOCP Checkpoint

When the TOCP parameter is specified, incomplete transactions are not normally backed out at the end of processing. This allows you to reexecute the utility function that corresponds to the TOCP checkpoint, followed by another ADARES operation with FROMCP specifying the starting checkpoint.

In situations where a REGENERATE/BACKOUT should end at the TOCP checkpoint, using the AUTOBACKOUT parameter to back out incomplete transactions ensures the logical consistency of the database. Note that AUTOBACKOUT is allowed only if TOCP is specified.

CONTINUE: Continue File Recovery with Autobackout

CONTINUE allows AUTOBACKOUT of any incomplete transaction changes during file regeneration. If specified, all changes made by incomplete transactions are backed out of the database data sets specified by the FILE parameter.

If the file list contains either coupled or expanded component files and CONTINUE is specified, the usual checking of the list for inclusion of complete coupled pairs and/or component file sets is not performed; in this case, IGNORECOUPLE or IGNOREEXP does not have to be specified to stop the respective file list check.

If CONTINUE is specified, the complete database is locked for use by the REGENERATE function only.

EXCLUDE: Exclude Specified Files from Regenerate

EXCLUDE lists the numbers of the files to be excluded from REGENERATE processing; that is, the files that are not to be regenerated. Any protection records that pertain to these files are ignored.

The parameter is optional: if not specified, no files are excluded. A file number may be listed only once.

When the FILE parameter is specified, all files specified in the EXCLUDE parameter must also be specified in the FILE parameter.

The EXCLUDE parameter has no bearing on whether the REGENERATE is performed with or without transaction logic.

The EXCLUDE parameter is provided for use in recovery jobs built by the Adabas Recovery Aid (ADARAI).

Excluded files are listed in the extended CPLIST of the ADAREP report.

FILE: Files to Be Included

FILE specifies the files to be included in the regeneration process. If all files are to be included, do not specify the FILE parameter. If the specified file is a component file of an Adabas expanded file, all other component files of the expanded file must also be specified here. If a specified file is coupled to other files, the coupled files must also be specified.



Note: Before beginning, ADARES locks all specified files for the duration of REGENERATE execution. If the FILE parameter is omitted, the entire database will be locked.

FROMBLK: Starting Block for Regeneration

FROMBLK specifies the block number in which the FROMCP checkpoint entry is contained. This block number may be obtained from the previous ADASAV restore output or database status report. It refers to PLOGNUM or FROMPLOG. FROMBLK can be specified only if FROMCP is specified.

FROMCP: Starting Checkpoint for Regeneration

FROMCP defines the checkpoint after which the REGENERATE process is to begin. Processing begins with the information following the specified checkpoint. The checkpoint name may be obtained from the previous ADASAV restore output (SYN2/5), the database status report, or the ADARES COPY/PLCOPY output resulting from specifying UTICPLIST. If processing is to begin at the beginning of the log, do not specify the FROMCP parameter. FROMCP refers to the protection log specified by PLOGNUM or FROMPLOG.

For information about the 'SYNS,INCLUDE' option, see the section *INCLUDE: Include Checkpoint in Regeneration*.

FROMNUCID: Starting Nucleus ID

In a cluster environment, the NUCID parameter or a combination of the FROMNUCID and TONUCID parameters may be required to identify the nuclei associated with the checkpoints

referenced in this regeneration run. If the starting checkpoint block (FROMBLK parameter) and ending checkpoint block (TOBLK parameter) are for the same nucleus, use the NUCID parameter; if the starting checkpoint block (FROMBLK) and ending checkpoint block (TOCP) are for different nuclei, use the FROMNUCID and TONUCID parameters.

The FROMNUCID parameter specifies the nucleus ID for the starting nucleus. If you specify the FROMNUCID parameter, a corresponding TONUCID parameter is expected.

IGNORECOUPLE: Ignore Unspecified Coupled Files

IGNORECOUPLE (or CONTINUE) stops the REGENERATE function from checking the FILE list for complete coupled file pairs. If neither CONTINUE nor IGNORECOUPLE is specified and the FILE list specifies a coupled file without specifying its mate, ADARES terminates and issues an error message.

IGNOREEXP: Ignore Expanded Component Files

If the FILE list includes any Adabas expanded component files, ADARES BACKOUT normally checks to ensure that all *related* component files are also in the list; if not, ADARES ends the REGENERATE operation and issues an error message. Specifying IGNOREEXP (or CONTINUE) stops the checking for related component files.

INCLUDE: Include Checkpoint in Regeneration

The optional keyword INCLUDE specified for FROMCP and/or TOCP includes the checkpoint where the regenerate starts/stops in the operation; that is, the function associated with the checkpoint is reexecuted. The checkpoint name must be SYNS, since ADARES can reexecute only functions associated with SYNS checkpoint. The checkpoint name and parameter combination 'SYNS,INCLUDE' must be enclosed in apostrophes.

If INCLUDE is not specified (the default), the REGENERATE operation starts immediately *after* the checkpoint specified by FROMCP and stops immediately *before* the checkpoint specified by TOCP.

The INCLUDE parameter is provided for use in recovery jobs built by the Adabas Recovery Aid (ADARAI).

MTR: Multithreaded Regenerate Switch

MTR=YES activates the multithreaded regenerate feature; MTR=NO disables it.

When the multithreaded regenerate feature is active, multiple buffers containing PLOG information are sent to the Adabas nucleus in parallel to improve performance. When the feature is not active, only one buffer is sent to Adabas at a time.

If the nucleus ADARUN parameter MODE=SINGLE, MTR is automatically set to NO. Multiple threads are not available to Adabas running in single user mode.

If the FILE parameter is not specified, or is specified with CONTINUE, the default value for MTR is YES. In these cases, multithreaded regenerate has exclusive control of the whole database and is generally effective.

Otherwise, the default value is NO. If it only has exclusive control of some files, as is the case when FILE is specified *without* CONTINUE, multithreaded regenerate can run in parallel with normal applications accessing different files and has the potential to negatively impact the performance of production applications.

NOAUTOBACKOUT: Prevent Incomplete Transaction Backout

NOAUTOBACKOUT stops the normal backout of incomplete transactions at the end of REGENERATE operation. Normally, ADARES performs an automatic backout of all incomplete logical transactions at the end of the function if both of the following are true:

- The REGENERATE was for the entire database (FILE parameter omitted), or the CONTINUE parameter was specified; and
- The TOCP parameter was omitted, which implies that processing is to be performed until the end of the input data set is reached.

If several consecutive REGENERATE runs are needed to process multiple protection logs resulting from a single Adabas session, an automatic backout should be performed only for the last input log. The NOAUTOBACKOUT parameter should therefore be specified for each REGENERATE run except for the run in which the last input log is used.



Notes:

1. NOAUTOBACKOUT cannot be specified in single-user mode.
2. NOAUTOBACKOUT is mutually exclusive with PARALLELREAD; only one of these parameters may be specified in an ADARES BACKOUT run.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

NPCALLS: Maximum Number of Parallel Calls

When MTR=YES, the NPCALLS parameter may be specified to limit the number of parallel calls sent to the Adabas nucleus.

If the FILE parameter is not specified, or is specified with CONTINUE, the default value for NPCALLS is the nucleus ADARUN parameter NT+1 or NC, whichever is smaller.

If the FILE parameter is specified without CONTINUE, the default value is the nucleus ADARUN parameter NT+1 or NC/2, whichever is smaller.

NPCALLS is primarily used to reduce the number of parallel calls allowed by the default value. Fewer parallel calls mean a smaller nucleus workload produced by ADARES. This is especially useful for increasing the resources available to application programs running in parallel with REGENERATE FILE.

NUCID: Nucleus ID

In a cluster environment, the NUCID parameter or a combination of the FROMNUCID and TONUCID parameters may be required to identify the nuclei associated with the checkpoints referenced in this regeneration run. If the starting checkpoint block (FROMBLK parameter) and ending checkpoint block (TOBLK parameter) are for the same nucleus, use the NUCID parameter; if the starting checkpoint block (FROMBLK) and ending checkpoint block (TOCP) are for different nuclei, use the FROMNUCID and TONUCID parameters.

PARALLELREAD: Enable Read-Only File Usage by Other Users

The PARALLELREAD parameter provides for concurrent read-only access to the files being processed by ADARES REGENERATE both for database-wide and file-oriented functions:

- for file-oriented functions, specifying PARALLELREAD causes ADARES to issue an OPEN call with "EXU=file-list" in the record buffer. This allows read-only access to the files for other users while ADARES is active.
- when FILE is not specified or when CONTINUE is specified, the PARALLELREAD parameter is effective for database-wide session regeneration. The parameter makes it possible for read-only users to access the database at the same time the database session is being regenerated.

Update commands are rejected.

If parallel access users read records that were updated in the database session being regenerated, they may see record images that are logically wrong in the sense of the application, or response codes such as 113 (ADARSP113) that indicate inconsistencies.

Notes:

1. During ADARES operation with PARALLELREAD, temporary differences between the Associator and Data Storage may cause nucleus responses 113 or 199 to occur.
2. NOAUTOBACKOUT is mutually exclusive with PARALLELREAD; only one of these parameters may be specified in an ADARES BACKOUT run.

PLOGDBID: Alternate Protection Log ID

PLOGDBID specifies an alternate DBID from which the PLOG has been taken. When regenerating with a protection log from a database other than that specified by the ADARUN statement's DBID parameter, use PLOGDBID to specify the database ID of the alternate protection log. The default is the database ID (DBID) from the ADARUN-specified database.

RAID: Action to Follow Receipt of Nucleus Response Code or Utility Checkpoint

The RAID parameter terminates a regeneration with error 146 whenever a file is to be excluded because a utility checkpoint (other than ADADBS or Adabas Online System checkpoints) was encountered or a nucleus response code was received for the file.

If RAID is not specified (the default), ADARES continues processing the other files after a file is excluded from REGENERATE processing.

RAID is provided for use in recovery jobs built by the Adabas Recovery Aid (ADARAI).

RPLDATA: Replicate protection log data

The RPLDATA parameter allows you to specify whether or not the nucleus should replicate the protection log data sent to it. Valid values are "YES" and "NO"; the default is "NO".

ADARES REGENERATE with RPLDATA=YES will not replicate data for a file with spanned DS record support.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

TOBLK: Ending TOCP Block

TOBLK specifies the block number in which the TOCP checkpoint entry is contained. TOBLK, which can be specified only if TOCP is also specified, refers to the protection log specified by TOPLOG, if specified, or else by PLOGNUM or FROMPLOG.

TOCP: Ending Checkpoint Block for Regenerate

TOCP specifies the checkpoint before which the REGENERATE process is to stop. Processing continues up to, but not including, the specified checkpoint. If REGENERATE processing is to continue until the end of the log, do not specify TOCP. TOCP refers to the protection log specified by TOPLOG, if specified, or else PLOGNUM or FROMPLOG.

For information about the 'SYNS,INCLUDE' option, see the section *INCLUDE: Include Checkpoint in Regeneration*.

TONUCID: Ending Nucleus ID

In a cluster environment, the NUCID parameter or a combination of the FROMNUCID and TONUCID parameters may be required to identify the nuclei associated with the checkpoints referenced in this regeneration run. If the starting checkpoint block (FROMBLK parameter) and ending checkpoint block (TOBLK parameter) are for the same nucleus, use the NUCID parameter; if the starting checkpoint block (FROMBLK) and ending checkpoint block (TOCP) are for different nuclei, use the FROMNUCID and TONUCID parameters.

The TONUCID parameter specifies the nucleus ID for the ending nucleus. Only specify a TONUCID parameter if a corresponding FROMNUCID parameter is also specified.

TOPLOG: Ending PLOG Session for Regenerate

TOPLOG specifies the last session to be processed by the specified ADARES function. If ADARES finds a session on the PLOG input file that is greater than the specified TOPLOG session, that session is excluded from ADARES processing. If TOPLOG is not specified, the FROMPLOG session becomes the default.

Examples

Example 1:

```
ADARES REGENERATE PLOGNUM=4
```

All files are to be included in regenerate processing. The protection log number is 4. Regenerate processing is to begin at the beginning of the log and is to end at the end of the log. At the end of REGENERATE processing, incomplete transactions are automatically backed out.

Example 2:

```
ADARES REGENERATE
FILE=4,7,FROMPLOG=11,FROMCP=CH01,FROMBLK=106,
ADARES TOPLOG=12,TOCP=CH05,TOBLK=2031
```

Regenerate processing is to be limited to files 4 and 7. All updates applied to files 4 and 7 between the taking of checkpoints CH01 and CH05 are to be reapplied. CH01 is located in block 106 of data protection log 11. Checkpoint CH05 is located in block 2031 of data protection log 12. No automatic backout of incomplete transactions occurs following REGENERATE processing, as in the previous example.

Example 3:

```
ADARES REGENERATE EXCLUDE=10,11,12
```

Files 10 through 12 are excluded from the REGENERATE database function. No changes to these files are replayed.

Example 4:

```
ADARES REGENERATE
ADARES FROMCP='SYNS,INCLUDE',FROMBLK=123
ADARES TOCP=SYNP,TOBLK=234
```

1. ADARES regenerates the database.
2. The REGENERATE starts at the SYNS checkpoint in PLOG block 123; ADARES reexecutes the associated ADADBS/Adabas Online System function.
3. The REGENERATE stops just before the SYNP checkpoint in block 234; ADARES does *not* replay the associated utility function.

Example 5:

```
ADARES REGENERATE FILE=10
ADARES FROMCP='SYNS,INCLUDE',FROMBLK=345
ADARES TOCP='SYNS,INCLUDE',TOBLK=456
```

1. ADARES regenerates file 10.

2. The REGENERATE starts at the SYNS checkpoint in PLOG block 345; ADARES reexecutes the associated ADADBS/Adabas Online System function if it pertains to file 10.
3. The REGENERATE stops at the SYNS checkpoint in block 456; ADARES replays the associated ADADBS/Adabas Online System function if it pertains to file 10.

Example 6:

```
ADARES REGENERATE
ADARES RAID
```

1. ADARES regenerates the database.
2. ADARES reexecutes all database updates found on the input PLOG.
3. ADARES immediately terminates with error 146 if it receives a nucleus response code or encounters a utility checkpoint other than from ADADBS or Adabas Online System.

Output Statistics

File processing statistics from ADARES REGENERATE function processing are provided at the end of the run. These statistics include the number of data storage records regenerated for each file as well as information about the PLOG blocks read and sent, the commands and transactions processed, the number of Adabas calls processed (including maximum and average calls processed in parallel), and the average record buffer size. The statistics are gathered via 4-byte counters that keep count of the total data storage updates for each file during REGENERATE processing.

Spanned records and records with large object (LB) fields can span more than one data storage block. Thus an update to a record of this type will increment the data record update counter by one for each data storage block. For example, suppose a spanned record is updated that is stored across three data storage blocks. In this case, the record update counter is incremented three times, one for each data storage block.

Data storage records that were modified in a transaction that did not successfully terminate with an ET command are counted, even though they are backed out by the nucleus.

The following is a sample of an ADARES REGENERATE report (when MTR=YES):

Note: The number of records listed as "Data Record Updates" is the number of data storage records processed; the number of records listed as "PLOG Records sent to ADABAS" is the number of input records from the protection log (PLOG). The PLOG record count is usually much larger than the data record update count.

```
(REGENERATE) File Processing Statistics

-------------------------------------------
I   File Number   I   Data Record Updates   I
I-----------------I-------------------------I
I        10       I               127       I
-------------------------------------------


Multi-Threading Processing Statistic
------------------------------------

PLOG Blocks Read from Input           17
PLOG Records Sent to ADABAS          405
Commands Processed                   158
Transactions Processed                28
Number of ADABAS Calls                28
Maximum Calls in Parallel             23
Average Calls in Parallel             10
Average Record Buffer Size          1574
```

The following is a sample of an ADARES REGENERATE report when MTR=NO:

```
(REGENERATE) File Processing Statistics

-------------------------------------------
I   File Number   I   Data Record Updates   I
I-----------------I-------------------------I
I        10       I               127       I
-------------------------------------------


ADARES (REGENERATE) Normal end: 17 blocks / 405 records processed.
```

159 REPAIR: Repair Data Storage Blocks

- Syntax 1332
- Essential Parameter 1137
- Optional Parameters 1137
- Examples 1138

Caution: The REPAIR function can cause data loss if not used correctly. It should only be used with guidance from your technical support representative.

The REPAIR function may be used to repair one or more Data Storage blocks, using the protection log and the output of the ADASAV utility.

Notes:

1. An interrupted REPAIR function must be reexecuted from the beginning.
2. The REPAIR function should *not* be run if any of the following utility functions have changed the RABN ranges since the last ADASAV SAVE operation: `ADAORD`, `ADALOD`, `ADADBS DEALLOCATE`, `ADASAV RESTORE FMOVE`
3. The DDSIIN input must be concatenated in the following sequence: `ADASAV SAVE (DDSAVEn) output;, protection log.`

Syntax

```
ADARES  REPAIR   DSRABN = { rabn  |  rabn - rabn }
                 [FILE = locked-file-list ]
                 [NOUSERABEND]
                 [TEST]
```

Essential Parameter

DSRABN: Data Storage RABN or RABNs to Be Repaired

DSRABN specifies one or more Data Storage RABNs to be repaired. Either a single RABN or a range of RABNs (for example, 1000-1234) can be specified.

Optional Parameters

FILE: Locked File List

FILE locks one or more files so that they cannot be read or updated by any user during REPAIR execution. Only the files specified are locked for the exclusive use of ADARES REPAIR. Files not included in the list remain available to other users of the database. If FILE is not specified, the entire database is locked; the user queue must be empty.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

Examples

Example 1:

```
ADARES REPAIR DSRABN=1434,FILE=20
```

Repair Data Storage block 1434. Only file 20 is locked during file processing.

Example 2:

```
ADARES REPAIR DSRABN=1462-2543
```

Repair Data Storage blocks 1462 through 2543.

160 Multithreaded Processing Statistics

When running ADARES BACKOUT, BACKOUT DPLOG, or REGENERATE with MTR=YES, a table with processing statistics is printed to DDDRUCK after successful completion of the utility. For example:

```
MULTI - THREADING PROCESSING STATISTIC
--------------------------------------

 PLOG BLOCKS READ FROM INPUT          20472
 PLOG RECORDS SENT TO ADABAS         764554
 COMMANDS PROCESSED                  302273
 TRANSACTION PROCESSED                55045
 NUMBER OF ADABAS CALLS               56450
 MAXIMUM CALLS IN PARALLEL               71
 AVERAGE RECORD BUFFER SIZE            1403  ↩
```

Field	Description
PLOG blocks read from input	Number of PLOG blocks read from the input protection log
PLOG records sent to Adabas	Number of PLOG records selected for backout or regenerate processing
Commands processed	Number of update commands processed (N1, E1, ...)
Transactions processed	Number of transactions backed out or regenerated
Number of Adabas calls	Number of Adabas calls issued to perform the backout or regenerate
Maximum calls in parallel	Maximum number of backout or regenerate calls processed in parallel by the nucleus
Average calls in parallel	Average number of backout or regenerate calls processed in parallel by the nucleus during this ADARES run
Average record buffer size	Average size of the record buffers used for the backout or regenerate calls

161 JCL Requirements and Examples

- Requirements to Run ADARES JCL 1142

This section describes the job control information required to run ADARES on the z/OS operating system and shows examples of job streams.

Notes:

1. The DDSIAUS1/2 device type used to copy the protection log may not support the BACKOUT function if it is an IDRC (hardware compression) device. For more information, see the description of the PLCOPY function earlier in this document.
2. When running with the optional Recovery Aid (ADARAI), all temporary data sets must also be cataloged in the job control.

Requirements to Run ADARES JCL

Data Set	DD Name	Storage	More Information
Sequential protection log or ADASAV DDSAVEn output	DDSIIN	tape/ disk	Input log for COPY, REGENERATE, and REPAIR functions.
Multiple protection log	DDPLOGRn	disk	Input logs for PLCOPY function, and BACKOUT DPLOG/MPLOG .
Multiple command log	DDCLOGRn	disk	Input logs for CLCOPY function.
Sequential protection log	DDBACK	tape/ disk	Input log for BACKOUT function (not BACKOUT DPLOG).
Copied log	DDSIAUS1	tape/ disk	Output of COPY, CLCOPY, PLCOPY functions.
Extra copied log	DDSIAUS2	tape/ disk	Required only if two copies are to be produced by a copy function (with TWOCOPIES).
Recovery log (RLOG)	DDRLOGR1	disk	Required when using ADARAI.
Data Storage	DDDATARn	disk	Required only for REGENERATE if FROMCP=SYN1 or SYN4.
Associator	DDASSORn	disk	
ADARUN parameters	DDCARD	reader	*Operations*
ADARES parameters	DDKARTE	reader	
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADARES messages	DDDRUCK	printer	*Messages and Codes*

ADARES JCL Examples

Copy Sequential Protection Log

```
//ADARESCP  JOB
//*
//*    ADARES: COPY SEQUENTIAL PROTECTION LOG
//*
//RES      EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD        <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1  <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1  <=== DATA
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1  <=== WORK
//DDSIIN    DD   DSN=EXAMPLE.DByyyyy.SIBA,            <=== PLOG
//            VOL=SER=vvvvvv,DISP=OLD,UNIT=TAPE
//DDSIAUS1  DD   DSN=EXAMPLE.DByyyyy.PLOG(+1),        <=== PLOG COPY
//            VOL=SER=vvvvvv,UNIT=TAPE,DISP=(NEW,CATLG)
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADARES,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADARES COPY
/*
```

Refer to ADARESCP in the JOBS data set for this example.

Copy Dual/Multiple Protection Log

```
//ADARESCD  JOB
//*
//*    ADARES: COPY DUAL/MULTIPLE PROTECTION LOG
//*             TWO COPIES OF OUTPUT ARE TO BE CREATED
//*
//RES      EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD       <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1  <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1  <=== DATA
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1  <=== WORK
//DDPLOGR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.PLOGR1  <=== PLOG1
//DDPLOGR2  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.PLOGR2  <=== PLOG2
//DDSIAUS1  DD   DSN=EXAMPLE.DByyyyy.PLOG1(+1),       <=== PLOG COPY 1
//            VOL=SER=vvvvvv,UNIT=TAPE,DISP=(NEW,CATLG)
//DDSIAUS2  DD   DSN=EXAMPLE.DByyyyy.PLOG2(+1),       <=== PLOG COPY 2
//            VOL=SER=vvvvvv,UNIT=TAPE,DISP=(NEW,CATLG)
```

```
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADARES,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADARES PLCOPY TWOCOPIES
/*
```

Refer to ADARESCD in the JOBS data set for this example.

Automatically Copy/Merge Nucleus Cluster Protection Logs

Note: Note that when intermediate data sets are used for both CLCOPY and PLCOPY, the data set names must be unique so that they are not overwritten.

Following is sample JCL for allocating the required intermediate data sets MERGIN1 and MERGIN2:

```
//ALLOC JOB
//*
//* Example to allocate the MERGIN1 and the MERGIN2 data sets
//*
//ALLOC   EXEC PGM=IEFBR14
//MERGIN1 DD  DISP=(NEW,CATLG,DELETE),DSN=EXAMPLE.PINTERI,
//             SPACE=(CYL,(1,10,0)),UNIT=3390,VOL=SER=volser,
//             RECFM=VB,BLKSIZE=27998,LRECL=27994
//MERGIN2 DD  DISP=(NEW,CATLG,DELETE),DSN=EXAMPLE.PINTERO,
//             SPACE=(CYL,(1,10,0)),UNIT=3390,VOL=SER=volser,
//             RECFM=VB,BLKSIZE=27998,LRECL=27994
```

Refer to ADARESMP in the JOBS data set for this example.

Automatically Copy/Merge Nucleus Cluster Protection Logs Ignoring PPT

```
//ADARESIP  JOB
//*
//*    ADARES: COPY/MERGE DUAL/MULTIPLE PROTECTION LOGS FROM ALL
//*             NUCLEI IN AN ADABAS CLUSTER
//*             PPT IS TO BE IGNORED
//*             THIS IS ONLY FOR EMERGENCY USE WHEN THE PPT HAS BEEN
//*             OVER-WRITTEN - USE CAUTION WHEN SUBMITTING
//*
//RES      EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD      <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DBYYYYY.ASSOR1   <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DBYYYYY.DATAR1   <=== DATA
//DDPLOG01  DD   DISP=SHR,DSN=EXAMPLE.DBYYYYY.PLOGR1   <=== PLOG1 NUC1
```

```
//DDPLOG02  DD   DISP=SHR,DSN=EXAMPLE.DBYYYYY.PLOGR2   <=== PLOG2 NUC1
//DDPLOG03  DD   DISP=SHR,DSN=EXAMPLE.DBYYYYY.PLOGR1A  <=== PLOG1 NUC2
//DDPLOG04  DD   DISP=SHR,DSN=EXAMPLE.DBYYYYY.PLOGR2A  <=== PLOG2 NUC2
//DDPLOG05  DD   DISP=SHR,DSN=EXAMPLE.DBYYYYY.PLOGR1B  <=== PLOG1 NUC3
//DDPLOG06  DD   DISP=SHR,DSN=EXAMPLE.DBYYYYY.PLOGR2B  <=== PLOG2 NUC3
//MERGIN2   DD   DISP=SHR,DSN=EXAMPLE.INTERO           <=== INTER
//MERGIN1   DD   DISP=SHR,DSN=EXAMPLE.INTERI           <=== INTER
//DDSIAUS1  DD   DSN=EXAMPLE.DBYYYYY.PLOG1(+1),        <=== PLOG COPY
//            VOL=SER=ADAXXX,UNIT=TAPE,DISP=(NEW,CATLG)
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADARES,MODE=MULTI,SVC=XXX,DEVICE=3390,DBID=YYYYY
/*
//DDKARTE   DD   *
ADARES PLCOPY NOPPT
/*
//                                                        ↩
```

Refer to ADARESIP in the JOBS data set.

Copy Dual/Multiple Command Log

```
//ADARESCC  JOB
//*
//*     ADARES: COPY DUAL/MULTIPLE COMMAND LOG
//*
//RES      EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD       <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <=== WORK
//DDCLOGR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.CLOGR1 <=== CLOG1
//DDCLOGR2  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.CLOGR2 <=== CLOG2
//DDSIAUS1  DD   DSN=EXAMPLE.DByyyyy.CLOG,           <=== OUTPUT OF
//            VOL=SER=vvvvvv,UNIT=TAPE,DISP=(NEW,CATLG)       CLCOPY
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADARES,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADARES CLCOPY
/*
```

Refer to ADARESCC in the JOBS data set for this example.

Automatically Copy/Merge Nucleus Cluster Command Logs

Note: Note that when intermediate data sets are used for both CLCOPY and PLCOPY, the data set names must be unique so that they are not overwritten.

Following is sample JCL for allocating the required intermediate data sets MERGIN1 and MERGIN2:

```
//ALLOC JOB
//*
//* Example to allocate the MERGIN1 and the MERGIN2 data sets
//*
//ALLOC   EXEC PGM=IEFBR14
//MERGIN1 DD  DISP=(NEW,CATLG,DELETE),DSN=EXAMPLE.CINTERI,
//             SPACE=(CYL,(1,10,0)),UNIT=3390,VOL=SER=volser,
//             RECFM=VB,BLKSIZE=27998,LRECL=27994
//MERGIN2 DD  DISP=(NEW,CATLG,DELETE),DSN=EXAMPLE.CINTERO,
//             SPACE=(CYL,(1,10,0)),UNIT=3390,VOL=SER=volser,
//             RECFM=VB,BLKSIZE=27998,LRECL=27994
```

Refer to ADARESMC in the JOBS data set for this example.

Manually Merge Sequential Command Logs in a Nucleus Cluster Environment

```
//ADARESCM  JOB
//*
//*    ADARES: MERGE SEQUENTIAL COMMAND LOGS
//*             FOR USE WITH AN ADABAS NUCLEUS CLUSTER
//*
//RES      EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD       <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DBYYYYY.ASSOR1  <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DBYYYYY.DATAR1  <=== DATA
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DBYYYYY.WORKR1  <=== WORK
//DDCLOG01  DD   DISP=SHR,DSN=EXAMPLE.DBYYYYY.CLOGR1A <=== CLOG1
//DDCLOG02  DD   DISP=SHR,DSN=EXAMPLE.DBYYYYY.CLOGR1B <=== CLOG2
//DDCLOG03  DD   DISP=SHR,DSN=EXAMPLE.DBYYYYY.CLOGR1C <=== CLOG3
//DDSIAUS1  DD   DSN=EXAMPLE.DBYYYYY.CLOGM,           <=== OUTPUT OF
//            VOL=SER=ADAXXX,UNIT=TAPE,DISP=(NEW,CATLG)       CLOG MERGE
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADARES,MODE=MULTI,SVC=XXX,DEVICE=3390,DBID=YYYYY
/*
//DDKARTE   DD   *
ADARES MERGE CLOG,NUMLOG=3
/*
//                         ↩
```

Refer to ADARESCM in the JOBS data set for this example.

Backout from a Sequential Protection Log

```
//ADARESSP  JOB
//*
//*    ADARES: BACKOUT FROM A SEQUENTIAL PLOG
//*
//RES      EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD       <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <=== WORK
//DDBACK    DD   DSN=EXAMPLE.DByyyyy.PLOG(-5),       <=== PLOG TAPE
//               UNIT=TAPE,DISP=OLD
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADARES,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADARES BACKOUT PLOGNUM=nnn
/*
```

Refer to ADARESSP in the JOBS data set for this example.

Backout from Dual/Multiple Protection Log

```
//ADARESB   JOB
//*
//*    ADARES: BACKOUT FROM DUAL/MULTIPLE PLOG
//*
//RES      EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD       <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <=== WORK
//DDPLOGR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.PLOGR1 <=== PLOG1
//DDPLOGR2  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.PLOGR2 <=== PLOG2
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADARES,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
```

```
ADARES BACKOUT DPLOG
/*
```

Refer to ADARESB in the JOBS data set for this example.

Regenerate Function

```
//ADARESR   JOB
//*
//*    ADARES: REGENERATE
//*
//RES      EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD       <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <=== WORK
//DDSIIN    DD   DSN=EXAMPLE.DByyyyy.PLOG(-5),       <=== PLOG TAPE
//            UNIT=TAPE,DISP=OLD
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADARES,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADARES REGENERATE FILE=1
/*
```

Refer to ADARESR in the JOBS data set for this example.

Repair Data Storage

```
//ADARESRP  JOB
//*
//*    ADARES:  REPAIR DATASTORAGE
//*
//RES      EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD       <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <=== WORK
//DDSIIN    DD   DSN=EXAMPLE.DByyyyy.SAVE,           <=== SAVE OUTPUT
//             DISP=OLD,UNIT=TAPE
//          DD   DSN=EXAMPLE.DByyyyy.PLOG(-5),       <=== PLOG TAPE
//             DISP=OLD,UNIT=TAPE
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
```

```
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADARES,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADARES REPAIR DSRABN=3456-3490
```

Refer to ADARESRP in the JOBS data set for this example.

XVIII ADASAV Utility: Save/Restore Database or Files

The ADASAV utility is used to save and restore the contents of the database, specific files, or an individual file to or from a sequential data set. This chapter covers the following topics:

Functional Overview
RESTONL: Restore Database from Online Source
RESTONL FILES: Restore Files to Original RABNs from Online Source
RESTONL FMOVE: Restore Files to Any RABNs from Online Source
RESTONL GCB: Restore Database Incremental from Online Source
RESTORE: Restore Database from Offline Source
RESTORE FILES: Restore Files to Original RABNs from Offline Source
RESTORE FMOVE: Restore Files to Any RABNs from Offline Source
RESTORE GCB: Restore Database Incremental from Offline Source
RESTPLOG: Restore Protection Log Only
SAVE: Save Database
SAVE FILES: Save Specified Files
JCL Requirements and Examples

162 Functional Overview

- RESTONL and RESTORE Functions 1154
- Adabas Release Support 1155

The ADASAV utility saves and restores the contents of the database, specific files, or an individual file to or from a sequential data set.

ADASAV should be run as often as required for the number and size of the files contained in the database, and depending on the amount and type of updating.

For large databases, ADASAV functions may be run in parallel for the various disk packs on which the database is contained.

Special ADASAV functions are available for use with the Adabas Delta Save Facility Facility. For more information, see the *Adabas Delta Save Facility Facility* documentation.

RESTONL and RESTORE Functions

For either RESTORE or RESTONL function operations, the Associator and Data Storage data sets must first be formatted. If either operation is interrupted, no database update activity should be attempted until the function has been successfully reexecuted.

RESTONL functions restore from a SAVE data set created while the Adabas nucleus was *active* (that is, online); RESTORE functions restore from a SAVE data set created while the Adabas nucleus was *inactive* (that is, offline).

RESTONL and RESTORE have the subfunctions GCB, FILES, and FMOVE:

- Without a subfunction, RESTONL and RESTORE restore entire databases.
- With the GCB subfunction, they restore the general control blocks (GCBs), Associator RABNs 3-30 of the database, and specified files.
- With the FILES subfunction, they restore one or more files into an existing database to their original RABNs.
- With the FMOVE subfunction, they restore one or more files into an existing database to any free space, allowing changes to extent sizes.

RESTPLOG and RESTONL Functions

If changes occurred during the online SAVE, the RESTONL function is followed automatically by the RESTPLOG function. RESTPLOG applies the updates that occurred during, and therefore were not included in, the online SAVE.

RESTPLOG is also executed following a RESTONL or RESTONL FILES function that ended before completing restoration of protection log (PLOG) updates. RESTPLOG applies the database updates not applied by the unsuccessful RESTONL function.

Online and Offline SAVEs

The SAVE function to save a database, or one or more files may be executed while the Adabas nucleus is active (online) or inactive (offline). If the Recovery Aid option is active, a SAVE database operation begins a new RLOG generation.

Adabas Release Support

You can restore entire Adabas databases only using the same Adabas release used to create the save data set. However, you can restore individual files from save data sets created by earlier Adabas versions (down to version 5.1) using the RESTORE FILES, RESTORE FMOVE, RESTONL FILES, or RESTONL FMOVE functions.

Considerations when Restoring Files from Adabas version 5

When restoring from an Adabas 5 save data set, the RESTORE FILE function discards the unused RABN chains that may be present for the normal index or upper index. This makes all blocks of these chains “unreachable index blocks” as reported by the ADACHK ICHECK function in the CHK356W message. These blocks will not be reused until they are reordered by the ADAORD REORFASSO or other reorder functions. The RESTORE FMOVE function does not discard the unused RABN chains, but rather transforms them to the new Adabas version structure.

If the database contains different device types for Data Storage and Work, restoring from an Adabas 5 save data set might be difficult if the Data Storage block size is larger than the Work block size. ADASAV may reject the restore because the maximum compressed record length of the file exceeds the length allowed by the Work block size. This is due to the increase in the length of protection record headers in later Adabas versions. To restore the file in this case, a new Work device type with a larger block size must be installed using the ADADEF NEWWORK function.

163 RESTONL: Restore Database from Online Source

- Conditions 1158
- Result 1159
- Syntax 1160
- Optional Parameters and Subparameters 1160
- Examples 1162

The RESTONL function restores a database from a database SAVE data set created while the Adabas nucleus was *active*.

Notes:

1. An interrupted RESTONL (database) operation must be reexecuted from the beginning. If the interruption occurred while RESTONL (database) was restoring the PLOG, the restore operation can be completed using the RESTPLOG function. Until successful completion or reexecution of the restore operation, the database is inaccessible.
2. If the ADASAV RESTONL (database) job control contains the DD names for DDWORKnn, these data sets are reset.

Conditions

To use the RESTONL (database) function, the following conditions must be met:

1 The correct SAVE data set must be supplied. It must have been created by an online database SAVE operation with the same version of Adabas as is used for the RESTONL.

2 The output database must have the same physical layout (device types, extent sizes) as the original database. The Associator and Data Storage data sets must be present and must have been previously formatted. The SAVE data set to be restored may have originated for this or from a different database.

3 No Adabas nucleus may be active on the output database or on a database with the DBID of the output database.

4 The protection log (PLOG) data set containing information written by the nucleus session at the time of the SAVE operation (see output of SAVE run) must be supplied. PLOG data sets from other sessions may also be included.

5 If the SAVE operation was performed with the DRIVES parameter, the SAVE data sets created can also be restored with the same DRIVES parameter. Alternatively, the SAVE data sets can be concatenated to a single SAVE data set for a RESTORE operation without the DRIVES parameter which then defaults to DRIVES=1. SAVE datasets can also be arranged in multiple concatenations for use with DRIVES values greater than one. In all cases the data sets must be in ascending order, but the sequence can be broken into separate concatenations at any point,

   For example, a SAVE taken with DRIVES=4 can be restored using DRIVES=1 by concatenating SAVE dataset 1 with 2, 3 and 4 in that order. DRIVES=2 may be used by concatenating dataset 1 with 2 and 3 with 4. When DRIVES is greater than 1, the RESTORE operation is performed from the separate DDREST*n* files in parallel.

Result

The result of this function is a database with the same physical status it had at the end of the ADASAV SAVE operation.

Syntax

```
ADASAV RESTONL  [BUFNO = { number-of-buffers | 1 } ]
                [CLOGDEV = CLOG1-device-type ]
                [DRIVES = { count | 1 } ]
                [EXCLUDE = file-list ]
                [NEWDBID = new-database-id ]
                [NEWDBNAME = new-database-name ]
                [NOUSERABEND]
                [OVERWRITE]
                [PLOGDEV = PLOG-device-type ]
                [PLOGNUM = PLOG-number [, SYN1 = PLOG-block-number ] ]
                [TEST]
```

Optional Parameters and Subparameters

BUFNO: Count of Buffers Per Drive

The BUFNO value, multiplied by the DRIVES parameter value, allocates fixed buffers for RESTONL operation. A value of 2 or 3 usually provides optimum performance; up to 255 is possible. A value greater than 5, however, provides little advantage and allocates a lot of space. The default is 1 (one buffer per drive).

CLOGDEV: Command Log Device Type

The device type of the dual/multiple command log (CLOG). This parameter is required only if the device type of the CLOG is different from that specified by the ADARUN DEVICE parameter, which is the default.

DRIVES: Tape Drives for Parallel Restore

DRIVES is the number of tape drives to be used for parallel restore processing. The number can range 1 to 8, inclusively; the default is 1.

EXCLUDE: Exclude Specified Files from Restore

EXCLUDE lists the numbers of the files to be excluded from the restore operation; that is, the files that are not to be restored. This list can include a list of more than one Adabas file number or a range of file numbers. Ranges of file numbers should be specified using a dash (-) in the format: *fnfirst-fnlast*.

The parameter is optional: if not specified, no files are excluded. A file number may be listed only once individually or in a range.

For a database restore:

- no files specified in the EXCLUDE parameter will exist in the restored database; and
- all files specified in the EXCLUDE parameter must exist on the save data set (if they are not included in a range of files).

The EXCLUDE parameter is provided for use in recovery jobs built by the Adabas Recovery Aid (ADARAI).

NEWDBID: New ID for Restored Database

NEWDBID may be used to assign a different database ID to the restored database. The ID can be in the range 1-64535; if Adabas Online System Security is installed, DBID 999 is reserved.

If NEWDBID is specified, the ADARUN DBID parameter must specify the ID of the database on the SAVE data set.

No Adabas nucleus may be active with the DBID specified on NEWDBID.

NEWDBNAME: New Name for Restored Database

NEWDBNAME assigns a new name to the restored database. If NEWDBNAME is not specified, the restored database keeps its old name.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

OVERWRITE: Overwrite Existing Database

If the restore operation is to overwrite an existing database, the OVERWRITE parameter must be specified. No Adabas nucleus may be active on the database to be overwritten.

PLOGDEV: Protection Log Device Type

The device type to be assigned to the dual/multiple protection log (PLOG). This parameter is required only if the device type of the PLOG is different from that specified by the ADARUN DEVICE parameter.

PLOGNUM: Protection Log Number

PLOGNUM specifies the number of the nucleus protection log used while the ADASAV SAVE operation was active (see output listing of the online SAVE function). Sequential protection (SIBA) logs from more than one nucleus session can be concatenated. ADASAV skips protection logs with a number lower than the PLOGNUM value. PLOGNUM is optional.

If PLOGNUM is not specified, ADASAV automatically determines the correct value from information stored in the SAVE data set.

SYN1: Beginning Block Number

SYN1 specifies the block number containing the SYN1 checkpoint at which the corresponding SAVE operation began (see output listing of the online SAVE function). This parameter is optional.

If SYN1 is not specified, ADASAV automatically determines the correct value from information stored in the SAVE data set.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

Examples

Example 1:

```
ADASAV RESTONL
```

Restore the database saved when the nucleus was active (online). The protection log number and SYN1 block number required for the restore operation are determined automatically by ADASAV.

Example 2:

```
ADASAV RESTONL
ADASAV EXCLUDE=255
ADASAV EXCLUDE=400
```

Files 255 and 400 are excluded from the restore of the database from an online-save data set.

164 RESTONL FILES: Restore Files to Original RABNs from Online Source

- Conditions 1164
- Result 1165
- Syntax 1166
- Optional Parameters and Subparameters 1167
- Examples 1173

The RESTONL FILES function restores files from a file or database SAVE data set created while the Adabas nucleus was *active*. One or more files can be restored. The files are restored into an existing database to their original RABNs.

Notes:

1. An interrupted RESTONL FILES operation must be reexecuted from the beginning. If the interruption occurred while RESTONL FILES was restoring the PLOG, the restore operation can be completed using the RESTPLOG function. Until successful completion or reexecution of the restore operation, the files to be restored are inaccessible.
2. Checkpoint and security files from Adabas version 5 cannot be restored.

Conditions

To use the RESTONL FILES function, the following conditions must be met:

1 The correct SAVE data set must be supplied. It can be a database or file SAVE data set and must contain the files to be restored. SAVE data sets from version 5.1 or above can be used.

2 A file may be restored using a SAVE tape created from a different database; however, the device types must be identical.

3 An existing database must be present. The files to be restored may have originated from this or from a different database.

4 All RABNs originally used by the file(s) to be restored must either be free (available according to the free space table) or be occupied by files to be overwritten.

5 The Adabas nucleus may be active or inactive on the output database.

If the Adabas nucleus is active for restoring the checkpoint or security files, the ADASAV utility requires exclusive database control; that is, no user may be active on the database.

6 The protection log (PLOG) data set containing information written by the nucleus session at the time of the SAVE operation (see output of SAVE run) must be supplied. PLOG data sets from other sessions may also be included. If none of the files to be restored were modified during the online SAVE operation, the protection log data set(s) can be omitted.

7 If the SAVE tape was created with Adabas version 5.1, the location of the SYN1/SYN4 checkpoint written by the Adabas nucleus at the beginning of the online SAVE operation must be specified.

8 If the SAVE operation was performed with the DRIVES parameter, the SAVE data sets created can also be restored with the same DRIVES parameter. Alternatively, the SAVE data sets can be concatenated to a single SAVE data set for a RESTORE operation without the DRIVES parameter which then defaults to DRIVES=1. SAVE datasets can also be arranged in multiple concatenations for use with DRIVES values greater than one. In all cases the data sets must

be in ascending order, but the sequence can be broken into separate concatenations at any point,

For example, a SAVE taken with DRIVES=4 can be restored using DRIVES=1 by concatenating SAVE dataset 1 with 2, 3 and 4 in that order. DRIVES=2 may be used by concatenating dataset 1 with 2 and 3 with 4. When DRIVES is greater than 1, the RESTORE operation is performed from the separate DDREST*n* files in parallel.

9 For restoring just a few files from a multivolume database SAVE data set, only those tape volumes that actually contain data of the files to be restored need to be supplied in the ADASAV job control. The job protocol of the SAVE operation as well as the corresponding SYNV checkpoints indicate the files or parts of files contained on each volume.

10 Expanded files and coupled files can only be restored or overwritten as a whole. That is, if one file in an expanded file is specified, all other files in the expanded file must be specified. If one file in a coupled relationship is specified, all other files in that relationship must be specified.

11 A checkpoint, security, trigger, or user-defined system file can be overwritten only by another checkpoint, security, trigger, or user-defined system file, respectively. A checkpoint, security, or trigger file cannot be restored if such a file already exists in the database with a different file number.

12 New file numbers can be assigned to the files to be restored using the NEWFILES parameter.

Result

The result of this function is the specified files with the same physical status they had at the end of the ADASAV SAVE operation.

Syntax

```
ADASAV RESTONL FILES = file-list   [ALLOCATION = { FORCE | NOFORCE } ]
                                   [BUFNO = { number-of-buffers | 1 } ]
                                   [DRIVES = { count | 1 } ]
                                   [EXCLUDE = file-list ]
                                   [NEWFILES = file-list ]
                                   [NOUSERABEND]
                                   [OVERWRITE]
                                   [PASSWORD = ' password-list ' ]
                                   [PLOGNUM = protection-log-number ]
                                       [ { SYN1 | SYN4 } = PLOG-block-number ]
                                   [READONLY = ' ro-file-list ' ]
                                   [RPLACTIVE = ' inactive-flag-file-list ' ]
                                   [RPLDATA = ' restore-data-to-be-sent-file-list ' ]
                                   [RPLDSBI = ' before-image-file-list ' ]
                                   [RPLKEY = ' primary-key-file-list ' ]
                                   [RPLTARGETID = ' target-ID-file-list ' ]
                                   [RPLUPDATEONLY = ' upd-only-file-list ' ]
                                   [TEST]
```

The FILES file list specifies one or more Adabas file numbers or a range of file numbers to be restored. Ranges of file numbers should be specified using a dash (-) in the format: *fnfirst-fnlast*.

If the specified file is a component file of an Adabas expanded file, all other component files of the expanded file must also be specified here. If a specified file is coupled to other files, the coupled files must also be specified.

The file list specified need not correspond to a file list used in the corresponding SAVE function. A file list may be specified even if no file list was used for the corresponding SAVE function.

A file may also be restored using a SAVE tape created from a different database; however, the device types must be identical.

Optional Parameters and Subparameters

ALLOCATION: Action to Follow File Extent Allocation Failure

ALLOCATION specifies the action to be taken if file extent allocations cannot be obtained according to the placement parameters ACRABN, DSRABN, NIRABN, or UIRABN.

ALLOCATION pertains to the implicit RABN specifications derived from the files on the save data set.

By default (that is, ALLOCATION=FORCE), the utility terminates with error if any file extent allocation cannot be met according to RABN placement parameters.

If ALLOCATION=NOFORCE is specified and any allocation with placement parameters fails, the utility retries the allocation without the placement parameter.



Note: An ADASAV RESTONL FILES operation with ALLOCATION=NOFORCE specified *cannot* be completed with RESTPLOG if the function fails after completing the restore from the save data set but before completing the restore from the protection log.

BUFNO: Count of Buffers Per Drive

The BUFNO value, multiplied by the DRIVES parameter value, allocates fixed buffers for RESTONL operation. A value of 2 or 3 usually provides optimum performance; up to 255 is possible. A value greater than 5, however, provides little advantage and allocates a lot of space. The default is 1 (one buffer per drive).

DRIVES: Tape Drives for Parallel Restore

DRIVES is the number of tape drives to be used for parallel restore processing. The number can range 1 to 8, inclusively; the default is 1.

EXCLUDE: Exclude Specified Files from Restore

EXCLUDE lists the numbers of the files to be excluded from the restore operation; that is, the files that are not to be restored. This list can include a list of more than one Adabas file number or a range of file numbers. Ranges of file numbers should be specified using a dash (-) in the format: *fnfirst-fnlast*.

The parameter is optional: if not specified, no files are excluded. A file number may be listed only once individually or in a range.

If the NEWFILES parameter:

- is *not* specified, all files specified in the EXCLUDE parameter must also be specified in the FILES parameter.
- *is* specified, all files specified in the EXCLUDE parameter must also be specified in the NEWFILES parameter. In this case, the file numbers specified in the EXCLUDE parameter refer to the new file numbers in NEWFILES, not to the old file numbers in the FILES parameter.

The EXCLUDE parameter is provided for use in recovery jobs built by the Adabas Recovery Aid (ADARAI).

NEWFILES: New File Numbers

The NEWFILES parameter specifies the new file number to be assigned to each file specified by FILES. The parameter is optional; if no new file number is assigned to a file, the file retains its original number. NEWFILES may not be specified for expanded files, physically coupled files, or replicated files.



Note: The NEWFILES parameter is not allowed if a range of files is specified in the FILES parameter.

If a file with a number specified by NEWFILES already exists in the database, the corresponding file will not be restored unless the OVERWRITE parameter is also specified. If the file to be overwritten is password-protected, the corresponding PASSWORD parameter must also be specified.

If several files are to be restored, the list of file numbers in the NEWFILES parameter must correspond to the list of files in the FILES parameter. If no new file number is to be assigned to a file, its entry in the file number list of NEWFILES must be specified as zero. See the **Examples**.

You can use NEWFILES to renumber a *base file* or *LOB file* only if both files of the *LOB file group* are restored. In this case, ADASAV assigns both files the new file numbers specified by the NEWFILES parameter and adjusts the links between the two files accordingly. However, if only one file of a *LOB file group* is restored, it cannot be assigned a new file number using the NEWFILES parameter; use the ADADBS or AOS RENUMBER function instead.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "`utility` TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

OVERWRITE: Overwrite Existing File

This parameter causes an existing file to be deleted and then restored. If a file which is to be restored is already present in the database, ADASAV will skip this file unless the OVERWRITE parameter is supplied.



Note: To avoid unintentionally overwriting the database, we recommend that you always specify the OVERWRITE parameter after, and not before, the FILES file list.

PASSWORD: File Password/Passwords

PASSWORD specifies one password or a list of passwords if one or more files in the FILES file list are password protected. This only applies to files already in the database that are to be overwritten. If the NEWFILES parameter is specified, the PASSWORD parameter must specify the passwords related to the new file numbers.

When restoring more than one password-protected file, the correct passwords must be specified as positional values corresponding to the protected file numbers' positions in the FILES list. Refer to the **Examples** for more information about the PASSWORD parameter. The Adabas nucleus must be active if password-protected files are being overwritten.

PLOGNUM: Protection Log Number

PLOGNUM specifies the number of the nucleus protection log (PLOG) used while the ADASAV SAVE operation was active (see output listing of the online SAVE function). This parameter is optional when restoring a SAVE tape created by ADASAV version 5.2 or above, or when none of the files to be restored were changed during the SAVE operation. Sequential protection (SIBA) logs from more than one nucleus session can be concatenated. ADASAV skips PLOGs with a number lower than the PLOGNUM value.

If PLOGNUM is not specified, ADASAV automatically determines the correct value from information stored in the SAVE data set.



Note: This is not possible when restoring from a version 5.1 SAVE data set.

READONLY: Read-only Status Indicator

READONLY indicates whether the read-only status is on or off for a file or a list of files. Valid values for this parameter are "YES" (read-only status is on) and "NO" (read-only status is off).

When restoring more than one file, the read-only status must be specified as positional values corresponding to the file number positions in the FILES list.

If READONLY is not specified, the read-only status of the file will be the same as it was on the SAVE data set.



Note: The READONLY parameter is not allowed if a range of files is specified in the FILES parameter.

RPLACTIVE: Reset the Replication Inactive Flag in the FCB

RPLACTIVE is an optional parameter that specifies the inactive flag setting for a file during restore processing. Valid inactive flag settings are "YES", "NO", or no setting at all. A setting of "YES" turns *off* the replication inactive flag (YFSTQRPI) for a file. A setting of "NO" turns *on* the replication inactive flag.

If no setting is specified, the default value is used. The default depends on the target database ID of the restore processing and its replication state. If *any of the following conditions* are met, the default is "YES"; otherwise the default is "NO":

- The original replication is turned off.

- The restore DBID is not the same as the original saved DBID.
- The original replication target ID has been changed.
- The original replication-before-image has been changed.
- The replication primary key has changed.
- The original replication is turned off.

When restoring more than one file, the correct RPLACTIVE settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLACTIVE parameter:

```
RPLACTIVE='YES,NO,,YES'
```

In this example, the inactive flag is turned off (YES) for the first and fourth files and turned on (NO) for the second file. No value is provided for the third file, so an default appropriate for the file is used.

RPLDATA: Send Data of the Restoring File to Replication Target ID

RPLDATA is an optional parameter that indicates whether the data in a file should be replicated to the replication target ID (RPLTARGETID parameter).

Valid replication settings are "YES", "NO", "CREATE", or no setting at all. A setting of "YES" causes the restore function to replicate the file data to the replication target during restore processing. A setting of "NO" will not replicate the data to the replication target during restore processing. A setting of CREATE causes the restore function to replicate the file data to the replication target during restore processing, but also sends a "create file" transaction to the replication target. If no setting is specified, the default "NO" is used.



Note: Values of "YES" or "CREATE" can only be specified if replication is turned on for the corresponding file.

When restoring more than one file in the FILE file list, the RPLDATA settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLACTIVE parameter:

```
RPLDATA='YES,NO,,YES'
```

In this example, the data in the first and fourth files (YES) will be replicated to the replication target, but it will not be replicated for the second and third files (NO and no setting for the third file).

RPLDSBI: Replication Data Storage Before Image

RPLDSBI is an optional parameter that indicates whether the collection of before images of data storage should occur for an update command to a file.

Valid RPLDSBI settings are "YES", "NO", or no setting at all. A setting of "YES" indicates that the collection of before images of data storage will occur for the file during restore processing. A setting of "NO" indicates that the collection of before images of data storage will *not* occur for the file during restore processing.

If no setting is specified, the default value is used. The default depends on the target database ID of the restore processing and its replication state. If the restore DBID is the same as the originally saved DBID and REPLICATION=YES in the target DBID, the default is "YES"; otherwise the default is "NO".



Note: A values of "YES" can only be specified if replication is turned on for the corresponding file.

When restoring more than one file in the FILE file list, the RPLDSBI settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLDSBI parameter:

```
RPLDSBI='YES,NO,,YES'
```

In this example, the before images are collected for the first and fourth files (YES), but are not collected for the second file (NO). No value is provided for the third file, so an default appropriate for the file is used..

RPLKEY: Primary Key for Replication

RPLKEY is an optional parameter that specifies the primary key for replication. Valid RPLKEY settings are a two-character field name, "OFF", or no setting at all. Specifying a field name identifies that field as the primary key for replication. A setting of "OFF" indicates that no primary key should be used for replication.

If no setting is specified, the default value is used. The default depends on the target database ID of the restore processing and its replication state. If the restore DBID is the same as the originally saved DBID and REPLICATION=YES in the target DBID, the original RPLKEY value for the file is used; otherwise the default is "OFF".



Note: A primary key can only be set if replication is turned on for the file and if the field name is a valid Adabas field according to the field definition table (FDT) for the file. When a new RPLKEY is specified it will not be confirmed as a valid Adabas field until the end of the ADASAV run. At that time, if any RPLKEY is found to be invalid, a warning message is issued, the RPLKEY is set to "OFF", and condition code 8 is returned.

When restoring more than one file in the FILE file list, the RPLKEY settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLKEY parameter:

```
RPLKEY='AA,BB,,OFF'
```

In this example, field AA is used as the replication primary key for the first file, BB is used as the replication primary key for the second file, and no replication primary key is used for the fourth file (OFF). No value is provided for the third file, so an default appropriate for the file is used..

RPLTARGETID: Replication Target ID

RPLTARGETID is an optional parameter that specifies the target ID of the Event Replicator Server to which the restored transactions should be sent.

Valid RPLTARGETID settings are a valid target ID, "OFF", or no setting at all. Specifying a target ID identifies that as the target for replication. A setting of "OFF" or "0" indicates that no replication target should be used for replication.

If no setting is specified, the default value is used. The default depends on the target database ID of the restore processing and its replication state. If the restore DBID is the same as the originally saved DBID and REPLICATION=YES in the target DBID, the original RPLTARGETID value for the file is used; otherwise the default is "OFF".



Note: A replication target ID can only be specified if replication is turned on for the file.

When restoring more than one file in the FILE file list, the RPLTARGETID settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLTARGETID parameter:

```
RPLTARGETID='23,24,,OFF'
```

In this example, target ID 23 is used as the replication target for the first file, 24 is used as the replication target for the second file, and no replication target is used for the fourth file (OFF). No value is provided for the third file, so an default appropriate for the file is used..

RPLUPDATEONLY: Allow Only Event Replicator Processing Updates

The `RPLUPDATEONLY` parameter can be used in the ADASAV RESTONL function to indicate whether an Adabas database file may be updated only by the Event Replicator Server as part of Adabas-to-Adabas replication or by other means as well. This parameter is optional.

Valid values are "YES" or "NO". A value of "YES" indicates that the file can only be updated via Event Replicator processing; a value of NO indicates that the file can be updated by any normal means, including Event Replicator processing.

If no value is specified, the default RPLUPDATEONLY setting of the file at the time of the corresponding SAVE operation is used.

SYN1|SYN4: Starting Block Number

The block number containing the SYN1/SYN4 checkpoint at which the restore operation is to begin (refer to the output listing of the online SAVE function for the block number). When restoring a SAVE tape created by ADASAV version 5.2 or above, this parameter is optional.

If SYN1/SYN4 is not specified, ADASAV automatically determines the correct value from information stored in the SAVE data set.

Note: This is not possible when restoring from a version 5.1 SAVE data set.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

Examples

Example 1:

```
ADASAV RESTONL FILES=3,4,5, OVERWRITE
ADASAV      PASSWORD='PWDFIL3,,PWDFIL5'
ADASAV      PLOGNUM=15,SYN1=20
```

Files 3, 4, and 5 are to be restored. Files 3 and 5 are password-protected and their passwords are PWDFIL3 and PWDFIL5. The PLOG number is 15 and the block containing the SYN1 checkpoint is 20. The old files are to be overwritten.

Example 2:

```
ADASAV RESTONL FILES=11,12,13,14,OVERWRITE
ADASAV      NEWFILES=16,0,17
```

Files 11, 12, 13, and 14 are to be restored. Files 11 and 13 are to be restored as files 16 and 17, respectively. The file numbers of files 12 and 14 will not be changed because the corresponding NEWFILES parameter values are specified as zero or omitted. Files 12, 14, 16, and 17 are to be overwritten, if already present in the database.

165 RESTONL FMOVE: Restore Files to Any RABNs from Online Source

- Conditions 1176
- Result 1177
- Syntax 1178
- Optional Parameters 1179
- Examples 1189

The RESTONL FMOVE function restores files from a file or database SAVE data set created while the Adabas nucleus was *active*. One or more files can be restored. The files are restored into an existing database to any free space. Their extent sizes may be changed.

Notes:

1. An interrupted RESTONL FMOVE operation must be reexecuted from the beginning. It is not possible to use the RESTPLOG function to recover from an interrupted RESTONL FMOVE operation that ended while restoring the PLOG. Until successful completion or reexecution of the restore operation, the files to be restored are inaccessible.
2. Checkpoint and security files from Adabas version 5 cannot be restored.

Conditions

To use the RESTONL FMOVE function, the following conditions must be met:

1 The correct SAVE data set must be supplied. It can be a database or file SAVE data set and must contain the files to be restored. SAVE data sets from Adabas version 5.1 or above can be used.

2 An existing database must be present. The files to be restored may have originated from this or from a different database.

3 The FMOVE file list specifies the file or files to be restored using new RABNs. The RABNs must be located on the same device type as used originally.

4 For the file(s) to be restored, sufficient space, either free space (according to the free space table) or space occupied by files to be overwritten, must be available in the database.

5 The Adabas nucleus may be active or inactive on the output database.

If the Adabas nucleus is active for restoring the checkpoint or security files, the ADASAV utility requires exclusive database control; that is, no user may be active on the database.

6 The protection log (PLOG) data set containing information written by the nucleus session at the time of the SAVE operation (see output of SAVE run) must be supplied. PLOG data sets from other sessions may also be included. If none of the files to be restored were modified during the online SAVE operation, the protection log data set(s) can be omitted.

7 If the SAVE tape was created with Adabas version 5.1, the location of the SYN1/SYN4 checkpoint written by the Adabas nucleus at the beginning of the online SAVE operation must be specified.

8 If the SAVE operation was performed with the DRIVES parameter, the SAVE data sets created can also be restored with the same DRIVES parameter. Alternatively, the SAVE data sets can be concatenated to a single SAVE data set for a RESTORE operation without the DRIVES parameter which then defaults to DRIVES=1. SAVE datasets can also be arranged in multiple

concatenations for use with DRIVES values greater than one. In all cases the data sets must be in ascending order, but the sequence can be broken into separate concatenations at any point,

For example, a SAVE taken with DRIVES=4 can be restored using DRIVES=1 by concatenating SAVE dataset 1 with 2, 3 and 4 in that order. DRIVES=2 may be used by concatenating dataset 1 with 2 and 3 with 4. When DRIVES is greater than 1, the RESTORE operation is performed from the separate DDREST*n* files in parallel.

9 For restoring just a few files from a multivolume database SAVE data set, only those tape volumes that actually contain data of the files to be restored need to be supplied in the ADASAV job control. The job protocol of the SAVE operation as well as the corresponding SYNV checkpoints indicate the files or parts of files contained on each volume.

10 Expanded files and coupled files can only be restored or overwritten as a whole. That is, if one file in an expanded file is specified, all other files in the expanded file must be specified. If one file in a coupled relationship is specified, all other files in that relationship must be specified.

11 A checkpoint, security, trigger, or user-defined system file can be overwritten only by another checkpoint, security, trigger, or user-defined system file, respectively. A checkpoint, security, or trigger file cannot be restored if such a file already exists in the database with a different file number.

12 New file numbers can be assigned to the files to be restored using the NEWFILES parameter.

Result

The result of this function is the specified files with the same contents they had at the end of the ADASAV SAVE operation but not necessarily in the same database blocks.

Syntax

```
ADASAV RESTONL FMOVE = file-list  [ACRABN = AC-start-rabn-list ]
                                  [AC2RABN = AC2-start-rabn-list ]
                                  [ALLOCATION = { FORCE | NOFORCE } ]
                                  [ASSOVOLUME = ' Associator-extent-volume ' ]
                                  [BUFNO = { number-of-buffers | 1 } ]
                                  [DATAVOLUME = ' Data-Storage-extent-volume ' ]
                                  [DRIVES = { count | 1 } ]
                                  [DSRABN = DS-start-rabn-list ]
                                  [DSSIZE = DS-size-list ]
                                  [EXCLUDE = file-list ]
                                  [MAXISN = isn-count-list ]
                                  [MAXISN2 = isn-count-list ]
                                  [NEWFILES = file-list ]
                                  [NIRABN = NI-start-rabn-list ]
                                  [NISIZE = NI-size-list ]
                                  [NOUSERABEND]
                                  [OVERWRITE]
                                  [PASSWORD = ' password-list ' ]
                                  [PLOGNUM = protection-log-number ]
                                      [ { SYN1 | SYN4 } = PLOG-block-number ]
                                  [READONLY = ' ro-file-list ' ]
                                  [RPLACTIVE = ' inactive-flag-file-list ' ]
                                  [RPLDATA = ' restore-data-to-be-sent-file-list ' ]
                                  [RPLDSBI = ' before-image-file-list ' ]
                                  [RPLKEY = ' primary-key-file-list ' ]
                                  [RPLTARGETID = ' target-ID-file-list ' ]
                                  [RPLUPDATEONLY = ' upd-only-file-list ' ]
                                  [TEST]
                                  [UIRABN = UI-start-rabn-list ]
                                  [UISIZE = UI-size-list ]
```

The FMOVE file list specifies one or more Adabas file numbers or a range of file numbers to be restored using new RABNs. The RABNs must be located on the same device type as used originally. Ranges of file numbers should be specified using a dash (-) in the format: `fnfirst-fnlast`.

If the specified file is a component file of an Adabas expanded file, all other component files of the expanded file must also be specified. If a specified file is coupled to other files, the coupled files must also be specified.

Optional Parameters

ACRABN: Starting Address Converter RABN/RABN List

ACRABN specifies the starting address converter RABN for each file specified by FMOVE. It can only be used in conjunction with the FMOVE parameter.

Note: The ACRABN parameter is not allowed if a range of files is specified in the FMOVE parameter.

If FMOVE is specified and ACRABN omitted, the location of the address converter is chosen by ADASAV from the free areas in the Associator that have the same device type as used originally.

If several files are to be restored, the list of RABNs in the ACRABN parameter must correspond to the list of files in the FMOVE parameter. If no ACRABN value is to be given for a file, its entry in the RABN list must be specified as zero. See the **examples**.

AC2RABN: Starting Secondary Address Converter RABN/RABN List

AC2RABN specifies the starting secondary address converter RABN for each file specified by FMOVE. It can only be used in conjunction with the FMOVE parameter.

Note: The AC2RABN parameter is not allowed if a range of files is specified in the FMOVE parameter.

If FMOVE is specified and AC2RABN omitted, the location of the secondary address converter is chosen by ADASAV from the free areas in the Associator that have the same device type as used originally. If the file contains no secondary address converter extents, this parameter is ignored.

If several files are to be restored, the list of RABNs in the AC2RABN parameter must correspond to the list of files in the FMOVE parameter. If no AC2RABN value is to be given for a file, its entry in the RABN list must be specified as zero.

ALLOCATION: Action to Follow File Extent Allocation Failure

ALLOCATION specifies the action to be taken if file extent allocations cannot be obtained according to the placement parameters ACRABN, DSRABN, NIRABN, or UIRABN.

By default (that is, ALLOCATION=FORCE), the utility terminates with error if any file extent allocation cannot be met according to RABN placement parameters.

If ALLOCATION=NOFORCE is specified and any allocation with placement parameters fails, the utility retries the allocation without the placement parameter.

ASSOVOLUME: Associator Extent Volume

Note: The value for ASSOVOLUME must be enclosed in apostrophes.

ASSOVOLUME identifies the volume on which the file's Associator space (that is, the AC, NI, and UI extents) is to be allocated. If the requested number of blocks cannot be found on the specified volume, ADASAV retries the allocation while disregarding the ASSOVOLUME parameter.

Note: The ASSOVOLUME parameter is not allowed if a range of files is specified in the FMOVE parameter.

If ACRABN, UIRABN, or NIRABN is specified, ADASAV ignores the ASSOVOLUME value when allocating the corresponding extent type. If ASSOVOLUME is not specified, the file's Associator space is allocated according to ADASAV's default allocation rules.

If several files are to be restored, the list of volumes in the ASSOVOLUME parameter must correspond to the list of files in the FMOVE parameter. If no volume is to be given for a file, its entry in the volume list must be left empty. See the **Examples** .

BUFNO: Count of Buffers Per Drive

The BUFNO value allocates fixed buffers for RESTONL operation. A value of 2 or 3 usually provides optimum performance; up to 255 is possible. A value greater than 5, however, provides little advantage and allocates a lot of space. The default is 1 (one buffer per drive).

DATAVOLUME: Data Storage Extent Volume

Note: The value for DATAVOLUME must be enclosed in apostrophes.

DATAVOLUME specifies the volume on which the file's Data Storage space (DS extents) is to be allocated. If the number of blocks requested with DSSIZE cannot be found on the specified volume, ADASAV retries the allocation while disregarding the DATAVOLUME value.

Note: The DATAVOLUME parameter is not allowed if a range of files is specified in the FMOVE parameter.

If DSRABN is specified, DATAVOLUME is ignored for the related file. If DATAVOLUME is not specified, the Data Storage space is allocated according to ADASAV's default allocation rules.

If several files are to be restored, the list of volumes in the DATAVOLUME parameter must correspond to the list of files in the FMOVE parameter. If no volume is to be given for a file, its entry in the volume list must be left empty. See the **Examples** .

DRIVES: Tape Drives for Parallel Restore

ADASAV is able to restore files from multiple save data set volumes in parallel to RABNs that are different from their original RABNs in the database. DRIVES is the number of tape drives to be used for parallel restore processing. The number can range 1 to 8, inclusively; the default is 1.

DSRABN: Starting Data Storage RABN/RABN List

DSRABN specifies the starting Data Storage RABN for each file specified by FMOVE. DSRABN can only be used in conjunction with the FMOVE parameter.

Note: The DSRABN parameter is not allowed if a range of files is specified in the FMOVE parameter.

If FMOVE is specified and DSRABN omitted, the location of the file's Data Storage is chosen by ADASAV from the free areas in Data Storage that have the same device type as used originally.

If several files are to be restored, the list of RABNs in the DSRABN parameter must correspond to the list of files in the FMOVE parameter. If no DSRABN value is specified for a file, its entry in the RABN list must be specified as zero. See the **examples** .

DSSIZE: New Data Storage Size

DSSIZE is the new size to be allocated for Data Storage for each file specified by FMOVE. It can only be used in conjunction with the FMOVE parameter.

Note: The DSSIZE parameter is not allowed if a range of files is specified in the FMOVE parameter.

The size can be specified in cylinders, or in blocks (by appending a "B" to the number). It must be at least as large as the used area of the original Data Storage.

If DSSIZE is omitted, the original Data Storage size is used.

If several files are to be restored, the list of sizes in the DSSIZE parameter must correspond to the list of files in the FMOVE parameter. If no size is to be given for a file, its entry in the size list must be specified as zero. See the **examples**.

EXCLUDE: Exclude Specified Files from Restore

EXCLUDE lists the numbers of the files to be excluded from the restore operation; that is, the files that are not to be restored. This list can include a list of more than one Adabas file number or a range of file numbers. Ranges of file numbers should be specified using a dash (-) in the format: `fnfirst-fnlast`.

The parameter is optional: if not specified, no files are excluded. A file number may be listed only once individually or in a range.

If the NEWFILES parameter:

- is *not* specified, all files specified in the EXCLUDE parameter must also be specified in the FMOVE parameter.
- *is* specified, all files specified in the EXCLUDE parameter must also be specified in the NEWFILES parameter. In this case, the file numbers specified in the EXCLUDE parameter refer to the new file numbers in NEWFILES, not to the old file numbers in the FMOVE parameter.

The EXCLUDE parameter is provided for use in recovery jobs built by the Adabas Recovery Aid (ADARAI).

MAXISN: New Maximum ISN

MAXISN is the new number of ISNs to be allocated for each file specified by FMOVE. It can only be used in conjunction with the FMOVE parameter.



Note: The MAXISN parameter is not allowed if a range of files is specified in the FMOVE parameter.

The value must be at least as large as the original highest allocated ISN (MAXISN).

If MAXISN is omitted, the original ISN count is used.

If several files are to be restored, the list of ISN counts in the MAXISN parameter must correspond to the list of files in the FMOVE parameter. If no ISN count is to be given for a file, its entry in the ISN count list must be specified as zero. See the **examples**.

If the database consists of several Associator extents with different device types, ERROR-171 may occur if MAXISN is specified and the nucleus allocated an additional address converter extent during the online save operation. If this happens, remove the MAXISN parameter for the file indicated in the error message and rerun RESTONL FMOVE.

MAXISN2: New Maximum Secondary ISN

MAXISN specifies the desired size of the secondary address converter (AC2) in ISNs. It can only be used in conjunction with the FMOVE parameter. The secondary address converter is used to map secondary ISNs of secondary spanned records to the RABNs of the Data Storage blocks where the secondary records are stored.



Note: The MAXISN2 parameter is not allowed if a range of files is specified in the FMOVE parameter.

The value must be at least as large as the original highest allocated ISN (MAXISN2).

If MAXISN2 is omitted, the original ISN count is used. If the file contains no secondary address converter extents, this parameter is ignored.

If several files are to be restored, the list of ISN counts in the MAXISN2 parameter must correspond to the list of files in the FMOVE parameter. If no ISN count is to be given for a file, its entry in the ISN count list must be specified as zero.

If the database consists of several Associator extents with different device types, ERROR-171 may occur if MAXISN2 is specified and the nucleus allocated an additional address converter extent during the online save operation. If this happens, remove the MAXISN2 parameter for the file indicated in the error message and rerun RESTONL FMOVE.

NEWFILES: New File Numbers

The NEWFILES parameter specifies the new file number to be assigned to each file specified by FMOVE. The parameter is optional: if no new file number is assigned to a file, the file retains

its original number. NEWFILES may not be specified for expanded files, physically coupled files, or replicated files.

Note: The NEWFILES parameter is not allowed if a range of files is specified in the FMOVE parameter.

If a file with a number specified by NEWFILES already exists in the database, the corresponding file will not be restored unless the OVERWRITE parameter is also specified. If the file to be overwritten is password-protected, the corresponding PASSWORD parameter must also be specified.

If several files are to be restored, the list of file numbers in the NEWFILES parameter must correspond to the list of files in the FMOVE parameter. If no new file number is to be assigned to a file, its entry in the file number list of NEWFILES must be specified as zero. See the **Examples**.

You can use NEWFILES to renumber a *base file* or *LOB file* only if both files of the *LOB file group* are restored. In this case, ADASAV assigns both files the new file numbers specified by the NEWFILES parameter and adjusts the links between the two files accordingly. However, if only one file of a *LOB file group* is restored, it cannot be assigned a new file number using the NEWFILES parameter; use the ADADBS or AOS RENUMBER function instead.

NIRABN: Starting Normal Index RABN/RABN List

NIRABN specifies the starting RABN for the normal index for each file specified by FMOVE. It can only be used in conjunction with the FMOVE parameter.

Note: The NIRABN parameter is not allowed if a range of files is specified in the FMOVE parameter.

If FMOVE is specified and NIRABN omitted, the location of the normal index is chosen by ADASAV from the free areas in the Associator that have the same device type as used originally.

If several files are to be restored, the list of RABNs in the NIRABN parameter must correspond to the list of files in the FMOVE parameter. If no NIRABN value is to be given for a file, its entry in the RABN list must be specified as zero. See the **Examples**.

NISIZE: New Size for Normal Index

NISIZE is the new size to be allocated for the normal index for each file specified by FMOVE. It can only be used in conjunction with the FMOVE parameter.

Note: The NISIZE parameter is not allowed if a range of files is specified in the FMOVE parameter.

The size can be specified in cylinders, or in blocks (by appending a "B" to the number). It must be at least as large as the used area of the original normal index.

If NISIZE is omitted, the original normal index size is used.

If several files are to be restored, the list of sizes in the NISIZE parameter must correspond to the list of files in the FMOVE parameter. If no size is to be given for a file, its entry in the size list must be specified as zero. See the **examples**.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

OVERWRITE: Overwrite Existing File

This parameter causes an existing file to be deleted and then restored. If a file which is to be restored is already present in the database, ADASAV will skip this file unless the OVERWRITE parameter is supplied.



Note: To avoid unintentionally overwriting the database, we recommend that you always specify the OVERWRITE parameter after, and not before, the FMOVE file list.

PASSWORD: Adabas Security File Password

PASSWORD specifies one password or a list of passwords if one or more files in the FILES or FMOVE file list are password-protected. This only applies to files already in the database that are to be overwritten. If the NEWFILES parameter is specified, the PASSWORD parameter must specify the passwords related to the new file numbers.

When restoring more than one password-protected file, the correct passwords must be specified as positional values corresponding to the protected file numbers' positions in the FILES or FMOVE list. Refer to the **examples** for more information about the PASSWORD parameter. When overwriting password-protected files, the Adabas nucleus must be active.

PLOGNUM: Protection Log Number

PLOGNUM specifies the number of the nucleus protection log (PLOG) used while the ADASAV SAVE operation was active (see output listing of the online SAVE function). This parameter is optional when restoring a SAVE tape created by ADASAV version 5.2 or above, or when none of the files to be restored were changed during the SAVE operation. Sequential protection (SIBA) logs from more than one nucleus session can be concatenated. ADASAV skips PLOGs with a number lower than the PLOGNUM value.

If PLOGNUM is not specified, ADASAV automatically determines the correct value from information stored in the SAVE data set.



Note: This is not possible when restoring from a version 5.1 SAVE data set.

READONLY: Read-only Status Indicator

READONLY indicates whether the read-only status is on or off for a file or a list of files. Valid values for this parameter are "YES" (read-only status is on) and "NO" (read-only status is off).

When restoring more than one file, the read-only status must be specified as positional values corresponding to the file number positions in the FILES list.

If READONLY is not specified, the read-only status of the file will be the same as it was on the SAVE data set.



Note: The READONLY parameter is not allowed if a range of files is specified in the FMOVE parameter.

RPLACTIVE: Reset the Replication Inactive Flag in the FCB

RPLACTIVE is an optional parameter that specifies the inactive flag setting for a file during restore processing. Valid inactive flag settings are "YES", "NO", or no setting at all. A setting of "YES" turns *off* the replication inactive flag (YFSTQRPI) for a file. A setting of "NO" turns *on* the replication inactive flag.

If no setting is specified, the default value is used. The default depends on the target database ID of the restore processing and its replication state. If *any of the following conditions* are met, the default is "YES"; otherwise the default is "NO":

- The original replication is turned off.
- The restore DBID is not the same as the original saved DBID.
- The original replication target ID has been changed.
- The original replication-before-image has been changed.
- The replication primary key has changed.
- The original replication is turned off.

When restoring more than one file, the correct RPLACTIVE settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLACTIVE parameter:

```
RPLACTIVE='YES,NO,,YES'
```

In this example, the inactive flag is turned off (YES) for the first and fourth files and turned on (NO) for the second file. No value is provided for the third file, so an default appropriate for the file is used.

RPLDATA: Send Data of the Restoring File to Replication Target ID

RPLDATA is an optional parameter that indicates whether the data in a file should be replicated to the replication target ID (RPLTARGETID parameter).

Valid replication settings are "YES", "NO", "CREATE", or no setting at all. A setting of "YES" causes the restore function to replicate the file data to the replication target during restore processing. A setting of "NO" will not replicate the data to the replication target during restore processing. A setting of CREATE causes the restore function to replicate the file data to the replication target during restore processing, but also sends a "create file" transaction to the replication target. If no setting is specified, the default "NO" is used.



Note: Values of "YES" or "CREATE" can only be specified if replication is turned on for the corresponding file.

When restoring more than one file in the FILE file list, the RPLDATA settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLACTIVE parameter:

```
RPLDATA='YES,NO,,YES'
```

In this example, the data in the first and fourth files (YES) will be replicated to the replication target, but it will not be replicated for the second and third files (NO and no setting for the third file).

RPLDSBI: Replication Data Storage Before Image

RPLDSBI is an optional parameter that indicates whether the collection of before images of data storage should occur for an update command to a file.

Valid RPLDSBI settings are "YES", "NO", or no setting at all. A setting of "YES" indicates that the collection of before images of data storage will occur for the file during restore processing. A setting of "NO" indicates that the collection of before images of data storage will *not* occur for the file during restore processing.

If no setting is specified, the default value is used. The default depends on the target database ID of the restore processing and its replication state. If the restore DBID is the same as the originally saved DBID and REPLICATION=YES in the target DBID, the default is "YES"; otherwise the default is "NO".



Note: A values of "YES" can only be specified if replication is turned on for the corresponding file.

When restoring more than one file in the FILE file list, the RPLDSBI settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLDSBI parameter:

```
RPLDSBI='YES,NO,,YES'
```

In this example, the before images are collected for the first and fourth files (YES), but are not collected for the second file (NO). No value is provided for the third file, so an default appropriate for the file is used..

RPLKEY: Primary Key for Replication

RPLKEY is an optional parameter that specifies the primary key for replication. Valid RPLKEY settings are a two-character field name, "OFF", or no setting at all. Specifying a field name identifies that field as the primary key for replication. A setting of "OFF" indicates that no primary key should be used for replication.

If no setting is specified, the default value is used. The default depends on the target database ID of the restore processing and its replication state. If the restore DBID is the same as the originally saved DBID and REPLICATION=YES in the target DBID, the original RPLKEY value for the file is used; otherwise the default is "OFF".



Note: A primary key can only be set if replication is turned on for the file and if the field name is a valid Adabas field according to the field definition table (FDT) for the file. When a new RPLKEY is specified it will not be confirmed as a valid Adabas field until the end of the ADASAV run. At that time, if any RPLKEY is found to be invalid, a warning message is issued, the RPLKEY is set to "OFF", and condition code 8 is returned.

When restoring more than one file in the FILE file list, the RPLKEY settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLKEY parameter:

```
RPLKEY='AA,BB,,OFF'
```

In this example, field AA is used as the replication primary key for the first file, BB is used as the replication primary key for the second file, and no replication primary key is used for the fourth file (OFF). No value is provided for the third file, so an default appropriate for the file is used..

RPLTARGETID: Replication Target ID

RPLTARGETID is an optional parameter that specifies the target ID of the Event Replicator Server to which the restored transactions should be sent.

Valid RPLTARGETID settings are a valid target ID, "OFF", or no setting at all. Specifying a target ID identifies that as the target for replication. A setting of "OFF" or "0" indicates that no replication target should be used for replication.

If no setting is specified, the default value is used. The default depends on the target database ID of the restore processing and its replication state. If the restore DBID is the same as the originally saved DBID and REPLICATION=YES in the target DBID, the original RPLTARGETID value for the file is used; otherwise the default is "OFF".



Note: A replication target ID can only be specified if replication is turned on for the file.

When restoring more than one file in the FILE file list, the RPLTARGETID settings must be specified as positional values corresponding to the file numbers' positions in the FILES list.

For example, if four files are listed in the FILES file list, the following might be the setting for the RPLTARGETID parameter:

```
RPLTARGETID='23,24,,OFF'
```

In this example, target ID 23 is used as the replication target for the first file, 24 is used as the replication target for the second file, and no replication target is used for the fourth file (OFF). No value is provided for the third file, so an default appropriate for the file is used..

RPLUPDATEONLY: Allow Only Event Replicator Processing Updates

The `RPLUPDATEONLY` parameter can be used in the ADASAV RESTONL function to indicate whether an Adabas database file may be updated only by the Event Replicator Server as part of Adabas-to-Adabas replication or by other means as well. This parameter is optional.

Valid values are "YES" or "NO". A value of "YES" indicates that the file can only be updated via Event Replicator processing; a value of NO indicates that the file can be updated by any normal means, including Event Replicator processing.

If no value is specified, the default RPLUPDATEONLY setting of the file at the time of the corresponding SAVE operation is used.

SYN1|SYN4: Beginning Block Number

The block number containing the SYN1/SYN4 checkpoint at which the restore operation is to begin (refer to the output listing of the online SAVE function for the block number). When restoring a SAVE tape created by ADASAV version 5.2 or above, this parameter is optional.

If SYN1/SYN4 is not specified, ADASAV automatically determines the correct value from information stored in the SAVE data set.



Note: This is not possible when restoring from a version 5.1 SAVE data set.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested; only the syntax of the specified parameters can be tested.

UIRABN: Starting Upper Index RABN/RABN List

UIRABN specifies the starting RABN for the upper index for each file specified by FMOVE. It can only be used in conjunction with the FMOVE parameter.



Note: The UIRABN parameter is not allowed if a range of files is specified in the FMOVE parameter.

If FMOVE is specified and UIRABN omitted, the location of the upper index is chosen by ADASAV from the free areas in the Associator that have the same device type as used originally.

If several files are to be restored, the list of RABNs in the UIRABN parameter must correspond to the list of files in the FMOVE parameter. If no UIRABN value is to be given for a file, its entry in the RABN list must be specified as zero. See the **examples**.

UISIZE: New Upper Index Size

UISIZE is the new size to be allocated for the upper index for each file specified by FMOVE. It can only be used in conjunction with the FMOVE parameter.

Note: The UISIZE parameter is not allowed if a range of files is specified in the FMOVE parameter.

The size can be specified in cylinders, or in blocks (by appending a "B" to the number). It must be at least as large as the used area of the original upper index.

If UISIZE is omitted, the original upper index size is used.

If several files are to be restored, the list of sizes in the UISIZE parameter must correspond to the list of files in the FMOVE parameter. If no size is to be given for a file, its entry in the size list must be specified as zero. See the **Examples**.

Examples

Example 1:

```
ADASAV RESTONL   PLOGNUM=25,SYN1=160,OVERWRITE
ADASAV              FMOVE=1,2
ADASAV              ACRABN=2100,2300
ADASAV              DSRABN=1500,2000
ADASAV              NIRABN=0,2380
ADASAV              UIRABN=2190
```

Protection log 25 is to be used. The block containing the SYN1 checkpoint is 160. Files 1 and 2 are to be deleted and restored. File 1 is to be restored using starting RABNs:

Address Converter	2100
Data Storage	1500
Normal Index	(chosen by ADASAV)
Upper Index	2190

File 2 is to be restored using starting RABNs:

Address Converter	2300
Data Storage	2000
Normal Index	2380
Upper Index	(chosen by ADASAV)

Example 2:

```
ADASAV RESTONL
PLOGNUM=4711,SYN4=99,FMOVE=3,4,5,OVERWRITE
ADASAV      PASSWORD='PWD3,,PWD5'
```

The files specified by the FMOVE file list may possibly be restored to different RABNs than they had before. Files 3 and 5 are password-protected and their passwords are PWD3 and PWD5.

Example 3:

```
ADASAV RESTONL FMOVE=11,12,13,14,OVERWRITE
ADASAV          NEWFILES=16,0,17
```

Files 11, 12, 13, and 14 are to be restored. Files 11 and 13 are to be restored as files 16 and 17, respectively. The file numbers of files 12 and 14 will not be changed because the corresponding NEWFILES parameter values are specified as zero or omitted. Files 12, 14, 16, and 17 are to be overwritten, if already present in the database.

166 RESTONL GCB: Restore Database Incremental from Online Source

- Conditions 1192
- Result 1193
- Syntax 1194
- Optional Parameters and Subparameters 1194
- Examples 1200

From a database SAVE data set created while the Adabas nucleus was *active,* the RESTONL GCB function restores:

- the general control blocks (GCBs);
- Associator RABNs 3-30 of the database;
- the checkpoint file;
- the security file (if present); and
- all files specified with the FILES parameter.



Notes:

1. An interrupted RESTONL GCB operation must be reexecuted from the beginning. If the interruption occurred while RESTONL GCB was restoring the PLOG, the restore operation can be completed using the RESTPLOG function. Until successful completion or reexecution of the restore operation, the database is inaccessible.
2. If the ADASAV RESTONL GCB job control contains the DD names for DDWORKnn, these data sets are reset.

Conditions

To use the RESTONL GCB function, the following conditions must be met:

1 The correct SAVE data set must be supplied. It must have been created by an online database SAVE operation with the same version of Adabas as is used for the RESTONL and must contain the file(s) to be restored.

2 The output database must have the same physical layout (device types, extent sizes) as the original database. The Associator and Data Storage data sets must be present and must have been previously formatted. The SAVE data set to be restored may have originated from this or from a different database.

3 No Adabas nucleus may be active on the output database or on a database with the DBID of the output database.

4 The protection log (PLOG) data set containing information written by the nucleus session at the time of the SAVE operation (see output of SAVE run) must be supplied. PLOG data sets from other sessions may also be included.

5 If the SAVE operation was performed with the DRIVES parameter, the SAVE data sets created can also be restored with the same DRIVES parameter. Alternatively, the SAVE data sets can be concatenated to a single SAVE data set for a RESTORE operation without the DRIVES parameter which then defaults to DRIVES=1. SAVE datasets can also be arranged in multiple concatenations for use with DRIVES values greater than one. In all cases the data sets must

be in ascending order, but the sequence can be broken into separate concatenations at any point,

For example, a SAVE taken with DRIVES=4 can be restored using DRIVES=1 by concatenating SAVE dataset 1 with 2, 3 and 4 in that order. DRIVES=2 may be used by concatenating dataset 1 with 2 and 3 with 4. When DRIVES is greater than 1, the RESTORE operation is performed from the separate DDREST*n* files in parallel.

6 For restoring just a few files from a multivolume database SAVE data set, only those tape volumes that actually contain data of the files to be restored need to be supplied in the ADASAV job control. The job protocol of the SAVE operation as well as the corresponding SYNV checkpoints indicate the files or parts of files contained on each volume.

Result

The result of this function is a database containing the specified files and the checkpoint and security files with the same physical status they had at the end of the ADASAV SAVE operation.

This operation is equivalent to a RESTONL (database), but excludes any files not specified in the FILES parameter.



Important: Any existing database in the target Associator and Data Storage data sets is completely overwritten and any files in that database are lost.

Syntax

```
ADASAV RESTONL GCB  [BUFNO = { number-of-buffers | 1 } ]
                    [CLOGDEV = CLOG1-device-type ]
                    [DRIVES = { count | 1 } ]
                    [EXCLUDE = file-list ]
                    [FILES = file-list ]
                    [NEWDBID = new-database-id ]
                    [NEWDBNAME = new-database-name ]
                    [NOUSERABEND]
                    [OVERWRITE]
                    [PLOGDEV = PLOG-device-type ]
                    [PLOGNUM = PLOG-number [, SYN1 = PLOG-block-number ] ]
                    [READONLY = ' ro-file-list ' ]
                    [RPLACTIVE = ' inactive-flag-file-list ' ]
                    [RPLDATA = ' restore-data-to-be-sent-file-list ' ]
                    [RPLDSBI = ' before-image-file-list ' ]
                    [RPLKEY = ' primary-key-file-list ' ]
                    [RPLTARGETID = ' target-ID-file-list ' ]
                    [RPLUPDATEONLY = ' upd-only-file-list ' ]
                    [TEST]
```

Optional Parameters and Subparameters

BUFNO: Count of Buffers Per Drive

The BUFNO value, multiplied by the DRIVES parameter value, allocates fixed buffers for RESTONL operation. A value of 2 or 3 usually provides optimum performance; up to 255 is possible. A value greater than 5, however, provides little advantage and allocates a lot of space. The default is 1 (one buffer per drive).

CLOGDEV: Command Log Device Type

The device type of the command log (CLOG). This parameter is required only if the device type of the CLOG is different from that specified by the ADARUN DEVICE parameter.

DRIVES: Tape Drives for Parallel Restore

DRIVES is the number of tape drives to be used for parallel restore processing. The number can range 1 to 8, inclusively; the default is 1.

EXCLUDE: Exclude Specified Files from Restore

EXCLUDE lists the numbers of the files to be excluded from the restore operation; that is, the files that are not to be restored. This list can include a list of more than one Adabas file number

or a range of file numbers. Ranges of file numbers should be specified using a dash (-) in the format: *fnfirst-fnlast*.

No files specified in the EXCLUDE parameter will exist in the restored database.

All files specified in the EXCLUDE parameter must exist on the save data set.

The parameter is optional: if not specified, no files are excluded. A file number may be listed only once individually or in a range.

The EXCLUDE parameter is provided for use in recovery jobs built by the Adabas Recovery Aid (ADARAI).

FILES: Files to Be Restored

FILES specifies one or more Adabas file numbers or a range of file numbers to be included in the database restore operation. Ranges of file numbers should be specified using a dash (-) in the format: *fnfirst-fnlast*.

If the specified file is a component file of an Adabas expanded file, all other component files of the expanded file must also be specified here. If a specified file is coupled to other files, the coupled files must also be specified. The checkpoint and security files are always restored.

NEWDBID: New ID for Restored Database

NEWDBID may be used to assign a different database ID to the restored database. The ID can be in the range 1-64535; if Adabas Online System Security is installed, DBID 999 is reserved.

If NEWDBID is specified, the ADARUN DBID parameter must specify the ID of the database on the SAVE data set.

No Adabas nucleus may be active with the DBID specified on NEWDBID.

NEWDBNAME: New Database Name

NEWDBNAME assigns a new name to the restored database. If NEWDBNAME is not specified, the restored database keeps its old name.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

OVERWRITE: Overwrite Existing Database

If the restore operation is to overwrite an existing database, the OVERWRITE parameter must be specified.

No Adabas nucleus may be active on the database to be overwritten.

PLOGDEV: Protection Log Device Type

The device type of the dual/multiple protection log (PLOG). This parameter is required only if the device type of the PLOG is different from that specified by the ADARUN DEVICE parameter.

PLOGNUM: Protection Log Number

PLOGNUM specifies the number of the nucleus protection log used while the ADASAV SAVE operation was active (see output listing of the online SAVE function). Sequential protection (SIBA) logs from more than one nucleus session can be concatenated. ADASAV skips protection logs with a number lower than the PLOGNUM value. The PLOGNUM parameter is optional.

If PLOGNUM is not specified, ADASAV automatically determines the correct value from information stored in the SAVE data set.

READONLY: Read-only Status Indicator

READONLY indicates whether the read-only status is on or off for a file or a list of files. Valid values for this parameter are "YES" (read-only status is on) and "NO" (read-only status is off).

When restoring more than one file, the read-only status must be specified as positional values corresponding to the file number positions in the FILES list.

If READONLY is not specified, the read-only status of the file will be the same as it was on the SAVE data set.

RPLACTIVE: Reset the Replication Inactive Flag in the FCB

RPLACTIVE is an optional parameter that specifies the inactive flag setting for a file during restore processing. Valid inactive flag settings are "YES", "NO", or no setting at all. A setting of "YES" turns *off* the replication inactive flag (YFSTQRPI) for a file. A setting of "NO" turns *on* the replication inactive flag.

If no setting is specified, the default value is used. The default depends on the target database ID of the restore processing and its replication state. If *any of the following conditions* are met, the default is "YES"; otherwise the default is "NO":

- The original replication is turned off.
- The restore DBID is not the same as the original saved DBID.
- The original replication target ID has been changed.
- The original replication-before-image has been changed.
- The replication primary key has changed.
- The original replication is turned off.

When restoring more than one file, the correct RPLACTIVE settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLACTIVE parameter:

```
RPLACTIVE='YES,NO,,YES'
```

In this example, the inactive flag is turned off (YES) for the first and fourth files and turned on (NO) for the second file. No value is provided for the third file, so an default appropriate for the file is used.

RPLDATA: Send Data of the Restoring File to Replication Target ID

RPLDATA is an optional parameter that indicates whether the data in a file should be replicated to the replication target ID (RPLTARGETID parameter).

Valid replication settings are "YES", "NO", "CREATE", or no setting at all. A setting of "YES" causes the restore function to replicate the file data to the replication target during restore processing. A setting of "NO" will not replicate the data to the replication target during restore processing. A setting of CREATE causes the restore function to replicate the file data to the replication target during restore processing, but also sends a "create file" transaction to the replication target. If no setting is specified, the default "NO" is used.



Note: Values of "YES" or "CREATE" can only be specified if replication is turned on for the corresponding file.

When restoring more than one file in the FILE file list, the RPLDATA settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLACTIVE parameter:

```
RPLDATA='YES,NO,,YES'
```

In this example, the data in the first and fourth files (YES) will be replicated to the replication target, but it will not be replicated for the second and third files (NO and no setting for the third file).

RPLDSBI: Replication Data Storage Before Image

RPLDSBI is an optional parameter that indicates whether the collection of before images of data storage should occur for an update command to a file.

Valid RPLDSBI settings are "YES", "NO", or no setting at all. A setting of "YES" indicates that the collection of before images of data storage will occur for the file during restore processing. A setting of "NO" indicates that the collection of before images of data storage will *not* occur for the file during restore processing.

If no setting is specified, the default value is used. The default depends on the target database ID of the restore processing and its replication state. If the restore DBID is the same as the originally saved DBID and REPLICATION=YES in the target DBID, the default is "YES"; otherwise the default is "NO".



Note: A values of "YES" can only be specified if replication is turned on for the corresponding file.

When restoring more than one file in the FILE file list, the RPLDSBI settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLDSBI parameter:

```
RPLDSBI='YES,NO,,YES'
```

In this example, the before images are collected for the first and fourth files (YES), but are not collected for the second file (NO). No value is provided for the third file, so an default appropriate for the file is used..

RPLKEY: Primary Key for Replication

RPLKEY is an optional parameter that specifies the primary key for replication. Valid RPLKEY settings are a two-character field name, "OFF", or no setting at all. Specifying a field name identifies that field as the primary key for replication. A setting of "OFF" indicates that no primary key should be used for replication.

If no setting is specified, the default value is used. The default depends on the target database ID of the restore processing and its replication state. If the restore DBID is the same as the originally saved DBID and REPLICATION=YES in the target DBID, the original RPLKEY value for the file is used; otherwise the default is "OFF".



Note: A primary key can only be set if replication is turned on for the file and if the field name is a valid Adabas field according to the field definition table (FDT) for the file. When a new RPLKEY is specified it will not be confirmed as a valid Adabas field until the end of the ADASAV run. At that time, if any RPLKEY is found to be invalid, a warning message is issued, the RPLKEY is set to "OFF", and condition code 8 is returned.

When restoring more than one file in the FILE file list, the RPLKEY settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLKEY parameter:

```
RPLKEY='AA,BB,,OFF'
```

In this example, field AA is used as the replication primary key for the first file, BB is used as the replication primary key for the second file, and no replication primary key is used for the fourth file (OFF). No value is provided for the third file, so an default appropriate for the file is used..

RPLTARGETID: Replication Target ID

RPLTARGETID is an optional parameter that specifies the target ID of the Event Replicator Server to which the restored transactions should be sent.

Valid RPLTARGETID settings are a valid target ID, "OFF", or no setting at all. Specifying a target ID identifies that as the target for replication. A setting of "OFF" or "0" indicates that no replication target should be used for replication.

If no setting is specified, the default value is used. The default depends on the target database ID of the restore processing and its replication state. If the restore DBID is the same as the originally saved DBID and REPLICATION=YES in the target DBID, the original RPLTARGETID value for the file is used; otherwise the default is "OFF".



Note: A replication target ID can only be specified if replication is turned on for the file.

When restoring more than one file in the FILE file list, the RPLTARGETID settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLTARGETID parameter:

```
RPLTARGETID='23,24,,OFF'
```

In this example, target ID 23 is used as the replication target for the first file, 24 is used as the replication target for the second file, and no replication target is used for the fourth file (OFF). No value is provided for the third file, so an default appropriate for the file is used..

RPLUPDATEONLY: Allow Only Event Replicator Processing Updates

The `RPLUPDATEONLY` parameter can be used in the ADASAV RESTONL function to indicate whether an Adabas database file may be updated only by the Event Replicator Server as part of Adabas-to-Adabas replication or by other means as well. This parameter is optional.

Valid values are "YES" or "NO". A value of "YES" indicates that the file can only be updated via Event Replicator processing; a value of NO indicates that the file can be updated by any normal means, including Event Replicator processing.

If no value is specified, the default RPLUPDATEONLY setting of the file at the time of the corresponding SAVE operation is used.

SYN1: Beginning Block Number

SYN1 specifies the protection log block number containing the SYN1 checkpoint at which the corresponding SAVE operation began (see output listing of the online SAVE function). This parameter is optional.

If SYN1 is not specified, ADASAV automatically determines the correct value from information stored in the SAVE data set.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

Examples

Example 1:

```
ADASAV RESTONL GCB
```

Restore the database GCBs saved when the nucleus was active (online). The checkpoint and security files are also restored. ADASAV determines the protection log number and SYN1 block number required for the restore operation automatically.

Example 2:

```
ADASAV RESTONL   GCB,FILES=3,4,5,OVERWRITE
ADASAV          PLOGNUM=15,SYN1=20
```

Files 3, 4 and 5 as well as the checkpoint and security files are restored. The protection log number is 15 and the block containing the SYN1 checkpoint is 20. The old database is to be overwritten.

167 RESTORE: Restore Database from Offline Source

- Conditions 1202
- Result 1203
- Syntax 1204
- Optional Parameters 1204
- Examples 1206

The RESTORE function restores a database from a database SAVE data set created while the Adabas nucleus was *inactive*.

Notes:

1. An interrupted RESTORE (database) operation must be reexecuted from the beginning. Until successful completion or reexecution of the restore operation, the database is inaccessible.
2. If the ADASAV RESTORE (database) job control contains the DD names for DDWORKnn, these data sets are reset.

Conditions

To use the RESTORE (database) function, the following conditions must be met:

1 The correct SAVE data set must be supplied. It must have been created by an offline database SAVE operation with the same version of Adabas as is used for the RESTORE.

2 The output database must have the same physical layout (device types, extent sizes) as the original database. The Associator and Data Storage data sets must be present and must have been previously formatted. The SAVE data set to be restored may have originated for this or from a different database.

3 No Adabas nucleus may be active on the output database or on a database with the DBID of the output database.

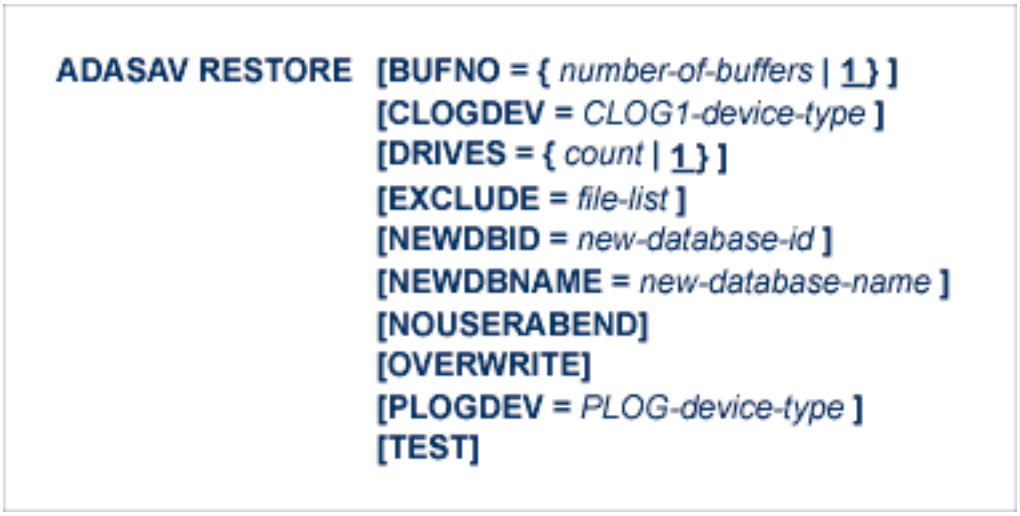
4 If the SAVE operation was performed with the DRIVES parameter, the SAVE data sets created can also be restored with the same DRIVES parameter. Alternatively, the SAVE data sets can be concatenated to a single SAVE data set for a RESTORE operation without the DRIVES parameter which then defaults to DRIVES=1. SAVE datasets can also be arranged in multiple concatenations for use with DRIVES values greater than one. In all cases the data sets must be in ascending order, but the sequence can be broken into separate concatenations at any point,

   For example, a SAVE taken with DRIVES=4 can be restored using DRIVES=1 by concatenating SAVE dataset 1 with 2, 3 and 4 in that order. DRIVES=2 may be used by concatenating dataset 1 with 2 and 3 with 4. When DRIVES is greater than 1, the RESTORE operation is performed from the separate DDREST*n* files in parallel.

Result

The result of this function is a database with the same physical status it had at the time of the ADASAV SAVE operation.

Syntax



Optional Parameters

BUFNO: Count of Buffers Per Drive

The BUFNO value, multiplied by the DRIVES parameter value, allocates fixed buffers for RESTORE operation. A value of 2 or 3 usually provides optimum performance; up to 255 is possible. A value greater than 5, however, provides little advantage and allocates a lot of space. The default is 1 (one buffer per drive).

CLOGDEV: Command Log Device Type

The device type to be assigned to the dual/multiple command log (CLOG). This parameter is required only if the device type to be used for the CLOG is different from that specified by the ADARUN DEVICE parameter.

DRIVES: Tape Drives for Parallel Restore

DRIVES is the number of tape drives to be used for parallel restore processing. The number can range 1 to 8, inclusively; the default is 1.

EXCLUDE: Exclude Specified Files from Restore

EXCLUDE lists the numbers of the files to be excluded from the restore operation; that is, the files that are not to be restored. This list can include a list of more than one Adabas file number or a range of file numbers. Ranges of file numbers should be specified using a dash (-) in the format: *fnfirst-fnlast*.

For a database restore:

- no files specified in the EXCLUDE parameter will exist in the restored database; and

- all files specified in the EXCLUDE parameter must exist on the save data set (if they are not included in a range of files).

The parameter is optional: if not specified, no files are excluded. A file number may be listed only once individually or in a range.

The EXCLUDE parameter is provided for use in recovery jobs built by the Adabas Recovery Aid (ADARAI).

NEWDBID: New ID for Restored Database

NEWDBID may be used to assign a different database ID to the restored database. The ID can be in the range 1-64535; if Adabas Online System Security is installed, DBID 999 is reserved.

If NEWDBID is specified, the ADARUN DBID parameter must specify the ID of the database on the SAVE data set.

No Adabas nucleus may be active with the DBID specified on NEWDBID.

NEWDBNAME: New Database Name

NEWDBNAME assigns a new name to the restored database. If NEWDBNAME is not specified, the restored database keeps its old name.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

OVERWRITE: Overwrite Existing Database

If the restore operation is to overwrite an existing database, the OVERWRITE parameter must be specified.

No Adabas nucleus may be active on the database to be overwritten.

PLOGDEV: Protection Log Device Type

The device type of the dual/multiple protection log (PLOG). This parameter is required only if the device type of the PLOG is different from that specified by the ADARUN DEVICE parameter.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

Examples

Example 1:

```
ADASAV RESTORE OVERWRITE
```

A database is to be restored. An existing database might be overwritten.

Example 2:

```
ADASAV RESTORE EXCLUDE=10,11,12
```

Files 10 through 12 are excluded from the restore of the database from an offline-save data set.

168 RESTORE FILES: Restore Files to Original RABNs from Offline Source

- Conditions 1208
- Result 1209
- Syntax 1210
- Optional Parameters 1211
- Examples 1216

The RESTORE FILES function restores files from a file or database SAVE data set created while the Adabas nucleus was *inactive*, or from a file SAVE data set created with UTYPE=EXU. One or more files can be restored. The files are restored into an existing database to their original RABNs.

Notes:

1. An interrupted RESTORE FILES operation must be reexecuted from the beginning. Until successful completion or reexecution of the restore operation, the files to be restored are inaccessible.
2. Checkpoint and security files from Adabas version 5 cannot be restored.

Conditions

To use the RESTORE FILES function, the following conditions must be met:

1 The correct SAVE data set must be supplied. It can be a database or file SAVE data set and must contain the files to be restored.

2 A file may be restored using a SAVE data set created using a different database as long as identical device types are used.

3 An existing database must be present. The files to be restored may have originated from this or from a different database. SAVE data sets from Adabas version 5.1 or above can be used.

4 All RABNs originally used by the file(s) to be restored must either be free (available according to the Free Space Table) or be occupied by files to be overwritten.

5 The Adabas nucleus may be active or inactive on the output database.

If the Adabas nucleus is active for restoring the checkpoint or security files, the ADASAV utility requires exclusive database control; that is, no user may be active on the database.

6 If the SAVE operation was performed with the DRIVES parameter, the SAVE data sets created can also be restored with the same DRIVES parameter. Alternatively, the SAVE data sets can be concatenated to a single SAVE data set for a RESTORE operation without the DRIVES parameter which then defaults to DRIVES=1. SAVE datasets can also be arranged in multiple concatenations for use with DRIVES values greater than one. In all cases the data sets must be in ascending order, but the sequence can be broken into separate concatenations at any point,

For example, a SAVE taken with DRIVES=4 can be restored using DRIVES=1 by concatenating SAVE dataset 1 with 2, 3 and 4 in that order. DRIVES=2 may be used by concatenating dataset 1 with 2 and 3 with 4. When DRIVES is greater than 1, the RESTORE operation is performed from the separate DDREST*n* files in parallel.

7 For restoring just a few files from a multivolume database SAVE data set, only those tape volumes that actually contain data of the files to be restored need to be supplied in the

ADASAV job control. The job protocol of the SAVE operation as well as the corresponding SYNV checkpoints indicate the files or parts of files contained on each volume.

8 Expanded files and coupled files can only be restored or overwritten as a whole. That is, if one file in an expanded file is specified, all other files in the expanded file must be specified. If one file in a coupled relationship is specified, all other files in that relationship must be specified.

9 A checkpoint, security, trigger, or user-defined system file can be overwritten only by another checkpoint, security, trigger, or user-defined system file, respectively. A checkpoint, security, or trigger file cannot be restored if such a file already exists in the database with a different file number.

10 New file numbers can be assigned to the files to be restored using the NEWFILES parameter.

Result

The result of this function is the specified files with the same physical status they had at the time of the ADASAV SAVE operation.

Syntax

```
ADASAV RESTORE FILES = file-list   [ALLOCATION = { FORCE | NOFORCE } ]
                                   [BUFNO = { number-of-buffers | 1 } ]
                                   [DRIVES = { count | 1 } ]
                                   [EXCLUDE = file-list ]
                                   [NEWFILES = file-list ]
                                   [NOUSERABEND]
                                   [OVERWRITE]
                                   [PASSWORD = ' password-list ' ]
                                   [READONLY = ' ro-file-list ' ]
                                   [RPLACTIVE = ' inactive-flag-file-list ' ]
                                   [RPLDATA = ' restore-data-to-be-sent-file-list ' ]
                                   [RPLDSBI = ' before-image-file-list ' ]
                                   [RPLKEY = ' primary-key-file-list ' ]
                                   [RPLTARGETID = ' target-ID-file-list ' ]
                                   [RPLUPDATEONLY = ' upd-only-file-list ' ]
                                   [TEST]
```

The FILES file list specifies one or more Adabas file numbers or a range of file numbers to be restored. Ranges of file numbers should be specified using a dash (-) in the format: *fnfirst-fnlast*.

For an Adabas expanded file, all component files of the expanded file including the anchor file must be specified. If a specified file is coupled to other files, the coupled files must also be specified.

The file list specified need not correspond to the file list used for the corresponding SAVE function. A file list may be specified even if no file list was used for the corresponding SAVE function.

A file may also be restored using a SAVE data set created using a different database as long as identical device types are used.

Optional Parameters

ALLOCATION: Action to Follow File Extent Allocation Failure

ALLOCATION specifies the action to be taken if file extent allocations cannot be obtained according to the placement parameters ACRABN, DSRABN, NIRABN, or UIRABN.

ALLOCATION pertains to the implicit RABN specifications derived from the files on the save data set.

By default (that is, ALLOCATION=FORCE), the utility terminates with error if any file extent allocation cannot be met according to RABN placement parameters.

If ALLOCATION=NOFORCE is specified and any allocation with placement parameters fails, the utility retries the allocation without the placement parameter.

BUFNO: Count of Buffers Per Drive

The BUFNO value, multiplied by the DRIVES parameter value, allocates fixed buffers for RESTORE operation. A value of 2 or 3 usually provides optimum performance; up to 255 is possible. A value greater than 5, however, provides little advantage and allocates a lot of space. The default is 1 (one buffer per drive).

DRIVES: Tape Drives for Parallel Restore

DRIVES is the number of tape drives to be used for parallel restore processing. The number can range 1 to 8, inclusively; the default is 1.

EXCLUDE: Exclude Specified Files from Restore

EXCLUDE lists the numbers of the files to be excluded from the restore operation; that is, the files that are not to be restored. This list can include a list of more than one Adabas file number or a range of file numbers. Ranges of file numbers should be specified using a dash (-) in the format: *fnfirst-fnlast*.

The parameter is optional: if not specified, no files are excluded. A file number may be listed only once individually or in a range.

If the NEWFILES parameter:

- is *not* specified, all files specified in the EXCLUDE parameter must also be specified in the FILES parameter.
- *is* specified, all files specified in the EXCLUDE parameter must also be specified in the NEWFILES parameter. In this case, the file numbers specified in the EXCLUDE parameter refer to the new file numbers in NEWFILES, not to the old file numbers in the FILES parameter.

The EXCLUDE parameter is provided for use in recovery jobs built by the Adabas Recovery Aid (ADARAI).

NEWFILES: New File Numbers

The NEWFILES parameter specifies the new file number to be assigned to each file specified by FILES. The parameter is optional: if no new file number is assigned to a file, the file retains its original number. NEWFILES may not be specified for expanded files, physically coupled files, or replicated files.

Note: The NEWFILES parameter is not allowed if a range of files is specified in the FILES parameter.

If a file with a number specified by NEWFILES already exists in the database, the corresponding file will not be restored unless the OVERWRITE parameter is also specified. If the file to be overwritten is password-protected, the corresponding PASSWORD parameter must also be specified.

If several files are to be restored, the list of file numbers in the NEWFILES parameter must correspond to the list of files in the FILES parameter. If no new file number is to be assigned to a file, its entry in the file number list of NEWFILES must be specified as zero. See the **examples**.

You can use NEWFILES to renumber a *base file* or *LOB file* only if both files of the *LOB file group* are restored. In this case, ADASAV assigns both files the new file numbers specified by the NEWFILES parameter and adjusts the links between the two files accordingly. However, if only one file of a *LOB file group* is restored, it cannot be assigned a new file number using the NEWFILES parameter; use the ADADBS or AOS RENUMBER function instead.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

OVERWRITE: Overwrite Existing File

This parameter causes an existing file to be deleted and then restored. If a file which is to be restored is already present in the database, ADASAV will skip this file unless the OVERWRITE parameter is supplied.

Note: To avoid unintentionally overwriting the database, we recommend that you always specify the OVERWRITE parameter after, and not before, the FILES file list.

PASSWORD

PASSWORD specifies one password or a list of passwords if one or more files specified in FILES are password-protected. This only applies to files already in the database which are to

be overwritten. If the NEWFILES parameter is specified, the PASSWORD parameter must specify the passwords related to the new file numbers.

When restoring more than one password-protected file, the correct passwords must be specified as positional values corresponding to the protected file numbers' positions in the FILES list. Refer to the **examples** for more information about the PASSWORD parameter. When overwriting password-protected files, the Adabas nucleus must be active.

READONLY: Read-only Status Indicator

READONLY indicates whether the read-only status is on or off for a file or a list of files. Valid values for this parameter are "YES" (read-only status is on) and "NO" (read-only status is off).

When restoring more than one file, the read-only status must be specified as positional values corresponding to the file number positions in the FILES list.

If READONLY is not specified, the read-only status of the file will be the same as it was on the SAVE data set.



Note: The READONLY parameter is not allowed if a range of files is specified in the FMOVE parameter.

RPLACTIVE: Reset the Replication Inactive Flag in the FCB

RPLACTIVE is an optional parameter that specifies the inactive flag setting for a file during restore processing. Valid inactive flag settings are "YES", "NO", or no setting at all. A setting of "YES" turns *off* the replication inactive flag (YFSTQRPI) for a file. A setting of "NO" turns *on* the replication inactive flag.

If no setting is specified, the default value is used. The default depends on the target database ID of the restore processing and its replication state. If *any of the following conditions* are met, the default is "YES"; otherwise the default is "NO":

- The original replication is turned off.
- The restore DBID is not the same as the original saved DBID.
- The original replication target ID has been changed.
- The original replication-before-image has been changed.
- The replication primary key has changed.
- The original replication is turned off.

When restoring more than one file, the correct RPLACTIVE settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLACTIVE parameter:

```
RPLACTIVE='YES,NO,,YES'
```

In this example, the inactive flag is turned off (YES) for the first and fourth files and turned on (NO) for the second file. No value is provided for the third file, so an default appropriate for the file is used.

RPLDATA: Send Data of the Restoring File to Replication Target ID

RPLDATA is an optional parameter that indicates whether the data in a file should be replicated to the replication target ID (RPLTARGETID parameter).

Valid replication settings are "YES", "NO", "CREATE", or no setting at all. A setting of "YES" causes the restore function to replicate the file data to the replication target during restore processing. A setting of "NO" will not replicate the data to the replication target during restore processing. A setting of CREATE causes the restore function to replicate the file data to the replication target during restore processing, but also sends a "create file" transaction to the replication target. If no setting is specified, the default "NO" is used.



Note: Values of "YES" or "CREATE" can only be specified if replication is turned on for the corresponding file.

When restoring more than one file in the FILE file list, the RPLDATA settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLACTIVE parameter:

```
RPLDATA='YES,NO,,YES'
```

In this example, the data in the first and fourth files (YES) will be replicated to the replication target, but it will not be replicated for the second and third files (NO and no setting for the third file).

ADASAV RESTORE with the RPLDATA parameter will not replicate data for a file with spanned DS record support.

RPLDSBI: Replication Data Storage Before Image

RPLDSBI is an optional parameter that indicates whether the collection of before images of data storage should occur for an update command to a file.

Valid RPLDSBI settings are "YES", "NO", or no setting at all. A setting of "YES" indicates that the collection of before images of data storage will occur for the file during restore processing. A setting of "NO" indicates that the collection of before images of data storage will *not* occur for the file during restore processing.

If no setting is specified, the default value is used. The default depends on the target database ID of the restore processing and its replication state. If the restore DBID is the same as the originally saved DBID and REPLICATION=YES in the target DBID, the default is "YES"; otherwise the default is "NO".



Note: A values of "YES" can only be specified if replication is turned on for the corresponding file.

When restoring more than one file in the FILE file list, the RPLDSBI settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLDSBI parameter:

```
RPLDSBI='YES,NO,,YES'
```

In this example, the before images are collected for the first and fourth files (YES), but are not collected for the second file (NO). No value is provided for the third file, so an default appropriate for the file is used..

RPLKEY: Primary Key for Replication

RPLKEY is an optional parameter that specifies the primary key for replication. Valid RPLKEY settings are a two-character field name, "OFF", or no setting at all. Specifying a field name identifies that field as the primary key for replication. A setting of "OFF" indicates that no primary key should be used for replication.

If no setting is specified, the default value is used. The default depends on the target database ID of the restore processing and its replication state. If the restore DBID is the same as the originally saved DBID and REPLICATION=YES in the target DBID, the original RPLKEY value for the file is used; otherwise the default is "OFF".



Note: A primary key can only be set if replication is turned on for the file and if the field name is a valid Adabas field according to the field definition table (FDT) for the file. When a new RPLKEY is specified it will not be confirmed as a valid Adabas field until the end of the ADASAV run. At that time, if any RPLKEY is found to be invalid, a warning message is issued, the RPLKEY is set to "OFF", and condition code 8 is returned.

When restoring more than one file in the FILE file list, the RPLKEY settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLKEY parameter:

```
RPLKEY='AA,BB,,OFF'
```

In this example, field AA is used as the replication primary key for the first file, BB is used as the replication primary key for the second file, and no replication primary key is used for the fourth file (OFF). No value is provided for the third file, so an default appropriate for the file is used..

RPLTARGETID: Replication Target ID

RPLTARGETID is an optional parameter that specifies the target ID of the Event Replicator Server to which the restored transactions should be sent.

Valid RPLTARGETID settings are a valid target ID, "OFF", or no setting at all. Specifying a target ID identifies that as the target for replication. A setting of "OFF" or "0" indicates that no replication target should be used for replication.

If no setting is specified, the default value is used. The default depends on the target database ID of the restore processing and its replication state. If the restore DBID is the same as the originally saved DBID and REPLICATION=YES in the target DBID, the original RPLTARGETID value for the file is used; otherwise the default is "OFF".



Note: A replication target ID can only be specified if replication is turned on for the file.

When restoring more than one file in the FILE file list, the RPLTARGETID settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLTARGETID parameter:

```
RPLTARGETID='23,24,,OFF'
```

In this example, target ID 23 is used as the replication target for the first file, 24 is used as the replication target for the second file, and no replication target is used for the fourth file (OFF). No value is provided for the third file, so an default appropriate for the file is used..

RPLUPDATEONLY: Allow Only Event Replicator Processing Updates

The `RPLUPDATEONLY` parameter can be used in the ADASAV RESTORE function to indicate whether an Adabas database file may be updated only by the Event Replicator Server as part of Adabas-to-Adabas replication or by other means as well. This parameter is optional.

Valid values are "YES" or "NO". A value of "YES" indicates that the file can only be updated via Event Replicator processing; a value of NO indicates that the file can be updated by any normal means, including Event Replicator processing.

If no value is specified, the default RPLUPDATEONLY setting of the file at the time of the corresponding SAVE operation is used.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

Examples

Example 1:

```
ADASAV RESTORE   FILES=3,4,5,OVERWRITE,
ADASAV      PASSWORD='PWD3,,PWD5'
```

Files 3, 4, and 5 are to be restored. Existing files 3, 4, and 5 are to be overwritten by the restored files. Passwords PWD3 and PWD5 are provided for files 3 and 5.

Example 2:

```
ADASAV RESTORE FILES=11,12,13,14,OVERWRITE
ADASAV          NEWFILES=16,0,17
```

Files 11, 12, 13, and 14 are to be restored. Files 11 and 13 are to be restored as files 16 and 17, respectively. The file numbers of files 12 and 14 will not be changed because the corresponding NEWFILES parameter values are specified as zero or omitted. Files 12, 14, 16, and 17 are to be overwritten, if already present in the database.

169 RESTORE FMOVE: Restore Files to Any RABNs from Offline Source

- Conditions 1220
- Result 1221
- Syntax 1221
- Optional Parameters 1223
- Examples 1232

The RESTORE FMOVE function restores files from a file or database SAVE data set created while the Adabas nucleus was *inactive*, or from a file SAVE data set created with UTYPE=EXU. One or more files can be restored. The files are restored into an existing database to any free space. Their extent sizes may be changed.

Notes:

1. An interrupted RESTORE FMOVE operation must be reexecuted from the beginning. Until successful completion or reexecution of the restore operation, the files to be restored are inaccessible.
2. Checkpoint and security files from Adabas version 5 cannot be restored.

Conditions

To use the RESTORE FMOVE function, the following conditions must be met:

1 The correct SAVE data set must be supplied. It can be a database or file SAVE data set and must contain the files to be restored. SAVE data sets from Adabas version 5.1 or above can be used.

2 An existing database must be present. The files to be restored may have originated from this or from a different database.

3 The FMOVE file list specifies a file or files to be restored using new RABNs (and sizes). The RABNs must be located on the same device type as used originally for the respective files. Files can be restored into other than the original database as long as device types are identical.

4 For the file(s) to be restored, sufficient space, either free space (according to the free space table) or space occupied by files to be overwritten, must be available in the database.

5 The Adabas nucleus may be active or inactive on the output database.

If the Adabas nucleus is active for restoring the checkpoint or security files, the ADASAV utility requires exclusive database control; that is, no user may be active on the database.

6 If the SAVE operation was performed with the DRIVES parameter, the SAVE data sets created can also be restored with the same DRIVES parameter. Alternatively, the SAVE data sets can be concatenated to a single SAVE data set for a RESTORE operation without the DRIVES parameter which then defaults to DRIVES=1. SAVE datasets can also be arranged in multiple concatenations for use with DRIVES values greater than one. In all cases the data sets must be in ascending order, but the sequence can be broken into separate concatenations at any point,

For example, a SAVE taken with DRIVES=4 can be restored using DRIVES=1 by concatenating SAVE dataset 1 with 2, 3 and 4 in that order. DRIVES=2 may be used by concatenating dataset

1 with 2 and 3 with 4. When DRIVES is greater than 1, the RESTORE operation is performed from the separate DDREST*n* files in parallel.

7 For restoring just a few files from a multivolume database SAVE data set, only those tape volumes that actually contain data of the files to be restored need to be supplied in the ADASAV job control. The job protocol of the SAVE operation as well as the corresponding SYNV checkpoints indicate the files or parts of files contained on each volume.

8 Expanded files and coupled files can only be restored or overwritten as a whole. That is, if one file in an expanded file is specified, all other files in the expanded file must be specified. If one file in a coupled relationship is specified, all other files in that relationship must be specified.

9 A checkpoint, security, trigger, or user-defined system file can be overwritten only by another checkpoint, security, trigger, or user-defined system file, respectively. A checkpoint, security, or trigger file cannot be restored if such a file already exists in the database with a different file number.

10 New file numbers can be assigned to the files to be restored using the NEWFILES parameter.

Result

The result of this function is the specified files with the same contents they had at the time of the ADASAV SAVE operation but not necessarily in the same database blocks.

Syntax

The FMOVE file list specifies one or more Adabas file numbers or a range of file numbers to be restored using new RABNs (and sizes). Ranges of file numbers should be specified using a dash (-) in the format: *fnfirst-fnlast*.

For an Adabas expanded file, all component files of the expanded file including the anchor file must be specified. If a specified file is coupled to other files, the coupled files must also be specified.

The RABNs must be located on the same device type as used originally for the respective files. Files can be restored into other than the original database as long as device types are identical.

```
ADASAV RESTORE FMOVE = file-list  [ACRABN = AC-start-rabn-list ]
                                  [AC2RABN = AC2-start-rabn-list ]
                                  [ALLOCATION = { FORCE | NOFORCE } ]
                                  [ASSOVOLUME = ' Associator-extent-volume ' ]
                                  [BUFNO = { number-of-buffers | 1 } ]
                                  [DATAVOLUME = ' Data-Storage-extent-volume ' ]
                                  [DRIVES = { count | 1 } ]
                                  [DSRABN = DS-start-rabn-list ]
                                  [DSSIZE = DS-size-list ]
                                  [EXCLUDE = file-list ]
                                  [MAXISN = isn-count-list ]
                                  [MAXISN2 = isn-count-list ]
                                  [NEWFILES = file-list ]
                                  [NIRABN = NI-start-rabn-list ]
                                  [NISIZE = NI-size-list ]
                                  [NOUSERABEND]
                                  [OVERWRITE]
                                  [PASSWORD = ' password-list ' ]
                                  [READONLY = ' ro-file-list ' ]
                                  [RPLACTIVE = ' inactive-flag-file-list ' ]
                                  [RPLDATA = ' restore-data-to-be-sent-file-list ' ]
                                  [RPLDSBI = ' before-image-file-list ' ]
                                  [RPLKEY = ' primary-key-file-list ' ]
                                  [RPLTARGETID = ' target-ID-file-list ' ]
                                  [RPLUPDATEONLY = ' upd-only-file-list ' ]
                                  [TEST]
                                  [UIRABN = UI-start-rabn-list ]
                                  [UISIZE = UI-size-list ]
```

The FMOVE file list specifies one or more Adabas file numbers or a range of file numbers to be restored using new RABNs. The RABNs must be located on the same device type as used originally. Ranges of file numbers should be specified using a dash (-) in the format: *fnfirst-fnlast*.

If the specified file is a component file of an Adabas expanded file, all other component files of the expanded file must also be specified. If a specified file is coupled to other files, the coupled files must also be specified.

Optional Parameters

ACRABN: Starting Address Converter RABN/RABN List

ACRABN specifies the starting address converter RABN for each file specified by FMOVE. It can only be used in conjunction with the FMOVE parameter.

Note: The ACRABN parameter is not allowed if a range of files is specified in the FMOVE parameter.

If FMOVE is specified and ACRABN omitted, the location of the address converter is chosen by ADASAV from the free areas in the Associator that have the same device type as used originally.

If several files are to be restored, the list of RABNs in the ACRABN parameter must correspond to the list of files in the FMOVE parameter. If no ACRABN value is to be given for a file, its entry in the RABN list must be specified as zero. See the **examples** .

AC2RABN: Starting Secondary Address Converter RABN/RABN List

AC2RABN specifies the starting secondary address converter RABN for each file specified by FMOVE. It can only be used in conjunction with the FMOVE parameter.

Note: The AC2RABN parameter is not allowed if a range of files is specified in the FMOVE parameter.

If FMOVE is specified and AC2RABN omitted, the location of the secondary address converter is chosen by ADASAV from the free areas in the Associator that have the same device type as used originally. If the file contains no secondary address converter extents, this parameter is ignored.

If several files are to be restored, the list of RABNs in the AC2RABN parameter must correspond to the list of files in the FMOVE parameter. If no AC2RABN value is to be given for a file, its entry in the RABN list must be specified as zero.

ALLOCATION: Action to Follow File Extent Allocation Failure

ALLOCATION specifies the action to be taken if file extent allocations cannot be obtained according to the placement parameters ACRABN, DSRABN, NIRABN, or UIRABN.

By default (that is, ALLOCATION=FORCE), the utility terminates with error if any file extent allocation cannot be met according to RABN placement parameters.

If ALLOCATION=NOFORCE is specified and any allocation with placement parameters fails, the utility retries the allocation without the placement parameter.

ASSOVOLUME: Associator Extent Volume

Note: The value for ASSOVOLUME must be enclosed in apostrophes.

ASSOVOLUME identifies the volume on which the file's Associator space (that is, the AC, NI, and UI extents) is to be allocated. If the requested number of blocks cannot be found on the specified volume, ADASAV retries the allocation while disregarding the ASSOVOLUME parameter.



Note: The ASSOVOLUME parameter is not allowed if a range of files is specified in the FMOVE parameter.

If ACRABN, UIRABN, or NIRABN is specified, ADASAV ignores the ASSOVOLUME value when allocating the corresponding extent type. If ASSOVOLUME is not specified, the file's Associator space is allocated according to ADASAV's default allocation rules.

If several files are to be restored, the list of volumes in the ASSOVOLUME parameter must correspond to the list of files in the FMOVE parameter. If no volume is to be given for a file, its entry in the volume list must be left empty. See the **examples** .

BUFNO: Count of Buffers

The BUFNO value allocates fixed buffers for RESTORE operation. A value of 2 or 3 usually provides optimum performance; up to 255 is possible. A value greater than 5, however, provides little advantage and allocates a lot of space. The default is 1 (one buffer per drive).

DATAVOLUME: Data Storage Extent Volume



Note: The value for DATAVOLUME must be enclosed in apostrophes.

DATAVOLUME specifies the volume on which the file's Data Storage space (DS extents) is to be allocated. If the number of blocks requested with DSSIZE cannot be found on the specified volume, ADASAV retries the allocation while disregarding the DATAVOLUME value.



Note: The DATAVOLUME parameter is not allowed if a range of files is specified in the FMOVE parameter.

If DSRABN is specified, DATAVOLUME is ignored for the related file. If DATAVOLUME is not specified, the Data Storage space is allocated according to ADASAV's default allocation rules.

If several files are to be restored, the list of volumes in the DATAVOLUME parameter must correspond to the list of files in the FMOVE parameter. If no volume is to be given for a file, its entry in the volume list must be left empty. See the **examples**.

DRIVES: Tape Drives for Parallel Restore

ADASAV is able to restore files from multiple save data set volumes in parallel to RABNs that are different from their original RABNs in the database. DRIVES is the number of tape drives to be used for parallel restore processing. The number can range 1 to 8, inclusively; the default is 1.

DSRABN: Starting Data Storage RABN/RABN List

DSRABN specifies the starting Data Storage RABN for each file specified by FMOVE. DSRABN can only be used in conjunction with the FMOVE parameter.

Note: The DSRABN parameter is not allowed if a range of files is specified in the FMOVE parameter.

If FMOVE is specified and DSRABN omitted, the location of the file's Data Storage is chosen by ADASAV from the free areas in Data Storage that have the same device type as used originally.

If several files are to be restored, the list of RABNs in the DSRABN parameter must correspond to the list of files in the FMOVE parameter. If no DSRABN value is specified for a file, its entry in the RABN list must be specified as zero. See the **examples**.

DSSIZE: New Data Storage Size

DSSIZE is the new size to be allocated for Data Storage for each file specified by FMOVE. It can only be used in conjunction with the FMOVE parameter.

Note: The DSSIZE parameter is not allowed if a range of files is specified in the FMOVE parameter.

The size can be specified in cylinders, or in blocks (by appending a "B" to the number). It must be at least as large as the used area of the original Data Storage.

If DSSIZE is omitted, the original Data Storage size is used.

If several files are to be restored, the list of sizes in the DSSIZE parameter must correspond to the list of files in the FMOVE parameter. If no size is to be given for a file, its entry in the size list must be specified as zero. See the **examples**.

EXCLUDE: Exclude Specified Files from Restore

If specified, EXCLUDE lists the numbers of the files to be excluded from the restore operation; that is, the files that are not to be restored. This list can include a list of more than one Adabas file number or a range of file numbers. Ranges of file numbers should be specified using a dash (-) in the format: *fnfirst-fnlast*.

The parameter is optional: if not specified, no files are excluded. A file number may be listed only once individually or in a range.

If the NEWFILES parameter:

- is *not* specified, all files specified in the EXCLUDE parameter must also be specified in the FMOVE parameter.
- *is* specified, all files specified in the EXCLUDE parameter must also be specified in the NEWFILES parameter. In this case, the file numbers specified in the EXCLUDE parameter refer to the new file numbers in NEWFILES, not to the old file numbers in the FMOVE parameter.

The EXCLUDE parameter is provided for use in recovery jobs built by the Adabas Recovery Aid (ADARAI).

MAXISN: New Maximum ISN

MAXISN is the new number of ISNs to be allocated for each file specified by FMOVE. It can only be used in conjunction with the FMOVE parameter. The value must be at least as large as the original highest allocated ISN (MAXISN).



Note: The MAXISN parameter is not allowed if a range of files is specified in the FMOVE parameter.

If MAXISN is omitted, the original ISN count is used.

If several files are to be restored, the list of ISN counts in the MAXISN parameter must correspond to the list of files in the FMOVE parameter. If no ISN count is to be given for a file, its entry in the ISN count list must be specified as zero. See the **examples**.

MAXISN2: New Maximum Secondary ISN

MAXISN2 specifies the desired size of the secondary address converter (AC2) in ISNs. It can only be used in conjunction with the FMOVE parameter. The secondary address converter is used to map the secondary ISNs of secondary spanned records to the RABNs of the Data Storage blocks where the secondary records are stored.



Note: The MAXISN2 parameter is not allowed if a range of files is specified in the FMOVE parameter.

The value must be at least as large as the original highest allocated ISN (MAXISN2).

If MAXISN2 is omitted, the original ISN count is used. If the file contains no secondary address converter extents, this parameter is ignored.

If several files are to be restored, the list of ISN counts in the MAXISN2 parameter must correspond to the list of files in the FMOVE parameter. If no ISN count is to be given for a file, its entry in the ISN count list must be specified as zero.

If the database consists of several Associator extents with different device types, ERROR-171 may occur if MAXISN2 is specified and the nucleus allocated an additional address converter extent during the online save operations. If this happens remove the MAXISN2 parameter for the file indicated in the error message and rerun RESTONL FMOVE.

NEWFILES: New File Numbers

The NEWFILES parameter specifies the new file number to be assigned to each file specified by FMOVE. The parameter is optional: if no new file number is assigned to a file, the file retains its original number. NEWFILES may not be specified for expanded files, physically coupled files, or replicated files.



Note: The NEWFILES parameter is not allowed if a range of files is specified in the FMOVE parameter.

If a file with a number specified by NEWFILES already exists in the database, the corresponding file will not be restored unless the OVERWRITE parameter is also specified. If the file to be overwritten is password-protected, the corresponding PASSWORD parameter must also be specified.

If several files are to be restored, the list of file numbers in the NEWFILES parameter must correspond to the list of files in the FMOVE parameter. If no new file number is to be assigned to a file, its entry in the file number list of NEWFILES must be specified as zero. See the **examples** .

You can use NEWFILES to renumber a *base file* or *LOB file* only if both files of the *LOB file group* are restored. In this case, ADASAV assigns both files the new file numbers specified by the NEWFILES parameter and adjusts the links between the two files accordingly. However, if only one file of a *LOB file group* is restored, it cannot be assigned a new file number using the NEWFILES parameter; use the ADADBS or AOS RENUMBER function instead.

NIRABN: Starting Normal Index RABN/RABN List

NIRABN specifies the starting RABN for the normal index for each file specified by FMOVE. It can only be used in conjunction with the FMOVE parameter.

Note: The NIRABN parameter is not allowed if a range of files is specified in the FMOVE parameter.

If FMOVE is specified and NIRABN omitted, the location of the normal index is chosen by ADASAV from the free areas in the Associator that have the same device type as used originally.

If several files are to be restored, the list of RABNs in the NIRABN parameter must correspond to the list of files in the FMOVE parameter. If no NIRABN value is to be given for a file, its entry in the RABN list must be specified as zero. See the **examples** .

NISIZE: New Size for Normal Index

NISIZE is the new size to be allocated for the normal index for each file specified by FMOVE. It can only be used in conjunction with the FMOVE parameter.

Note: The NISIZE parameter is not allowed if a range of files is specified in the FMOVE parameter.

The size can be specified in cylinders, or in blocks (by appending a "B" to the number). It must be at least as large as the used area of the original normal index.

If NISIZE is omitted, the original normal index size is used.

If several files are to be restored, the list of sizes in the NISIZE parameter must correspond to the list of files in the FMOVE parameter. If no size is to be given for a file, its entry in the size list must be specified as zero. See the **examples** .

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

OVERWRITE: Overwrite Existing File

This parameter causes an existing file to be deleted and then restored. If a file which is to be restored is already present in the database, ADASAV will skip this file unless the OVERWRITE parameter is supplied.



Note: To avoid unintentionally overwriting the database, we recommend that you always specify the OVERWRITE parameter after, and not before, the FMOVE file list.

PASSWORD: Adabas Security File Password

PASSWORD specifies one password or a list of passwords if one or more files in the FILES or FMOVE file list are password-protected. This only applies to files already in the database that are to be overwritten. If the NEWFILES parameter is specified, the PASSWORD parameter must specify the passwords related to the new file numbers.

When restoring more than one password-protected file, the correct passwords must be specified as positional values corresponding to the positions of the protected file numbers in the FILES or FMOVE list. Refer to the **examples** for more information about the PASSWORD parameter. When overwriting password-protected files, the Adabas nucleus must be active.

READONLY: Read-only Status Indicator

READONLY indicates whether the read-only status is on or off for a file or a list of files. Valid values for this parameter are "YES" (read-only status is on) and "NO" (read-only status is off).

When restoring more than one file, the read-only status must be specified as positional values corresponding to the file number positions in the FILES list.

If READONLY is not specified, the read-only status of the file will be the same as it was on the SAVE data set.



Note: The READONLY parameter is not allowed if a range of files is specified in the FMOVE parameter.

RPLACTIVE: Reset the Replication Inactive Flag in the FCB

RPLACTIVE is an optional parameter that specifies the inactive flag setting for a file during restore processing. Valid inactive flag settings are "YES", "NO", or no setting at all. A setting

of "YES" turns *off* the replication inactive flag (YFSTQRPI) for a file. A setting of "NO" turns *on* the replication inactive flag.

If no setting is specified, the default value is used. The default depends on the target database ID of the restore processing and its replication state. If *any of the following conditions* are met, the default is "YES"; otherwise the default is "NO":

- The original replication is turned off.
- The restore DBID is not the same as the original saved DBID.
- The original replication target ID has been changed.
- The original replication-before-image has been changed.
- The replication primary key has changed.
- The original replication is turned off.

When restoring more than one file, the correct RPLACTIVE settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLACTIVE parameter:

```
RPLACTIVE='YES,NO,,YES'
```

In this example, the inactive flag is turned off (YES) for the first and fourth files and turned on (NO) for the second file. No value is provided for the third file, so an default appropriate for the file is used.

RPLDATA: Send Data of the Restoring File to Replication Target ID

RPLDATA is an optional parameter that indicates whether the data in a file should be replicated to the replication target ID (RPLTARGETID parameter).

Valid replication settings are "YES", "NO", "CREATE", or no setting at all. A setting of "YES" causes the restore function to replicate the file data to the replication target during restore processing. A setting of "NO" will not replicate the data to the replication target during restore processing. A setting of CREATE causes the restore function to replicate the file data to the replication target during restore processing, but also sends a "create file" transaction to the replication target. If no setting is specified, the default "NO" is used.



Note: Values of "YES" or "CREATE" can only be specified if replication is turned on for the corresponding file.

When restoring more than one file in the FILE file list, the RPLDATA settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLACTIVE parameter:

```
RPLDATA='YES,NO,,YES'
```

In this example, the data in the first and fourth files (YES) will be replicated to the replication target, but it will not be replicated for the second and third files (NO and no setting for the third file).

RPLDSBI: Replication Data Storage Before Image

RPLDSBI is an optional parameter that indicates whether the collection of before images of data storage should occur for an update command to a file.

Valid RPLDSBI settings are "YES", "NO", or no setting at all. A setting of "YES" indicates that the collection of before images of data storage will occur for the file during restore processing. A setting of "NO" indicates that the collection of before images of data storage will *not* occur for the file during restore processing.

If no setting is specified, the default value is used. The default depends on the target database ID of the restore processing and its replication state. If the restore DBID is the same as the originally saved DBID and REPLICATION=YES in the target DBID, the default is "YES"; otherwise the default is "NO".



Note: A values of "YES" can only be specified if replication is turned on for the corresponding file.

When restoring more than one file in the FILE file list, the RPLDSBI settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLDSBI parameter:

```
RPLDSBI='YES,NO,,YES'
```

In this example, the before images are collected for the first and fourth files (YES), but are not collected for the second file (NO). No value is provided for the third file, so an default appropriate for the file is used..

RPLKEY: Primary Key for Replication

RPLKEY is an optional parameter that specifies the primary key for replication. Valid RPLKEY settings are a two-character field name, "OFF", or no setting at all. Specifying a field name identifies that field as the primary key for replication. A setting of "OFF" indicates that no primary key should be used for replication.

If no setting is specified, the default value is used. The default depends on the target database ID of the restore processing and its replication state. If the restore DBID is the same as the originally saved DBID and REPLICATION=YES in the target DBID, the original RPLKEY value for the file is used; otherwise the default is "OFF".



Note: A primary key can only be set if replication is turned on for the file and if the field name is a valid Adabas field according to the field definition table (FDT) for the file. When a new RPLKEY is specified it will not be confirmed as a valid Adabas field until the end of the ADASAV run. At that time, if any RPLKEY is found to be invalid, a warning message is issued, the RPLKEY is set to "OFF", and condition code 8 is returned.

When restoring more than one file in the FILE file list, the RPLKEY settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLKEY parameter:

```
RPLKEY='AA,BB,,OFF'
```

In this example, field AA is used as the replication primary key for the first file, BB is used as the replication primary key for the second file, and no replication primary key is used for the fourth file (OFF). No value is provided for the third file, so an default appropriate for the file is used..

RPLTARGETID: Replication Target ID

RPLTARGETID is an optional parameter that specifies the target ID of the Event Replicator Server to which the restored transactions should be sent.

Valid RPLTARGETID settings are a valid target ID, "OFF", or no setting at all. Specifying a target ID identifies that as the target for replication. A setting of "OFF" or "0" indicates that no replication target should be used for replication.

If no setting is specified, the default value is used. The default depends on the target database ID of the restore processing and its replication state. If the restore DBID is the same as the originally saved DBID and REPLICATION=YES in the target DBID, the original RPLTARGETID value for the file is used; otherwise the default is "OFF".

When restoring more than one file in the FILE file list, the RPLTARGETID settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLTARGETID parameter:

```
RPLTARGETID='23,24,,OFF'
```

In this example, target ID 23 is used as the replication target for the first file, 24 is used as the replication target for the second file, and no replication target is used for the fourth file (OFF). No value is provided for the third file, so an default appropriate for the file is used..

RPLUPDATEONLY: Allow Only Event Replicator Processing Updates

The `RPLUPDATEONLY` parameter can be used in the ADASAV RESTORE function to indicate whether an Adabas database file may be updated only by the Event Replicator Server as part of Adabas-to-Adabas replication or by other means as well. This parameter is optional.

Valid values are "YES" or "NO". A value of "YES" indicates that the file can only be updated via Event Replicator processing; a value of NO indicates that the file can be updated by any normal means, including Event Replicator processing.

If no value is specified, the default RPLUPDATEONLY setting of the file at the time of the corresponding SAVE operation is used.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

UIRABN: Starting Upper Index RABN/RABN List

UIRABN specifies the starting RABN for the upper index for each file specified by FMOVE. It can only be used in conjunction with the FMOVE parameter.

Note: The UIRABN parameter is not allowed if a range of files is specified in the FMOVE parameter.

If FMOVE is specified and UIRABN omitted, the location of the upper index is chosen by ADASAV from the free areas in the Associator that have the same device type as used originally.

If several files are to be restored, the list of RABNs in the UIRABN parameter must correspond to the list of files in the FMOVE parameter. If no UIRABN value is to be given for a file, its entry in the RABN list must be specified as zero. See the **examples**.

UISIZE: New Upper Index Size

UISIZE is the new size to be allocated for the upper index for each file specified by FMOVE. It can only be used in conjunction with the FMOVE parameter.

Note: The UISIZE parameter is not allowed if a range of files is specified in the FMOVE parameter.

The size can be specified in cylinders, or in blocks (by appending a "B" to the number). It must be at least as large as the used area of the original upper index.

If UISIZE is omitted, the original upper index size is used.

If several files are to be restored, the list of sizes in the UISIZE parameter must correspond to the list of files in the FMOVE parameter. If no size is to be given for a file, its entry in the size list must be specified as zero. See the **examples**.

Examples

Example 1:

```
ADASAV RESTORE FMOVE=4,6, ACRABN=0,3820,
MAXISN=0,2000000
```

Three tape drives are available for parallel RESTORE processing. Files 4 and 6 are to be restored with new RABNs. The space allocation for file 4 is to be done using original sizes.

The address converter for file 6 is to begin at Associator RABN 3820, and the value for the file's MAXISN is to be increased to 2,000,000.

Example 2:

```
ADASAV RESTORE FMOVE=3,4,5, OVERWRITE
ADASAV        PASSWORD='PWD3,,PWD5'
```

Files 3, 4 and 5 are to be restored. If they already exist in the database, they are overwritten. Passwords are provided for files 3 and 5 to allow them to be overwritten. All original size values are used. The files might be restored to other than the original RABNs.

Example 3:

```
ADASAV RESTORE   FMOVE=1,2
ADASAV          FMOVE=3,4
```

Restore files 1 through 4.

Example 4:

```
ADASAV RESTORE FMOVE=11,12,13,14,OVERWRITE
ADASAV          NEWFILES=16,0,17
```

Files 11, 12, 13, and 14 are to be restored. Files 11 and 13 are to be restored as files 16 and 17, respectively. The file numbers of files 12 and 14 will not be changed because the corresponding NEWFILES parameter values are specified as zero or omitted. Files 12, 14, 16, and 17 are to be overwritten, if already present in the database.

170 RESTORE GCB: Restore Database Incremental from Offline Source

- Conditions 1236
- Result 1237
- Syntax 1238
- Optional Parameters 1238
- Examples 1243

From a database SAVE data set created while the Adabas nucleus was *inactive,* the RESTORE GCB function restores

- the general control blocks (GCBs);
- Associator RABNs 3-30 of the database;
- the checkpoint file;
- the security file (if present); and
- all files specified with the FILES parameter.

Notes:

1. An interrupted RESTORE GCB operation must be reexecuted from the beginning. Until successful completion or reexecution of the restore operation, the database is inaccessible.
2. If the ADASAV RESTORE GCB job control contains the DD names for DDWORKnn, these data sets are reset.

Conditions

To use the RESTORE GCB function, the following conditions must be met:

1 The correct SAVE data set must be supplied. It must have been created by an offline database SAVE operation with the same version of Adabas as is used for the RESTORE and must contain the file(s) to be restored.

2 The output database must have the same physical layout (device types, extent sizes) as the original database. The Associator and Data Storage data sets must be present and must have been previously formatted. The SAVE data set to be restored may have originated for this or from a different database.

3 No Adabas nucleus may be active on the output database or on a database with the DBID of the output database.

4 If the SAVE operation was performed with the DRIVES parameter, the SAVE data sets created can also be restored with the same DRIVES parameter. Alternatively, the SAVE data sets can be concatenated to a single SAVE data set for a RESTORE operation without the DRIVES parameter which then defaults to DRIVES=1. SAVE datasets can also be arranged in multiple concatenations for use with DRIVES values greater than one. In all cases the data sets must be in ascending order, but the sequence can be broken into separate concatenations at any point,

   For example, a SAVE taken with DRIVES=4 can be restored using DRIVES=1 by concatenating SAVE dataset 1 with 2, 3 and 4 in that order. DRIVES=2 may be used by concatenating dataset

1 with 2 and 3 with 4. When DRIVES is greater than 1, the RESTORE operation is performed from the separate DDREST*n* files in parallel.

5 For restoring just a few files from a multivolume database SAVE data set, only those tape volumes that actually contain data of the files to be restored need to be supplied in the ADASAV job control. The job protocol of the SAVE operation as well as the corresponding SYNV checkpoints indicate the files or parts of files contained on each volume.

Result

The result of this function is a database containing the specified files and the checkpoint and security files with the same physical status they had at the time of the ADASAV SAVE operation.

This operation is equivalent to a RESTORE (database), but excludes any files not specified in the FILES parameter.



Important: Any existing database in the target Associator and Data Storage data sets is completely overwritten and any files in that database are lost.

Syntax

```
ADASAV RESTORE GCB  [BUFNO = { number-of-buffers | 1 } ]
                    [CLOGDEV = CLOG1-device-type ]
                    [DRIVES = { count | 1 } ]
                    [EXCLUDE = file-list ]
                    [FILES = file-list ]
                    [NEWDBID = new-database-id ]
                    [NEWDBNAME = new-database-name ]
                    [NOUSERABEND]
                    [OVERWRITE]
                    [PLOGDEV = PLOG-device-type ]
                    [READONLY = ' ro-file-list ' ]
                    [RPLACTIVE = ' inactive-flag-file-list ' ]
                    [RPLDATA = ' restore-data-to-be-sent-file-list ' ]
                    [RPLDSBI = ' before-image-file-list ' ]
                    [RPLKEY = ' primary-key-file-list ' ]
                    [RPLTARGETID = ' target-ID-file-list ' ]
                    [RPLUPDATEONLY = ' upd-only-file-list ' ]
                    [TEST]
```

Optional Parameters

BUFNO: Count of Buffers Per Drive

The BUFNO value, multiplied by the DRIVES parameter value, allocates fixed buffers for RESTORE operation. A value of 2 or 3 usually provides optimum performance; up to 255 is possible. A value greater than 5, however, provides little advantage and allocates a lot of space. The default is 1 (one buffer per drive).

CLOGDEV: Command Log Device Type

The device type of the command log (CLOG). This parameter is required only if the device type of the CLOG is different from that specified by the ADARUN DEVICE parameter.

DRIVES: Tape Drives for Parallel Restore

DRIVES is the number of tape drives to be used for parallel restore processing. The number can range 1 to 8, inclusively; the default is 1.

EXCLUDE: Exclude Specified Files from Restore

EXCLUDE lists the numbers of the files to be excluded from the restore operation; that is, the files that are not to be restored. This list can include a list of more than one Adabas file number or a range of file numbers. Ranges of file numbers should be specified using a dash (-) in the format: *fnfirst-fnlast*.

No files specified in the EXCLUDE parameter will exist in the restored database.

All files specified in the EXCLUDE parameter must exist on the save data se (if they are not included in a range of files).

The parameter is optional: if not specified, no files are excluded. A file number may be listed only once individually or in a range.

The EXCLUDE parameter is provided for use in recovery jobs built by the Adabas Recovery Aid (ADARAI).

FILES: Files to Be Restored

FILES specifies one or more Adabas file numbers or a range of file numbers to be included in the database restore operation. Ranges of file numbers should be specified using a dash (-) in the format: *fnfirst-fnlast*.

If the specified file is a component file of an Adabas expanded file, all other component files of the expanded file must also be specified here. If a specified file is coupled to other files, the coupled files must also be specified. The checkpoint and security files are always restored.

NEWDBID: New ID for Restored Database

NEWDBID may be used to assign a different database ID to the restored database. The ID can be in the range 1-64535; if Adabas Online System Security is installed, DBID 999 is reserved.

If NEWDBID is specified, the ADARUN DBID parameter must specify the ID of the database on the SAVE data set.

No Adabas nucleus may be active with the DBID specified on NEWDBID.

NEWDBNAME: New Database Name

NEWDBNAME assigns a new name to the restored database. If NEWDBNAME is not specified, the restored database keeps its old name.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

OVERWRITE: Overwrite Existing Database

If the restore operation is to overwrite an existing database, the OVERWRITE parameter must be specified.

No Adabas nucleus may be active on the database to be overwritten.

PLOGDEV: Protection Log Device Type

The device type of the dual/multiple protection log (PLOG). This parameter is required only if the device type of the PLOG is different from that specified by the ADARUN DEVICE parameter.

READONLY: Read-only Status Indicator

READONLY indicates whether the read-only status is on or off for a file or a list of files. Valid values for this parameter are "YES" (read-only status is on) and "NO" (read-only status is off).

When restoring more than one file, the read-only status must be specified as positional values corresponding to the file number positions in the FILES list.

If READONLY is not specified, the read-only status of the file will be the same as it was on the SAVE data set.



Note: The READONLY parameter is not allowed if a range of files is specified in the FILES parameter.

RPLACTIVE: Reset the Replication Inactive Flag in the FCB

RPLACTIVE is an optional parameter that specifies the inactive flag setting for a file during restore processing. Valid inactive flag settings are "YES", "NO", or no setting at all. A setting of "YES" turns *off* the replication inactive flag (YFSTQRPI) for a file. A setting of "NO" turns *on* the replication inactive flag.

If no setting is specified, the default value is used. The default depends on the target database ID of the restore processing and its replication state. If *any of the following conditions* are met, the default is "YES"; otherwise the default is "NO":

- The original replication is turned off.
- The restore DBID is not the same as the original saved DBID.
- The original replication target ID has been changed.
- The original replication-before-image has been changed.
- The replication primary key has changed.
- The original replication is turned off.

When restoring more than one file, the correct RPLACTIVE settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLACTIVE parameter:

```
RPLACTIVE='YES,NO,,YES'
```

In this example, the inactive flag is turned off (YES) for the first and fourth files and turned on (NO) for the second file. No value is provided for the third file, so an default appropriate for the file is used.

RPLDATA: Send Data of the Restoring File to Replication Target ID

RPLDATA is an optional parameter that indicates whether the data in a file should be replicated to the replication target ID (RPLTARGETID parameter).

Valid replication settings are "YES", "NO", "CREATE", or no setting at all. A setting of "YES" causes the restore function to replicate the file data to the replication target during restore processing. A setting of "NO" will not replicate the data to the replication target during restore processing. A setting of CREATE causes the restore function to replicate the file data to the replication target during restore processing, but also sends a "create file" transaction to the replication target. If no setting is specified, the default "NO" is used.



Note: Values of "YES" or "CREATE" can only be specified if replication is turned on for the corresponding file.

When restoring more than one file in the FILE file list, the RPLDATA settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLACTIVE parameter:

```
RPLDATA='YES,NO,,YES'
```

In this example, the data in the first and fourth files (YES) will be replicated to the replication target, but it will not be replicated for the second and third files (NO and no setting for the third file).

RPLDSBI: Replication Data Storage Before Image

RPLDSBI is an optional parameter that indicates whether the collection of before images of data storage should occur for an update command to a file.

Valid RPLDSBI settings are "YES", "NO", or no setting at all. A setting of "YES" indicates that the collection of before images of data storage will occur for the file during restore processing. A setting of "NO" indicates that the collection of before images of data storage will *not* occur for the file during restore processing.

If no setting is specified, the default value is used. The default depends on the target database ID of the restore processing and its replication state. If the restore DBID is the same as the originally saved DBID and REPLICATION=YES in the target DBID, the default is "YES"; otherwise the default is "NO".



Note: A values of "YES" can only be specified if replication is turned on for the corresponding file.

When restoring more than one file in the FILE file list, the RPLDSBI settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example,

if four files are listed in the FILES file list, the following might be the setting for the RPLDSBI parameter:

```
RPLDSBI='YES,NO,,YES'
```

In this example, the before images are collected for the first and fourth files (YES), but are not collected for the second file (NO). No value is provided for the third file, so an default appropriate for the file is used..

RPLKEY: Primary Key for Replication

RPLKEY is an optional parameter that specifies the primary key for replication.

Valid RPLKEY settings are a two-character field name, "OFF", or no setting at all. Specifying a field name identifies that field as the primary key for replication. A setting of "OFF" indicates that no primary key should be used for replication.

If no setting is specified, the default value is used. The default depends on the target database ID of the restore processing and its replication state. If the restore DBID is the same as the originally saved DBID and REPLICATION=YES in the target DBID, the original RPLKEY value for the file is used; otherwise the default is "OFF".



Note: A primary key can only be set if replication is turned on for the file and if the field name is a valid Adabas field according to the field definition table (FDT) for the file. When a new RPLKEY is specified it will not be confirmed as a valid Adabas field until the end of the ADASAV run. At that time, if any RPLKEY is found to be invalid, a warning message is issued, the RPLKEY is set to "OFF", and condition code 8 is returned.

When restoring more than one file in the FILE file list, the RPLKEY settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLKEY parameter:

```
RPLKEY='AA,BB,,OFF'
```

In this example, field AA is used as the replication primary key for the first file, BB is used as the replication primary key for the second file, and no replication primary key is used for the fourth file (OFF). No value is provided for the third file, so an default appropriate for the file is used..

RPLTARGETID: Replication Target ID

RPLTARGETID is an optional parameter that specifies the target ID of the Event Replicator Server to which the restored transactions should be sent.

Valid RPLTARGETID settings are a valid target ID, "OFF", or no setting at all. Specifying a target ID identifies that as the target for replication. A setting of "OFF" or "0" indicates that no replication target should be used for replication.

If no setting is specified, the default value is used. The default depends on the target database ID of the restore processing and its replication state. If the restore DBID is the same as the

originally saved DBID and REPLICATION=YES in the target DBID, the original RPLTARGETID value for the file is used; otherwise the default is "OFF".

When restoring more than one file in the FILE file list, the RPLTARGETID settings must be specified as positional values corresponding to the file numbers' positions in the FILES list. For example, if four files are listed in the FILES file list, the following might be the setting for the RPLTARGETID parameter:

```
RPLTARGETID='23,24,,OFF'
```

In this example, target ID 23 is used as the replication target for the first file, 24 is used as the replication target for the second file, and no replication target is used for the fourth file (OFF). No value is provided for the third file, so an default appropriate for the file is used..

RPLUPDATEONLY: Allow Only Event Replicator Processing Updates

The `RPLUPDATEONLY` parameter can be used in the ADASAV RESTORE function to indicate whether an Adabas database file may be updated only by the Event Replicator Server as part of Adabas-to-Adabas replication or by other means as well. This parameter is optional.

Valid values are "YES" or "NO". A value of "YES" indicates that the file can only be updated via Event Replicator processing; a value of NO indicates that the file can be updated by any normal means, including Event Replicator processing.

If no value is specified, the default RPLUPDATEONLY setting of the file at the time of the corresponding SAVE operation is used.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

Examples

Example 1:

```
ADASAV RESTORE GCB,FILES=2,4,6,8
```

The database Associator GCBs, RABNs 3-30, the checkpoint and security files, and files 2,4,6, and 8 are to be restored.

Example 2:

```
ADASAV RESTORE GCB,FILES=3,4,5,OVERWRITE
```

The Associator's GCBs and files 3, 4, and 5 are to be restored; the existing database will be overwritten.

171 RESTPLOG: Restore Protection Log Only

- Essential Parameters 1246
- Optional Parameters 1247
- Example 1248

RESTPLOG restores changes contained in the PLOG to the already restored database or (if specified) files. RESTPLOG restores only the PLOG changes that were recorded during the related online SAVE (database or FILES) operation.

The RESTPLOG function is used when the following sequence occurs:

1. A SAVE data set is created online; that is, while the Adabas nucleus is active.
2. Using output created during the online SAVE, the RESTONL function is executed to restore the database or files, completes restoring the database or files from the SAVE tape, but ends due to an error condition before completing the updates recorded in PLOG.
3. The RESTPLOG function is executed to reapply all updates to the restored database or files that were recorded in PLOG. This avoids the need for restoring the complete database or files again with RESTONL.

RESTPLOG cannot be used to complete an ADASAV RESTONL FMOVE or an ADASAV RESTONL FILES with ALLOCATION=NOFORCE operation. These operations must be restarted.

```
ADASAV RESTPLOG   PLOGNUM = protection-log-number
                  { SYN1 | SYN4 } = starting-block-number
                  [FILES = file-list ]
                      [NEWFILES = file-list ]
                  [NOUSERABEND]
                  [TEST]
```

Essential Parameters

PLOGNUM: Protection Log Number

PLOGNUM specifies the number of the protection log to be restored.

SYN1|SYN4: Starting Block Number

SYN1 or SYN4 specifies the block number containing the respective SYN1 or SYN4 checkpoint at which the restore operation is to begin.

Optional Parameters

FILES: List of Files to Restore

The FILES parameter specifies the files that were being restored in the RESTONL FILES or RESTONL GCB execution that was interrupted. For RESTPLOG, the same files must be specified that were specified for the interrupted function.

The FILES parameter must be omitted if a RESTONL (database) execution was interrupted. In this case, the RESTPLOG function is performed for all files of the database.

NEWFILES: New File Numbers

The NEWFILES parameter specifies the new file numbers to be assigned to each file listed in the FILES parameter. The same new file number assignments must be specified that were specified for the interrupted RESTONL FILE function that RESTPLOG is to complete.



Note: The NEWFILES parameter is not allowed if a range of files is specified in the FILES parameter.

If NEWFILES is not specified, the files to be restored retain their original numbers.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

Example

```
ADASAV RESTPLOG PLOGNUM=30,SYN1=150
```

All updates contained on protection log 30 are to be reapplied to all affected files. The block containing the SYN1 checkpoint is 150.

172 SAVE: Save Database

- Syntax 1252
- Optional Parameters 1252
- Example 1254

The ADASAV SAVE (database) function saves the contents of the database to a sequential data set. It saves all blocks that are in use in the database.

The SAVE (database) function may be executed with the Adabas nucleus active or inactive. If executed while the Adabas nucleus is

- *active*, the RESTONL function must be used to later restore the database.
- *inactive*, the RESTORE function must be used to later restore the database.

In both cases, it is possible to restore just one or a few files from the database saved on the SAVE data set.

If the Adabas nucleus is *inactive*, it cannot be started while the SAVE function is executing, and no utility (such as ADALOD, for example) that makes changes to the database being saved can be run during the save. The SAVE function cannot be executed offline if a nucleus session autorestart is pending, or if another offline utility (such as ADALOD or ADASAV) is currently running.

If the Adabas nucleus is *active* during the execution of the save operation, users have full access to the database being saved. They can perform read, find, update, insert, and delete commands. However, utilities that make changes to the database to be saved (such as ADALOD, ADAINV, or ADADBS REFRESH, for example) must not be running and cannot be started while the save function is executing. An online save operation is also not possible if the nucleus is running without protection logging.

In an online save operation, the database to be saved may be changed while ADASAV is performing the save operation. Therefore, the Adabas nucleus writes all changed blocks to the protection log as well. This protection log must be supplied for a subsequent restore operation (that is, a RESTONL function).

The start of an online database save is marked by a SYN1 checkpoint. At the end of the online save, the nucleus synchronizes all currently active transactions. This means that Adabas performs no more update commands for users at ET status but allows the other active users to continue until they reach ET status. This status is then marked by a SYN2 checkpoint. The SYN2 checkpoint thus marks a consistent state of the database where no transactions are in progress. This state is reproduced when the database or files are restored from the SAVE data set later on.

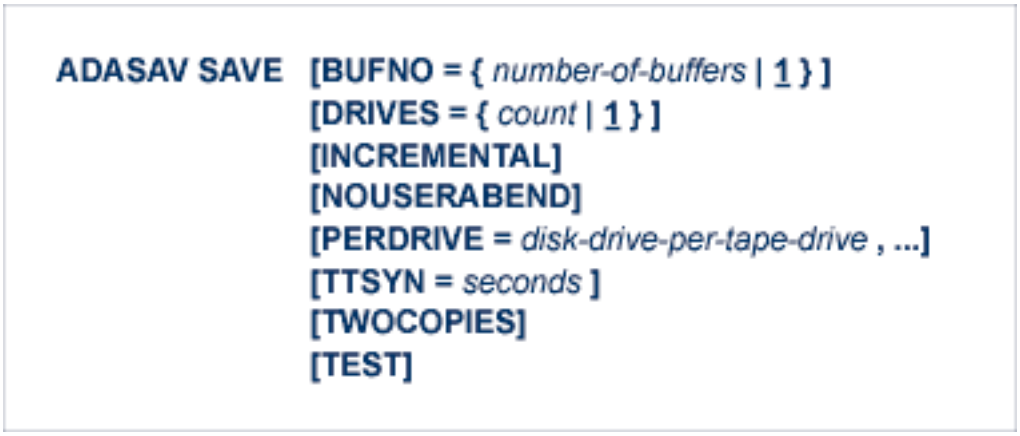
The maximum time required for the transaction synchronization can be limited by the TTSYN parameter.

Databases residing on several disk volumes are saved to several SAVE data sets in parallel when the DRIVES parameter is specified. This mode of operation may significantly reduce the duration of the save. The resulting SAVE data sets, when concatenated in the order of ascending drive number, are equivalent to a single SAVE data set produced without the DRIVES parameter.

The SAVE (database) function does not save files that are in invert, load, refresh, reorder, or restore status. In fact, it removes such files from the file list, prints message ADAU15, and performs the save operation for the remaining files. At the end, ADASAV terminates with return code 4.

If the Recovery Aid (RLOG) option is active, the SAVE (database) function starts a new RLOG generation.

Syntax



Optional Parameters

BUFNO: Count of Buffers

The BUFNO value allocates fixed buffers for the SAVE operation. A value of 2 or 3 usually provides optimum performance; up to 255 is possible. A value greater than 5, however, provides little advantage and allocates a lot of space. The default is 1 (one buffer per drive).

DRIVES: Tape Drives for Parallel Save Processing

DRIVES is the number of sequential output data sets (usually on tape drives) to be used for parallel SAVE operations. A maximum of 8 drives may be specified. The default is 1.

INCREMENTAL: Save Changed Files Only

INCREMENTAL saves only those files that have been changed since the last ADASAV SAVE operation. If INCREMENTAL is not specified, the SAVE function saves all database files.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PERDRIVE: Disk Drives Per Tape Drive

PERDRIVE specifies the number of disk drives to be assigned to a single DRIVES tape drive. For example, if the database is contained on seven disk drives and three tape drives are available for SAVE processing, PERDRIVE=3,2,2 would cause the first three disk drives to be written to tape drive 1, the next two disk drives to be written to tape drive 2, and the next two disk drives to be written to tape drive 3. The drive sequence corresponds to the DDSAVEn and DDDUALn job control specifications, as described at the end of this document.

The total number of drives specified by PERDRIVE must equal the sum of all Associator (ASSO) and DATA disks; if both ASSO and DATA are on a single disk, this counts as two separate disks.

If a single disk contains a part or the whole of ASSO or DATA in one portion larger than 65,520 cylinders, ADASAV treats this one (physical) disk as multiple (logical) disks of up to 65,520 cylinders in size. This allows the assignment of data from one (physical) disk to different tape drives. The "VOLSER-Table" in the ADASAV protocol may be consulted to check the number of (logical) disk volumes that must be matched by the PERDRIVE parameter.

If the DRIVES parameter is used and the PERDRIVE parameter is omitted, ADASAV determines the most efficient utilization of the tape drives.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

TTSYN: SYN2 Checkpoint Control

TTSYN allows the user to decrease the ADARUN TT (maximum transaction time) of the nucleus during the synchronized checkpoint processing of the current ADASAV operation. The value specified is the approximate time in seconds (TT ⊚ 1.05 seconds), and must be less than the current ADARUN TT value. If TTSYN is not specified or if TTSYN is greater than the TT value of the nucleus, that TT value becomes the default.

If the Adabas nucleus is active while ADASAV SAVE is running, a synchronized SYN2 checkpoint is taken at the end of the SAVE operation. This ensures that there is a point in time where all users are at ET status. If a user is not at ET status, no new transactions can be started for other users; they must wait until the SYN2 checkpoint can be taken.

The ADARUN TT value controls the maximum elapsed time permitted for a logical transaction. This is the maximum wait time until the SYN2 checkpoint can be processed. The ADASAV SAVE TTSYN parameter allows the user to decrease the TT value only during the synchronized checkpoint processing. The original TT value becomes effective again when ADASAV ends the SAVE operation.

TWOCOPIES: Create Two Copies of Output

TWOCOPIES creates two physical copies of the ADASAV output.

Example

```
ADASAV SAVE DRIVES=4
```

The SAVE function is to be executed using four tape drives in parallel.

173 SAVE FILES: Save Specified Files

- Syntax 1258
- Optional Parameters 1258
- Examples 1260

The ADASAV SAVE FILES function saves the contents of one or more files to a sequential data set. It saves all blocks that are in use in the file(s).

The SAVE FILES function may be executed with the Adabas nucleus active or inactive. If executed while the Adabas nucleus is

- *active*, the RESTONL function must be used to later restore the file(s).
- *inactive*, the RESTORE function must be used to later restore the file(s).

In both cases, it is possible to restore just one or a few files from all files saved on the SAVE data set.

If the Adabas nucleus is *inactive*, it cannot be started while the SAVE function is executing, and no utility (such as ADALOD, for example) that makes changes to the file(s) being saved can be run during the save. The SAVE function cannot be executed offline if a nucleus session autorestart is pending, or if another offline utility (such as ADALOD or ADASAV) is currently running on the file(s) to be saved.

If the Adabas nucleus is *active* during the execution of the save operation, users have full access to the file(s) being saved. They can perform read, find, update, insert, and delete commands. However, utilities that make changes to the files to be saved (such as ADALOD, ADAINV, or ADADBS REFRESH, for example) must not be running and cannot be started while the save function is executing. An online save operation is also not possible if the nucleus is running without protection logging.

In an online save operation, the file(s) to be saved may be changed while ADASAV is performing the save operation. Therefore, the Adabas nucleus writes all changed blocks of the file(s) being saved to the protection log as well. This protection log must be supplied for a subsequent restore operation (that is, a RESTONL function).

The start of an online file save is marked by a SYN4 checkpoint. At the end of the online save, the nucleus synchronizes all currently active transactions. This means that Adabas performs no more update commands for users at ET status but allows the other active users to continue until they reach ET status. This status is then marked by a SYN5 checkpoint. The SYN5 checkpoint thus marks a consistent state of the database where no transactions are in progress. This state is reproduced when files are restored from the SAVE data set later on.

The maximum time required for the transaction synchronization can be limited by the TTSYN parameter.

If the parameter UTYPE=EXU is specified and the Adabas nucleus is active, the save operation is performed like an offline save. ADASAV locks all files to be saved with an EXU-open against concurrent updates. The RESTORE function (rather than RESTONL) must be used for a later restore of the file(s).

Several offline file save operations, or file saves with UTYPE=EXU can be performed on different files in parallel. Only one online file save operation can be active at a time.

Files from databases residing on several disk volumes are saved to several SAVE data sets in parallel when the DRIVES parameter is specified. This mode of operation may significantly reduce the duration of the save. The resulting SAVE data sets, when concatenated in the order of ascending drive number, are equivalent to a single SAVE data set produced without the DRIVES parameter.

The SAVE FILES function does not save files that are in invert, load, refresh, reorder, or restore status. In fact, it removes such files from the file list, prints message ADAU15, and performs the save operation for the remaining files. At the end, ADASAV terminates with return code 4.

Syntax

```
ADASAV SAVE FILES = file-list  [BUFNO = { number-of-buffers | 1 } ]
                               [DRIVES = { count | 1 } ]
                               [INCREMENTAL]
                               [NOUSERABEND]
                               [PASSWORD = 'password-list' ]
                               [PERDRIVE = disk-drive-per-tape-drive , ...]
                               [TEST]
                               [TTSYN = seconds ]
                               [TWOCOPIES]
                               [UTYPE = EXU]
```

The FILES file list specifies one or more Adabas file numbers or a range of file numbers to be restored. Ranges of file numbers should be specified using a dash (-) in the format: `fnfirst-fnlast`.

If a specified file is coupled to another file or is a component of an expanded file, and the ADASAV SAVE FILES function is executed

- with an *active* nucleus, ADASAV only accepts a SAVE FILES operation if all component files of an expanded file and all files coupled to the file are specified in the FILES file list.
- with an *inactive* nucleus, the SAVE FILES operation is accepted if any one component file of an expanded file or one of several coupled files is specified. ADASAV then extends the file list automatically.

Optional Parameters

BUFNO: Count of Buffers

The BUFNO value allocates fixed buffers for the SAVE operation. A value of 2 or 3 usually provides optimum performance; up to 255 is possible. A value greater than 5, however, provides little advantage and allocates a lot of space. The default is 1 (one buffer per drive).

DRIVES: Tape Drives for Parallel Save Processing

DRIVES is the number of tape drives to be used for parallel SAVE operation. A maximum of 8 drives can be specified. The default is 1.

INCREMENTAL: Save Changed Files Only

INCREMENTAL saves only those files in the FILES list that have been changed since the last ADASAV SAVE operation. If INCREMENTAL is not specified, the SAVE function saves all files in the FILES list.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

PASSWORD: File Password

PASSWORD specifies one password or a list of passwords if one or more files in the FILES file list are password-protected. For more than one password-protected file, the correct passwords must be specified as positional values corresponding to the protected file numbers' positions in the FILES list. Refer to the examples at the end of this section for more information about the PASSWORD parameter. When saving password-protected files, the Adabas nucleus must be active.

PERDRIVE: Disk Drives per Tape Drive

PERDRIVE specifies the number of disk drives to be assigned to a single tape drive. For example, if the database is contained on seven disk drives, and three tape drives are available for SAVE processing, PERDRIVE=3,2,2 would cause the first three disk drives to be written to tape drive 1, the next two disk drives to be written to tape drive 2, and the next two disk drives to be written to tape drive 3. The drive sequence corresponds to the DDSAVEn/DDDUALn job control specifications, as described at the end of this document.

The total number of drives specified by PERDRIVE must equal the sum of all Associator (ASSO) and DATA disks; if both ASSO and DATA are one a single disk, this counts as two separate disks. If the DRIVES parameter is used and the PERDRIVE parameter is omitted, ADASAV will determine the most efficient utilization of the tape drives.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

TTSYN: SYN5 Checkpoint Control

TTSYN allows the user to decrease the ADARUN TT (maximum transaction time) of the nucleus during the synchronized checkpoint processing of the current ADASAV operation. The value specified is the approximate time in seconds (TT ⊚ 1.05 seconds), and must be less than the current ADARUN TT value. If TTSYN is not specified or if TTSYN is greater than the TT value of the nucleus, the nucleus' TT value becomes the default.

Note: TTSYN is ineffective if UTYPE=EXU.

If the Adabas nucleus is active while ADASAV SAVE is running, a synchronized SYN5 checkpoint is taken at the end of the SAVE operation. This ensures that there is a point in time where all users are at ET status. If a user is not at ET status, no new transactions can be started for other users; they must wait until the SYN5 checkpoint can be taken.

The ADARUN TT value controls the maximum elapsed time permitted for a logical transaction. This is the maximum wait time until the SYN5 checkpoint can be processed. The ADASAV SAVE TTSYN parameter allows the user to decrease the TT value only during the synchronized checkpoint processing. The original TT value becomes effective again when ADASAV ends the SAVE operation.

TWOCOPIES: Create Two Copies of Output

TWOCOPIES creates two physical copies of the ADASAV output.

UTYPE=EXU: User Type for Open

ADASAV issues an Adabas open command with a record buffer "EXU=file-list". This enables a file to be saved where an Adabas nucleus is active with no protection log. No updates to the files being saved are permitted while the SAVE function is operating. The corresponding RESTORE file operation does not require a protection log.

Examples

Example 1:

```
ADASAV SAVE FILES=10,15
```

Files 10 and 15 are to be saved.

Example 2:

```
ADASAV SAVE FILES=3,4,5,
ADASAV      PASSWORD='PWD3,,PWD5'
```

Save files 3, 4, and 5. Files 3 and 5 are password protected and their passwords are PWD3 and PWD5.

174 JCL Requirements and Examples

- Requirements to Run ADASAV JCL 1262

This section describes the job control information required to run ADASAV on the z/OS operating system and shows examples of job streams.

Requirements to Run ADASAV JCL

Data Set	DD Name	Storage	More Information
Associator	DDASSOR*n*	disk	If exclusive access is needed for your Adabas nucleus data, please review the operating system JCL for dataset disposition allocation and specify the option that works your environment. For example, `DISP=OLD`.
Data Storage	DDDATAR*n*	disk	If exclusive access is needed for your Adabas nucleus data, please review the operating system JCL for dataset disposition allocation and specify the option that works your environment. For example, `DISP=OLD`.
Work	DDWORKR1 DDWORKR*n* (1-9) to DDWORK*nn* (10-32) in cluster environments	disk	A Work data set is *not* required for: ■ any ADASAV SAVE functions ■ ADASAV RESTONL FMOVE and ADASAV RESTONL FILES functions ■ ADASAV RESTORE FMOVE and ADASAV RESTORE FILES functions ■ ADASAV RESTPLOG functions A Work data set is *recommended* for: ■ ADASAV RESTONL and ADASAV RESTONL GCB functions ■ ADASAV RESTORE and ADASAV RESTORE GCB functions In these cases, if a Work data set is specified, the ADASAV function resets it; if a Work data set is not specified, no Work data sets are reset. **Note:** In an Adabas Cluster Services or Adabas Parallel Services cluster, where each cluster has its own Work data set, the Work data sets of all nuclei in the cluster should be specified for the RESTONL/RESTORE or RESTONL GCB/RESTORE GCB functions. If a database is restored and the Work data set has not been reset, nucleus error 72 (PARM ERROR 72) may occur. To resolve this, you can reset any Work data sets using the ADAFRM utility.

Data Set	DD Name	Storage	More Information
			If exclusive access is needed for your Adabas nucleus data, please review the operating system JCL for dataset disposition allocation and specify the option that works your environment. For example, `DISP=OLD`.
Recovery log (RLOG)	DDRLOGR1	disk	Required for ADARAI
Backup copy	DDSAVE1-8	tape/ disk	Required for SAVE
Dual copy	DDDUAL1-8	tape/ disk	Required for SAVE with two backup copies
Backup copy (input for RESTORE function)	DDREST1-8	tape/ disk	Required for RESTORE and RESTONL
Sequential protection log	DDPLOG	tape/ disk	Required for RESTONL and RESTPLOG
ADARUN parameters	DDCARD	reader	*Operations*
ADASAV parameters	DDKARTE	reader	
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADASAV messages	DDDRUCK	printer	*Messages and Codes*

Note: For RESTONL, the input SAVE tapes and the sequential protection log can be concatenated, using the name DDREST1.

ADASAV JCL Examples

Save Database

```
//ADASAV    JOB
//*
//*    ADASAV:
//*       EXAMPLE HOW TO USE ADASAV TO SAVE THE
//*       ENTIRE DATABASE
//*
//SAVE     EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD          <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1    <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1    <=== DATA
//DDSAVE1   DD   DSN=EXAMPLE.DByyyyy.DDSAVE1,UNIT=TAPE, <=== OUTPUT
//            DISP=(,CATLG),VOL=SER=ADABCK
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
```

```
//DDCARD    DD   *
ADARUN PROG=ADASAV,SVC=xxx,DE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADASAV SAVE
/*
```

Refer to ADASAV in the JOBS data set for this example.

Save Database with Two Copies of Output

```
//ADASAVT   JOB
//*
//*    ADASAV:
//*       EXAMPLE HOW TO USE ADASAV TO SAVE THE
//*       ENTIRE DATABASE CREATING TWO COPIES OF THE OUTPUT
//*
//SAVE     EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD          <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1    <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1    <=== DATA
//DDSAVE1   DD   DSN=EXAMPLE.DByyyyy.DDSAVE1,UNIT=TAPE, <=== OUTPUT
//            DISP=(,CATLG),VOL=SER=ADABCK
//DDDUAL1   DD   DSN=EXAMPLE.DByyyyy.DDSAVD1,UNIT=TAPE, <=== OUTPUT
//            DISP=(,CATLG),VOL=SER=ADABCK1
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADASAV,SVC=xxx,DE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADASAV SAVE TWOCOPIES
/*
```

Refer to ADASAVT in the JOBS data set for this example.

Restore Database

```
//ADASAVR   JOB
//*
//*    ADASAV:
//*       EXAMPLE HOW TO USE ADASAV TO RESTORE THE
//*       ENTIRE DATABASE
//*
//RESTORE  EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD       <=== ADABAS LOAD
//*
```

```
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <=== WORK
//DDREST1   DD   DSN=EXAMPLE.DByyyyy.SAVE,           <=== SAVE OUTPUT
//          DISP=OLD,UNIT=TAPE,VOL=SER=ADABCK
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADASAV,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADASAV RESTORE OVERWRITE
/*
```

Refer to ADASAVR in the JOBS data set for this example.

Restore Files From SAVE Data Sets Created Offline

```
//ADASAVRF  JOB
//*
//*    ADASAV:
//*       EXAMPLE HOW TO USE ADASAV TO RESTORE A FILE
//*       TO ANY RABNS FROM AN OFFLINE SAVE
//*
//RESTORE  EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD       <=== ADABASLOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDREST1   DD   DSN=EXAMPLE.DByyyyy.SAVE,           <=== SAVE OUTPUT
//          DISP=OLD,UNIT=TAPE,VOL=SER=ADABCK
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADASAV,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADASAV RESTORE FMOVE=2
/*
```

Refer to ADASAVRF in the JOBS data set for this example.

Restore Files From SAVE Data Sets Created Online

```
//ADASAVRO  JOB
//*
//*    ADASAV:
//*       EXAMPLE HOW TO USE ADASAV TO RESTORE FILES
//*       FROM SAVE DATA SETS CREATED ONLINE
//*
//RESTORE  EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD       <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDREST1   DD   DSN=EXAMPLE.DByyyyy.SAVE,           <=== SAVE OUTPUT
//          DISP=OLD,UNIT=TAPE,VOL=SER=ADABCK
//DDPLOG    DD   DSN=EXAMPLE.DByyyyy.PLOG,           <=== PLOG OUTPUT
//          DISP=OLD,UNIT=TAPE,VOL=SER=PLOGD1
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADASAV,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADASAV RESTONL FILES=2,PLOGNUM=ppp,SYN1=1
/*
```

Refer to ADASAVRO in the JOBS data set for this example.

Restore Protection Log after an Interrupted RESTONL Function

```
//ADASAVRP  JOB
//*
//*    ADASAV:
//*       EXAMPLE HOW TO USE ADASAV TO RESTORE THE
//*       PROTECTION LOG AFTER AN INTERRUPTED RESTONL
//*
//RESTORE  EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD       <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDPLOG    DD   DSN=EXAMPLE.DByyyyy.PLOG,           <=== PLOG INPUT
//          DISP=OLD,UNIT=TAPE,VOL=SER=PLOGD1
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADASAV,SVC=xxx,DEVICE=dddd,DBID=yyyyy
```

```
/*
//DDKARTE   DD   *
ADASAV RESTPLOG PLOGNUM=ppp,SYN1=1
/*
```

Refer to ADASAVRP in the JOBS data set for this example.

XIX ADASEL Utility: Select Protection Data

This chapter covers the following topics:

Functional Overview
ADASEL Syntax
ADASEL Scenarios
JCL Requirements and Examples

175 Functional Overview

- Spanned Record Handling 1272

The ADASEL utility selects information in the Adabas sequential (SIBA) or dual/multiple (PLOG) protection log. ADASEL decompresses the information and writes it to a print data set (DDDRUCK) or to a user-specified output data set.

The protection log contains information on all updates applied to the database during a given Adabas session. Information selected with ADASEL can be used for auditing or as input to a Natural or non-Adabas program.

Some of the information you can select includes:

- Before-images, after-images, or both for new, updated, and deleted records;
- Data written to the protection log with an Adabas C5 command;
- End transaction (ET) information;
- Data by command sequence number or Adabas session number;
- Fields for an output file; and
- Counts for MU, PE, and PE/MU fields in the PLOG record.
- Updates that were split into a delete and update by Adabas can be identified as an update if desired.



Notes:

1. A logically deleted field cannot be selected by the ADASEL utility.
2. Date-time fields defined with the TZ (time zone) option will be displayed and output in UTC time (Coordinated Universal Time, also known as Greenwich Mean Time).

If the Adabas session used *dual/multiple protection logging*, use the ADARES PLCOPY function to copy the protection log before using it as input to ADASEL. If the Adabas session used *sequential protection logging*, and if the session terminated abnormally, use the ADARES COPY function to copy the protection log before using it as input to ADASEL.

Spanned Record Handling

The ADASEL utility decompresses complete spanned records written to the PLOG. If the ADASEL output instruction is to DISPLAY something, the record is always processed. If the ADASEL output instruction is to OUTPUT the decompressed records to an output data set, ADASEL first looks to see if the SPANREC parameter is specified in the OUTPUT instruction.

- If no SPANREC parameter is found, ADASEL skips processing of the spanned record, issues a warning message, and continues processing the other PLOG records.
- If the SPANREC parameter is specified, two alternate spanned record output headers, SELH and SELC, are used for all EXPA*n* output. This allows for the possibility that the output decom-

pressed spanned records will exceed the physical record length limitation. DSECTs for the SELH and SELC headers can be found in the Adabas source library. For complete information, read ***OUTPUT Instruction***, elsewhere in this section.

The SELH output header indicates, via a flag, whether a spanned PLOG record is complete or partial. It also indicates, via another flag, whether a partial field has been skipped. Relevant MU and PE indices are identified in both cases.

Standalone secondary spanned records encountered in the PLOG are rejected from further processing. A warning message is issued. Likewise, decompression of a partial field at the end of a spanned record is skipped. All fields up to the partial field are decompressed but the partial field and any remaining fields on the spanned record are not available for processing.

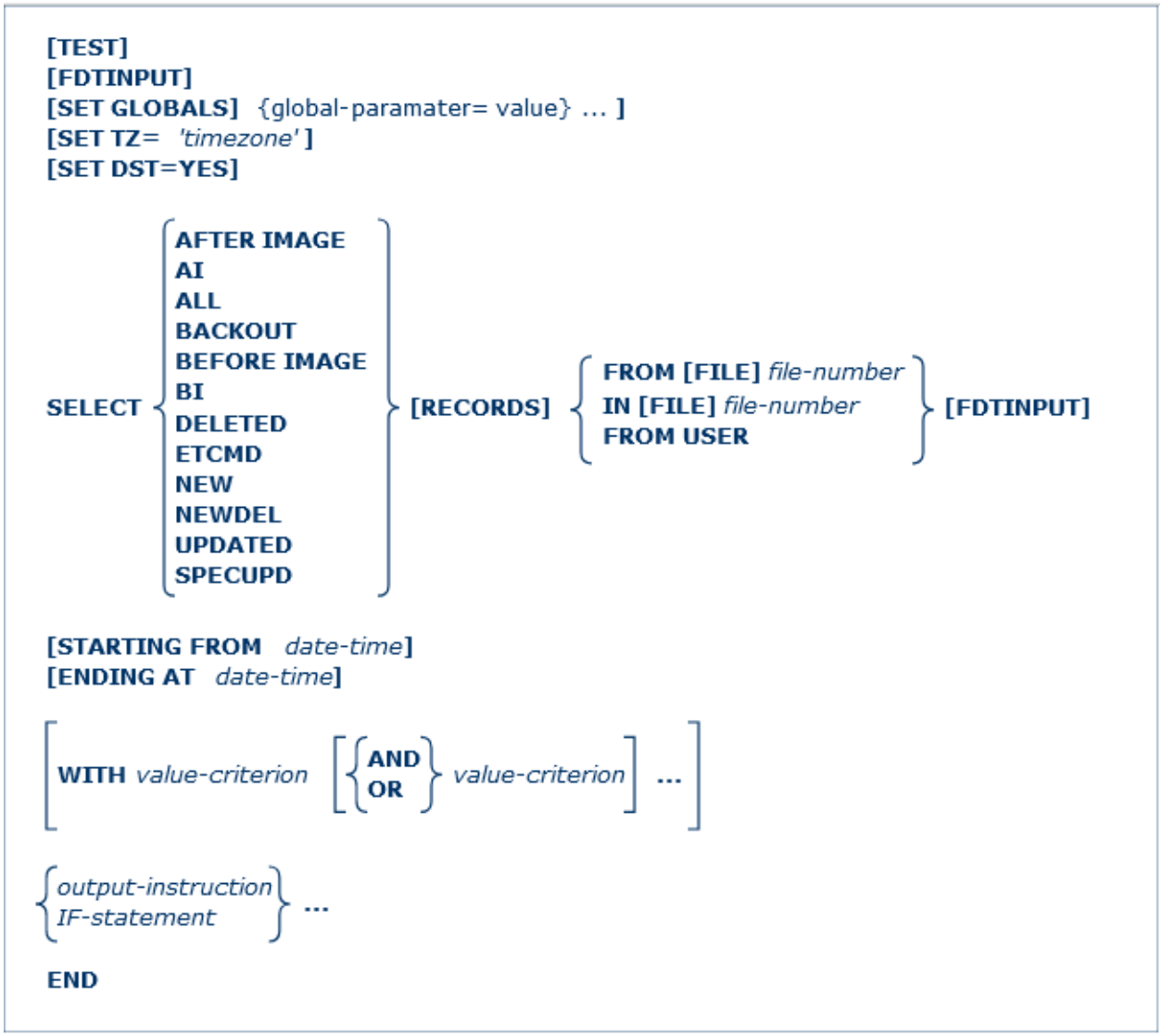
176 ADASEL Syntax

- TEST Parameter 1277
- FDTINPUT Parameter 1277
- SET GLOBALS Parameter 1280
- SET TZ Parameter - Time Zone 1282
- SET DST Parameter - Daylight Savings Indicator 1282
- SELECT Parameter 1283

Unlike other Adabas utilities, ADASEL does not require the utility name (ADASEL) at the beginning of each parameter line. A selection request must include the following parts:

- the keyword SELECT, followed by a selection option and either a file number or the keywords FROM USER
- one or more output instructions or IF statements
- the keyword END.

Optional clauses and other parameters can be included that specify additional selection criteria or processing. The following depicts an overview of the ADASEL syntax.



TEST Parameter

The ADASEL utility now includes a syntax-checking-only mode. When the optional TEST parameter is specified, the ADASEL utility syntax is checked, but not performed. The TEST parameter should be specified before any other ADASEL parameters, such as any SELECT or SET GLOBALS parameters.

In the following example, the syntax of the SELECT and other parameters will be tested. No actual data will be processed.

```
TEST
SELECT ALL RECORDS FROM FILE 1
   DISPLAY AA BB CC
END
SELECT BEFORE IMAGE FILE 2
   OUTPUT TO EXPA1
END
```

FDTINPUT Parameter

The optional FDTINPUT parameter can be used to indicate that the FDTs used for ADASEL processing should come from an alternate FDT source. This alternate FDT source can be referenced for all selections in an ADASEL run or for individual files or users selected in the ADASEL run. This functionality allows you to handle data situations where the FDT has been modified in some way so it differs from the actual data in the database. In these cases, an older or different FDT might be necessary for ADASEL to accurately process the data in the database.

If FDTINPUT is not specified in an ADASEL run, the FDTs for the files in the database are used by default.

If FDTINPUT is specified in an ADASEL run, a corresponding job statement (DDSAVE or DDEBAND) must be specified in the ADASEL run to identify the alternate FDT source to be used, as described in *ADASEL Job Requirements for FDTINPUT*, later in this section.

You can specify the FDTINPUT parameter in an ADASEL run in one of two ways:

1. You can specify it as a global parameter for the ADASEL run, in which case the alternate FDT source identified by the DDEBAND or DDSAVE job control statement is used for all files selected and processed in the ADASEL run. In the following example, where FDTINPUT is specified as a global parameter, the FDTs in the alternate FDT source are used for files 20, 35, and 36:

   ```
   FDTINPUT
   SELECT ALL FROM FILE 20
      DISPLAY AA BB CC
   END
   ```

```
SELECT ALL FROM FILE 35
   OUTPUT TO EXPA1
END
SELECT ALL FROM FILE 36
   DISPLAY ALL
END
```

In the following example, the FDTs in the alternate FDT source are used for all records in the database for user ETID1.

```
FDTINPUT
SELECT ALL FROM USER
   WITH USERID='ETID1'
   DISPLAY ALL
END
```

2. You can specify it separately for individual SELECT statements in an ADASEL run. In the following example, where FDTINPUT is specified for two files, the usual FDT in the database is used for file 20, but the FDTs in the alternate FDT source are used for both files 35 and 36:

```
SELECT ALL FROM FILE 20
   DISPLAY AA BB CC
END
SELECT ALL FROM FILE 35 FDTINPUT
   OUTPUT TO EXPA1
END
SELECT ALL FROM FILE 36 FDTINPUT
   DISPLAY ALL
END
```

Likewise, in the following example, the FDTs in the alternate FDT source are used for all the records in the database for user ETID1.

```
SELECT ALL FROM USER FDTINPUT
   WITH USERID='ETID1'
   DISPLAY ALL
END
```

This section describes the following topics:

ADASEL Job Requirements for FDTINPUT

For ADASEL FDTINPUT processing to be successful, either (but not both) the DDEBAND or DDSAVE ADASEL job statements must be specified in the ADASEL run to identify the alternate FDT source that should be used in the run. If neither or both are specified, errors will result. If FDTINPUT is not specified, but either DDEBAND or DDSAVE is, a warning message is issued and the job statements are ignored.

- If FDTINPUT is specified for only one file in an ADASEL run, either DDEBAND or DDSAVE can be used to identify the alternate FDT source for the run.
- If FDTINPUT is specified for multiple files in an ADASEL run (either on multiple SELECT statements or as a global parameter) or for a SELECT FROM USER selection in an ADASEL run, a DDSAVE job statement must be used to identify the alternate FDT source for the run. Specifying a DDEBAND job statement in these instances will result in errors.
- If the alternate FDT source for the ADASEL run is a save tape, the DDSAVE job statement must be used to identify the alternate FDT source. Specifying a DDEBAND job statement in this instance will result in errors.

Obtaining an FDT Source for Use with FDTINPUT

Prior to running an ADASEL job with the FDTINPUT parameter, an FDT source must be produced and stored. This can be accomplished in one of two ways:

1. Run the ADAULD utility for a database file that uses the FDT you want to use in your ADASEL run and specify NUMREC=0. This will only unload the FDT. For complete information, read *ADAULD Utility: Unload Files* , elsewhere in this guide. FDTs obtained in this manner should be specified in the DDEBAND job statement in the ADASEL job.
2. An old save tape can be used as input to ADASEL FDTINPUT processing. When ADASEL processes the save tape, it reads it sequentially to determine the file numbers of the files on the tape. In addition, if a SELECT FROM USER selection is requested in an ADASEL run, the entire save tape is read in advance to obtain all of the FDTs and their associated file numbers in advance of ADASEL processing.

   To produce a new save tape, run the ADASAV or ADASAV FILES utility function for one or more database files that use the FDTs you want to use in your ADASEL run. This will produce a save tape that can be used as input to the ADASEL run.

In either case, the FDT source provided for FDTINPUT must match the version of the protection log provided in the ADASEL job. In addition, only one alternate FDT source can be specified for FDTINPUT processing in a single ADASEL run.

SET GLOBALS Parameter

ADASEL global parameters override default table and buffer sizes. Overrides are in effect only for the ADASEL run in which the SET GLOBALS statement is specified.

If used, the SET GLOBALS settings must be specified before the first ADASEL SELECT parameter. Comment statements as well as the FDTINPUT and TEST parameters can precede the SET GLOBALS settings. SET GLOBALS settings are specified in the following syntax:

SET GLOBALS { *global-parameter* = *value* } ...

No spaces are permitted between the parameter name, the equal sign, and the value. However, at least one space must separate parameters. Special characters are not permitted as separators. If multiple lines are used, the SET GLOBALS keyword must be repeated on each line. The first non-blank character string that does not begin with a parameter name terminates the SET GLOBALS statement. Thus, trailing comments are not permitted.

ADASEL provides the following global parameters. Default values are underscored.

Global Parameter	Description
`LPV={n | 0}`	Use this parameter to specify the length of the PE-value table used in the evaluation of field values for a PE. Normally, ADASEL uses an estimated number of PE occurrences to compute the table size. If the table size is insufficient, a SEL047 error occurs; you can increase the table size using the global LPV parameter as indicated on the screen.
`LS={n | 80}`	Use this parameter to specify the line size parameter is used to alter the number of printed columns. If an output line is longer than the line size, the line is truncated at the nearest blank. The rest of the line is continued on the next output line, beginning in Column 1. The minimum line size is 1; the maximum is 132.
`LST={len | 12000 ↩ }`	Use this parameter to specify the length of the statement table, which is used to store the translated ADASEL statements. Depending on its complexity, a statement is translated into one or more segments. Each segment is 44 bytes plus a value length. For example: `IF BA EQ 'SMITH'...` requires 49 bytes: 44 bytes plus 5 bytes for "SMITH". The default table size (12 ,000 bytes) handles approximately 200 segments. If the table size is exceeded, a SEL003 error occurs.
`LWP={n | 1048576}`	Use this parameter to specify the size of the work pool used internally by the ADASEL utility for spanned record processing. The default value of LWP is 1048576 bytes (or 1MB). If the value specified for LWP is appended with the letter "K", it is multiplied by 1024. Valid values range from 100K - 1048576K (or 1 GB).
`MAXLOGRECLEN={n ↩ | 1048576`	Use this parameter to specify the size of the uncompressed record buffer allocated by the ADASEL utility for use in spanned record processing. The default value of

Global Parameter	Description
	MAXLOGRECLEN is 1048576 bytes (or 1MB). If the value specified for MAXLOGRECLEN is appended with the letter "K", it is multiplied by 1024. The minimum value is 32768 bytes.
NCFLD={*n* | 10} NCUPD={*n* | 10}	Use this parameter to specify the maximum count of "*field-name* CHANGES" statements allowed in the selection query, and the maximum number of parallel updates during the original session. When a statement includes a CHANGES criterion, ADASEL uses a change pool with NCFLD * NCUPD entries to track changed field values. If this pool is too small, a SEL060 error occurs. In this case, it is necessary to increase one or both of the parameters and then rerun ADASEL.
NF={*n* | 20}	Use this parameter to specify the maximum number of files that can be processed during a single ADASEL run. NF is used to allocate space for the FDT for each file processed. A SEL014 error occurs if the NF value is exceeded. This value is *not* related to the maximum number of output files (DDEXPA*n*; although more than 20 files can be processed during an ADASEL run, a maximum of 20 output files can be written.
NIF={*n* | 20}	Use this parameter to specify the number of nested IF levels permitted.
NOUSERABEND	If specified, ADASEL terminates with condition code 20 instead of a user abend 034 after an error is encountered.
NU={*n* | 20}	Use this parameter to specify the maximum number of user values (or ranges of user values) that can be processed during a single ADASEL run. NU is used to allocate work storage for the user data requested via a SELECT FROM USER specification in ADASEL. The default is 20 user values.

NV={*n* | 100}	Use this parameter to specify the number of field values. NV is used to allocate a table for the evaluation of field values. One entry is required for every field specified in the statements (including duplications). For example, the following statement requires two entries even though the same Adabas field name is used:
	
	`IF BA ='SMITH'`
	`  THEN OUTPUT TO EXPA1`
	`ELSE IF BA ='SMYTH'`
	`  THEN OUTPUT TO EXPA2`

PS={*n* | 60}	Use this parameter to specify the page size parameter is used to alter the number of lines printed before a new page is started. The minimum page size is 2; the maximum is 999.

Example

```
SET GLOBALS LST=15000 NF=15
SET GLOBALS LS=132
```

SET TZ Parameter - Time Zone

The TZ parameter can be used to specify the local time zone. As records are decompressed, the date-time data is converted from UTC time (Coordinated Universal Time, also known as Greenwich Mean Time) to the corresponding time for the specified time zone. Date-time field data is always stored in UTC time in an Adabas file. Valid time zone names are listed in the TZINFO member of the Adabas source library and are specified in single quotes. The following is an example of a valid TZ specification:

```
SET TZ='America/New_York'
```

Note: Time zone names are case-sensitive.

This example sets the time zone to Eastern US. When decompressing input, local time is assumed to be Eastern US. When retrieving the stored UTC time data, it is converted to Eastern US time.

Adabas uses the time zone data taken from the tz database, which is also called the zoneinfo or Olson database. The specific list of time zone names that Adabas supports in any given release can be found in the TZINFO member of the Adabas source library (ADA*vrs*.SRCE). For more information about the TZINFO member of the time zone library, read *Supported Time Zones*.

Note: Review important information about the daylight savings time (**DST**) parameter. If a time zone uses daylight savings time, you must be sure to retrieve the daylight savings indicator with your date-time data or there will be no way to distinguish date-time values in the hour before the time is switched back to standard time.

SET DST Parameter - Daylight Savings Indicator

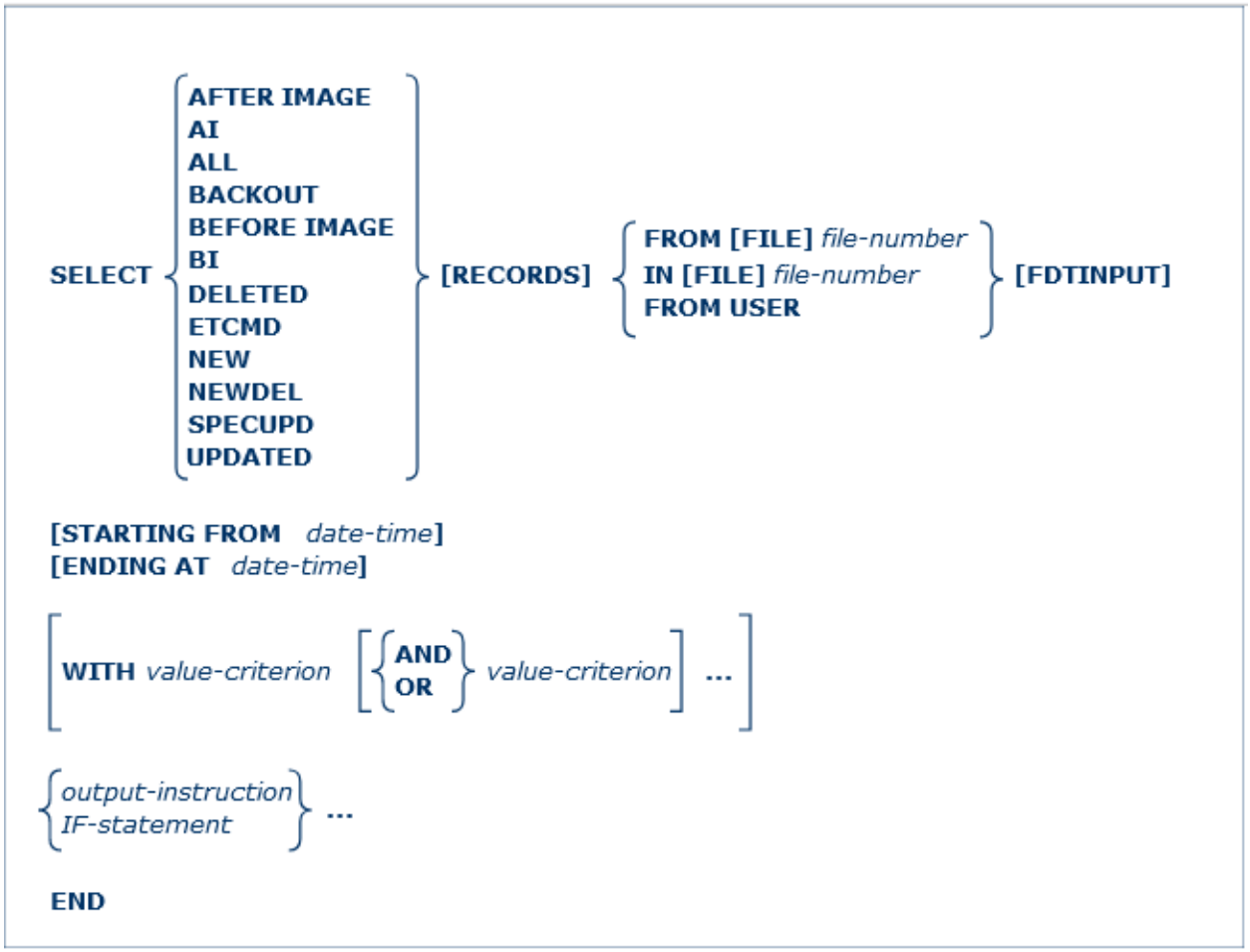
The DST parameter can be specified to indicate that date-time data includes a daylight savings time indicator. If a time zone uses daylight savings time, you must be sure to store and retrieve the daylight savings indicator with your date-time data or there will be no way to distinguish date-time values in the hour before the time is switched back to standard time. The two-byte daylight savings indicator directly follows the date-time value in uncompressed output and specifies the binary integer value of the daylight saving time offset from standard time in seconds.

If you specify the DST parameter for files containing date-time fields defined with option TZ, the field value is followed by the daylight savings time indicator in the uncompressed output, otherwise it is not. The default is that the date-time data is without a daylight savings time indicator in the uncompressed output.

The DST parameter requires that the **TZ** parameter be set in the same ADASEL run.

SELECT Parameter

The syntax of the SELECT parameter is shown below. It begins with a SELECT keyword and ends with the END keyword.



The SELECT parameter requires the following minimal specifications:

- the keyword SELECT, followed by a selection option and either a file number or the keywords FROM USER
- one or more output instructions or IF statements
- the keyword END.

Optional clauses and other parameters can be included that specify additional selection criteria or processing. The following depicts an overview of the ADASEL syntax.

This section describes both the required and optional elements of a SELECT parameter, in the order they are shown in the syntax above.

- The SELECT Keyword
- Selection Options (ALL, BEFORE IMAGE, AFTER IMAGE, etc.)
- RECORDS Keyword
- RECSYNC Keyword
- FROM and IN [FILE] file-number Clause
- FROM USER Clause
- FDTINPUT Parameter
- STARTING FROM and ENDING AT date-time Clauses
- WITH Clause
- IF Statement
- value-criterion
- output-instruction
- The END Keyword
- Examples

The SELECT Keyword

The SELECT keyword is a required element of the SELECT parameter and must be specified first.

Selection Options (ALL, BEFORE IMAGE, AFTER IMAGE, etc.)

One of the selection options described in the following table is required in a SELECT parameter and must be specified immediately after the SELECT keyword:

Selection Option	Records Selected
AFTER IMAGE | AI	After-images derived from A1 and N1 commands.
ALL	Before-images derived from A1 (update) and E1 (delete) commands; after-images derived from A1 and N1 (add) commands.
BACKOUT	Backout Transaction (BT) information.
BEFORE IMAGE | BI	Before-images derived from A1 and E1 commands.
DELETED	Before-images derived from E1 commands.
ETCMD	End transaction (ET) information. An example of the use of this option can be found in ***Selecting End Transaction (ET) Information***, elsewhere in this section.
NEW	After-images derived from N1 commands.
NEWDEL	After-images derived from N1 commands and before-images derived from E1 commands.
UPDATED	Before-images and after-images derived from A1 commands.
SPECUPD	Before-images and after-images that originated as an update but were split or migrated.

RECORDS Keyword

The RECORDS keyword is optional in the SELECT parameter. When specified, it should immediately follow the **selection option**.

RECSYNC Keyword

The RECSYNC keyword is optional in the SELECT parameter. RECSYNC can only be supplied in conjunction with SPECUPD and specifies that the BI and AI images are displayed together.

For further information on SPECUPD or RECSYNC please see *ADASEL Scenarios*, elsewhere in this section.

FROM and IN [FILE] file-number Clause

One of the FROM [FILE], IN [FILE} or the FROM USER clauses is required in a SELECT parameter. Only one is required; if more than one is specified, errors will result. The FROM USER clause is described in *FROM USER*, elsewhere in this section.



Note: A single ADASEL run cannot include both FROM [FILE] (or IN [FILE]) clauses and FROM USER clauses. The FROM [FILE] (IN [FILE]) and FROM USER clauses are mutually exclusive in an ADASEL run. An ADASEL run should either process protection log data by file (FROM [FILE] or IN [FILE] clauses) or by user (FROM USER clause), but not both. Any attempt to process protection log data by file and user in the same ADASEL run will cause errors.

The FROM [FILE] and IN [FILE] clauses are equivalent clauses. Both specify the number of the Adabas file for which protection log data will be selected and processed in the ADASEL run. In both cases the keyword FILE is optional. Valid file numbers range from 0-5000 or 0 through one less than the ASSO block size, whichever is lower. To select user data written by a C5 command, specify the file number of the checkpoint file.

FROM USER Clause

One of the FROM [FILE], IN [FILE} or the FROM USER clauses is required in a SELECT parameter. Only one is required; if more than one is specified, errors will result. The FROM and IN [FILE] clauses are described in *FROM and IN [FILE]* , elsewhere in this section.



Note: A single ADASEL run cannot include both FROM [FILE] (or IN [FILE]) clauses and FROM USER clauses. The FROM [FILE] (IN [FILE]) and FROM USER clauses are mutually exclusive in an ADASEL run. An ADASEL run should either process protection log data by file (FROM [FILE] or IN [FILE] clauses) or by user (FROM USER clause), but not both. Any attempt to process protection log data by file and user in the same ADASEL run will cause errors.

The FROM USER clause indicates that all records from all files that satisfy the selection criteria should be selected and processed in the ADASEL run, regardless of file number.

To select records from a specific user or terminal ID, specify the actual user or terminal ID used for selection using the WITH clause of the SELECT parameter. For example, the following SELECT parameter selects all protection log data for user or terminal ID ETID1:

```
SELECT ALL FROM USER FDTINPUT
   WITH USERID='ETID1'
   DISPLAY ALL
END
```

Be aware that FROM USER clauses can generate quite a bit of output for display, so be prepared to limit the selection criteria in the SELECT parameter if necessary. In addition, you can use the NU global parameter to limit the number of user or terminal IDs that can be processed by a SELECT parameter. If this number is exceeded, errors result (the default is 20). For more information about the NU global parameter, read ***SET GLOBALS Parameter***, elsewhere in this section.

Finally, when FROM USER is specified, records from different files may be encountered for the same SELECT statement. For this reason, no field names can be specified in any associated **value criteria** or in **DISPLAY instructions**. However, the DISPLAY ALL option can be used to display all fields of a selected record, regardless of the file the record belongs to.

FDTINPUT Parameter

The optional FDTINPUT parameter can be used to indicate that a different FDT from the FDT on the database should be used for specific files or users selected in the ADASEL run. This functionality allows you to handle data situations where the FDT has been modified in some way so it differs from the actual data in the database. In these cases, an older FDT might be necessary for ADASEL to accurately process the data in the database. For more information, read ***FDTINPUT Parameter***, elsewhere in this section.

STARTING FROM and ENDING AT date-time Clauses

The optional STARTING FROM and ENDING AT clauses can be used to restrict selections to records added, updated, or deleted within a time range. The following are valid formats for the *date-time* variable:

Format	Description
yyyymmdd/hhmmss	date/time
J(*yyyyddd/hhmmss*)	Julian date/time
X' *xxxxxxxx* '	store-clock (STCK) representation

Note: The lowest valid value for *yyyy* is "1980".

Examples:

Select all records from file 1 that were added, deleted, or updated on or before midnight of May 12, 1996 (Julian date 132):

```
SELECT ALL RECORDS FROM FILE 1
   ENDING AT J(1996132/240000)
   DISPLAY AA BB CC
END
```

Select all records from file 112 that were added, deleted, or updated on or between January 1 and December 31, 1996:

```
SELECT ALL 112
   STARTING FROM 19960101/000000
   ENDING AT 19961231/240000
   OUTPUT TO EXPA1
END
```

WITH Clause

The optional WITH clause can be used to select records that satisfy the **value-criteria** specified. Multiple conditions can be specified using the logical operators AND and OR.

- If **value-criteria** are connected by the AND operator, *all* specified conditions must be satisfied in order for the record to be selected.
- If **value-criteria** are connected by the OR operator, the record is selected if *any* of the conditions is satisfied.

The syntax of the *value-criterion* variable is described in section ***value-criterion***.

Example:

The protection log contains before and after images for two updated records. The contents of the field BB in the records are shown below:

Before-Image	After-Image
BB = SMITH	BB = ZINN
BB = SMITH	BB = JONES

The SELECT statement includes a WITH clause that further qualifies the selection:

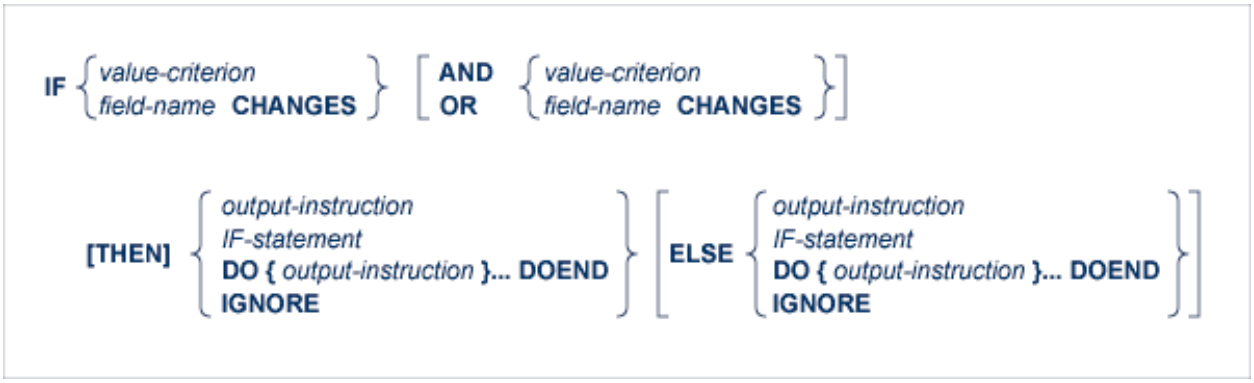
```
SELECT ALL RECORDS FROM FILE 1
   WITH BB ='SMITH'
   DISPLAY AA BB CC
END
```

In this example, despite the fact that the ALL option is used, only the two before-images are selected (because the BB field contains "SMITH" in the before-images). ADASEL ignores all records (in this case, the two after-images) in which the BB field has a value other than "SMITH". If the AFTER IMAGE option were specified, no records would be selected.

IF Statement

The IF statement can be used to select records and execute **output instructions** on a conditional basis. An IF statement is optional in a SELECT parameter, but can be used to specify conditional output instructions for the SELECT parameter. At least one output instruction is required in a SELECT parameter, so if one has not been specified outside an IF statement, an IF statement is necessary to supply the output instruction information.

By default, ADASEL permits up to 20 nested IF statements. To change this, use the NIF ADASEL global parameter. For further information about the NIF global parameter, read ***SET GLOBALS Parameter***, elsewhere in this section.



The syntax of the *value-criterion* variable is described in section **value-criterion** . Output instructions are described in section **output-instruction**.

The "*field-name* CHANGES" criterion selects records in which the value of a specified field changed during an update. ADASEL detects the change between the before-image and the after-image. Thus, this criterion is valid only for the A1 (UPDATE) command, which writes both a before-image and an after-image to the protection log. The *field-name* must be the two-character Adabas name of an elementary field in the FDT. It *cannot* refer to a group, periodic group (PE), superdescriptor, subdescriptor, phonetic descriptor, or hyperdescriptor. However, it can refer to a multiple-value field (MU) or a member field of a periodic group (PE); see the section **value-criterion**, particularly in the subsection **Indexes for MUs and PE Member Fields** .

Note: By default, only the after-image is reported for "IF *field-name* CHANGES" criterion. If you want to report both the before-image and the after-images of a changed field using ADASEL, either specify the BOTH option in the DISPLAY instruction or specify the LOGINFO, EXTENDED, EXTRACT or SPANREC options on the OUTPUT instruction for the

run. For more information, read *DISPLAY Instruction* or *OUTPUT Instruction*, elsewhere in this section.

The syntax for DO group is as follows:



A DO group is a sequence of output instructions (NEWPAGE, SKIP, DISPLAY, and OUTPUT). The group must begin with the keyword DO and end with the keyword DOEND. A DO group cannot contain nested IF statements and cannot be nested within another DO group.

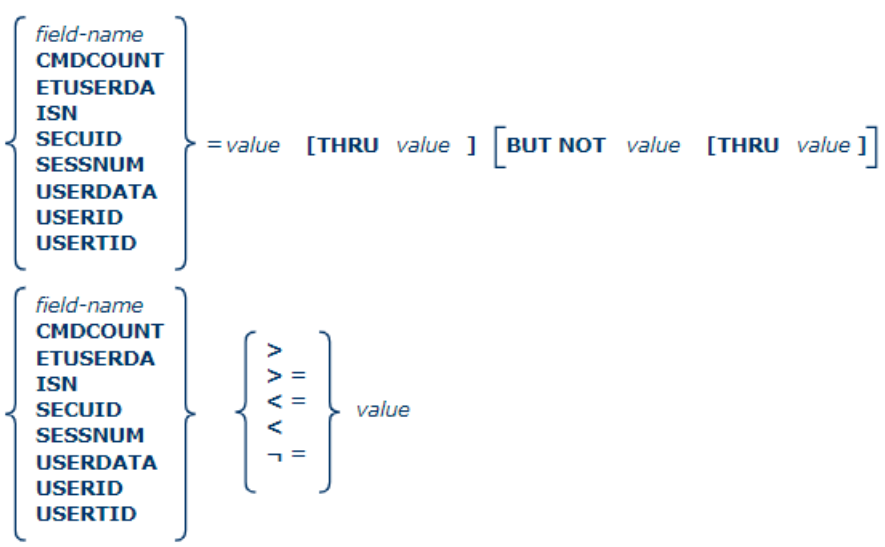
IGNORE instructs ADASEL not to display or output an item. Neither the before-image (BI) or the after-image (AI) is produced as output when an item is ignored. When specified in a THEN instruction, IGNORE will not display or output the item *if it meets* the specified value-criterion or the CHANGES criterion of the IF statement. When specified in an ELSE instruction, IGNORE will not display or output the item *if it does not meet* the specified value-criterion or the CHANGES criterion of the IF statement.

Example:

```
SELECT ALL FROM FILE 77
   IF AA ='SMITH' THEN
      IF BB CHANGES THEN DO
         DISPLAY 'Field BB changed:' BB AA CC
         SKIP 1 LINE
      DOEND
      ELSE DISPLAY AA BB CC
   ELSE IGNORE
END
```

value-criterion

The value-criterion is used in a **WITH clause** or an **IF statement** to select records on the basis of a value or values. It has the following syntax:



The BUT NOT clause excludes a value or subrange of values within the select records.

Object of the Comparison

ADASEL can compare a value or range of values to any of the following:

- The contents of the specified field. The *field-name* must be the two-character Adabas name of an elementary field in the FDT. It *cannot* refer to a group, periodic group (PE), super-descriptor, subdescriptor, phonetic descriptor, or hyperdescriptor. However, it can refer to a multiple-value field (MU) or a member field of a periodic group (PE); see ***Indexes for MUs and PE Member Fields*** .
- CMDCOUNT; that is, the command sequence number. The command sequence number can be used to select PLOG records based on their CMDCOUNT values. An example of the use of this option can be found in ***Selecting Data by Command Sequence Number***, elsewhere in this section.

  Note: The command sequence number in an Adabas PLOG is not unique. There may be multiple occurrences of this number in your PLOG data set. To ensure accurate data selection and processing, we recommend that you select on the date or time values in addition to the command sequence number. The command sequence number, date, and time values are also stored in any **EXTENDED** or **EXTRACT** output produced by the OUTPUT statement.

- ETUSERDA; that is, the user data stored with an ET command.
- The ISN; that is, the Adabas internal sequence number of the record.
- SECUID; that is, the security system user ID of the user who added, deleted, or updated the record. The SECUID is stored in the EXTRACT output produced by ADASEL's OUTPUT statement.

- SESSNUM; that is, the Adabas session number. When SESSNUM is specified in the value criterion of WITH or IF statements, the Adabas session number can be used to select PLOG records based on their SESSNUM values. In addition, the SESSNUM value is stored in any **EXTENDED** or **EXTRACT** output produced by ADASEL's OUTPUT statement. An example of the use of this option can be found in ***Selecting Data by Adabas Session Number***, elsewhere in this section.
- USERDATA; that is, the user data written by a C5 command.
- USERID; that is, the user ID (ETID) of the user who added, deleted, or updated the record.
- USERTID; that is, the terminal ID of the user who added, deleted, or updated the record.

Logical Operator

You can express logical operators for equalities and inequalities in words, abbreviations, or symbols as shown in the following table:

Comparison	Words	Abbreviation	Symbol
Equals	EQUAL	EQ	=
Greater than	GREATER THAN	GT	>
Greater than or equal to	GREATER EQUAL	GE	> =
Less than or equal to	LESS EQUAL	LE	< =
Less than	LESS THAN	LT	<
Not equal to	NOTEQUAL	NE	¬=

Note: The hexadecimal representation of the ¬= symbol is X'5F7E'.

Format of the Value

The format of the criterion value depends on the *default format* of the item that is the object of the comparison.

The default format of an Adabas field (`field-name`) is the format specified in the FDT. The following table shows the maximum length (in bytes) and valid formats for expressing the criterion value:

Criterion Value		Max. Bytes	Max. Digits
Field Format in FDT	Valid Formats		
Alphanumeric	Alphanumeric	253	
	Hexadecimal	253	506
Decimal (Packed or Unpacked)	Decimal digits (0-9)	29 *	
Binary	Decimal	4 *	10
	Hexadecimal	126	252
Floating-Point	Hexadecimal	8	16
Fixed-Point	Hexadecimal	4	8

Criterion Value		Max. Bytes	Max. Digits
Field Format in FDT	Valid Formats		
Wide-character	Hexadecimal	253	506

* *Excluding minus sign*

The default formats and maximum lengths (in bytes) for other items are as follows:

Item	Default Format	Criterion Value	
		Valid Formats	Max. Length
CMDCOUNT	Binary	Decimal	3
ETUSERDA	Alphanumeric	Alphanumeric, hexadecimal	30
ISN	Binary	Decimal, hexadecimal	4
SECUID	Alphanumeric	Alphanumeric, hexadecimal	8
SESSNUM	Binary	Decimal	2
USERDATA	Alphanumeric	Alphanumeric, hexadecimal	30
USERID	Binary, alphanumeric	Decimal, hexadecimal	8
USERTID	Binary, alphanumeric	Decimal, hexadecimal	8

Value Format Example 1:

If the default format is alphanumeric, the value can be expressed in alphanumeric or hexadecimal format.

```
BA EQ 'SMITH' or BA EQ X'E2D4C9E3C8'
```

Value Format Example 2:

If the default format is packed or unpacked decimal, the value is expressed in decimal digits (0-9). A leading minus sign indicates a negative value. Up to 29 digits (excluding the minus sign) are permitted. Other special characters ($, decimal points, commas, etc.) are not permitted.

```
NU = 123456789
NU = -987654321
```

Value Format Example 3:

If the default format is binary, the value can be expressed in hexadecimal or numeric format.

Up to 252 hexadecimal digits (126 bytes) are permitted for a binary Adabas field.

In numeric format, up to 10 decimal digits (4 binary bytes) are permitted. Thus, a binary value expressed in decimal digits can range from -2147483648 through 2147483647.

```
BB = 2147483647  or   BB = X'80000000'
BB = -2147483648 or   BB = X'7FFFFFFF'
```

Alphanumeric Values

Enclose an alphanumeric value in apostrophes:

```
AA ='SMITH'
```

To indicate an apostrophe within an alphanumeric string, use two successive apostrophes with no intervening space or character:

```
JJ ='Smith''s Market'
```

Hexadecimal Values

Begin a hexadecimal value with an "X" and enclose the value in apostrophes:

```
AA = X'E2D4C9E3C8'
```

A hexadecimal value must have an even number of hexadecimal characters:

```
JJ = X'04D2'
```

Continuation Lines

ADASEL treats columns 1-72 as the input line. To continue an alphanumeric or hexadecimal value on additional lines, place the closing apostrophe only at the end of the entire string. The value is concatenated until the closing apostrophe is found.

In an alphanumeric string, ADASEL includes leading and trailing spaces within apostrophes as part of the string; it ignores them in a hexadecimal string.

Example 1: Alphanumeric String

```
                                                                      7
1.....................................................................2

       AA ='THIS IS AN EXAMPLE OF HOW TO CONTINUE AN ALPHANUMERIC VALU
E. KEY THROUGH COLUMN 72 AND CONTINUE IN COLUMN 1 OF THE NEXT
LINE.'

                                                                      7
1.....................................................................2

       AA ='DO NOT CONTINUE AN ALPHA VALUE THIS WAY. LEADING AND
   TRAILING SPACES IN COLUMNS 1-72 ARE INCLUDED.'
```

ADASEL treats the second value above as follows:

```
'DO NOT CONTINUE AN ALPHA VALUE THIS WAY. LEADING AND         TRAILING BLANKS
IN COLUMNS 1-72 ARE INCLUDED.'
```

Example 2: Hexadecimal String

```
                                                                       7
1.....................................................................2

          XX = X'C1C2C3C4C5C6C7C8C9
                 D1D2D3D4D5D6D7D8D9'
```

ADASEL treats the hexadecimal value above as follows:
`X'C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9'`

Indexes for MUs and PE Member Fields

MU Field or a Member Field of a PE

If the *field-name* refers to an multiple-value field (MU) or to a member field of a periodic group (PE), you must include the index (occurrence number) immediately after the name:

AAi	where "AA" is the field name of an MU and *i* is the index
BBk	where "BB" is a member field of a PE and *k* is the index of the PE

Valid values for *i* and *k* range from "1" through "65534" if you have Adabas 8 or later installed and if extended MU and PE counts are requested; otherwise the valid values range from "1" through "191".

Note: The use of more than 191 MU fields or PE groups in a file must be explicitly allowed for a file (it is not allowed by default). This is accomplished using the ADADBS MUPEX function or the ADACMP COMPRESS `MUPEX` and `MUPECOUNT`parameters.

Examples:

In file 12, the field JT is an MU. The following statement selects all before-images where the second occurrence of JT is "Programmer":

```
SELECT BI FROM FILE 12
   WITH JT2 = 'Programmer'
   DISPLAY NA
END
```

The field SA is a member of a PE. The following statement selects all records where SA in the third occurrence of the periodic group is greater than or equal to 35000:

```
SELECT ALL FROM 12
   WITH SA3 >= 35000
```

```
   DISPLAY NA SA3
END
```

MU Contained Within a PE

If an MU is contained within a PE, *both* indexes (PE and MU) must be specified:

ABk(i)	where "AB" is the name of an MU, *i* is the occurrence of AB, and *k* is the occurrence of the PE to which AB belongs

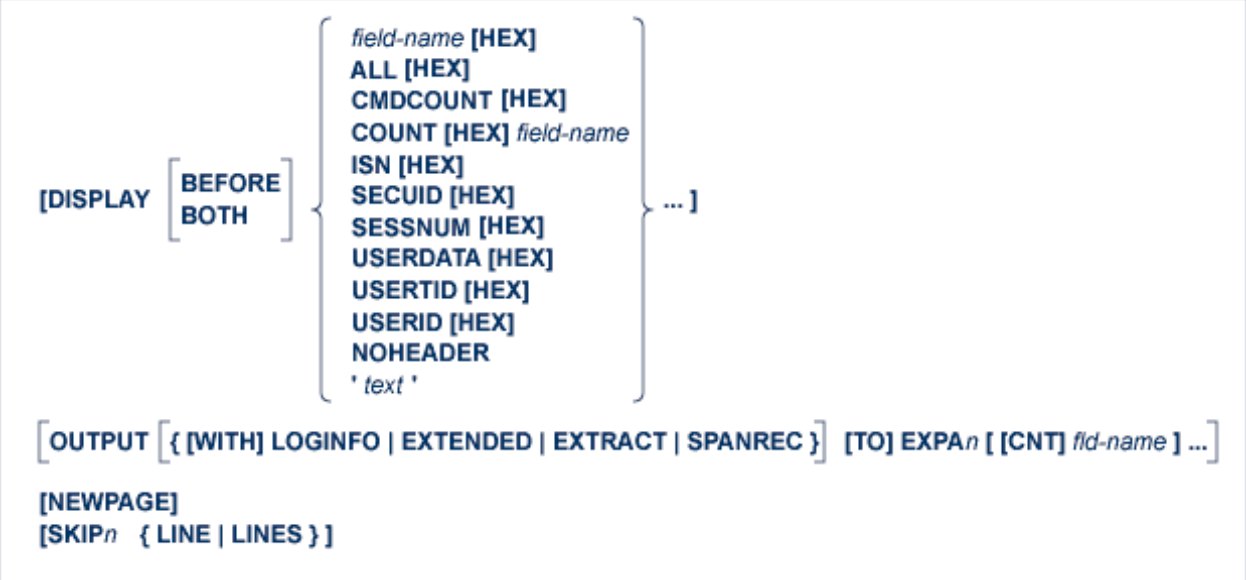
Example:

In file 211, the multiple-value field ST is a member of a PE. The following statement selects all records in which the third occurrence of ST in the second occurrence of the periodic group is "PAST DUE":

```
SELECT ALL FROM FILE 211
   WITH ST2(3) ='PAST DUE'
   DISPLAY AA BB ST2(3)
END
```

output-instruction

ADASEL output instructions include DISPLAY, OUTPUT, SKIP, and NEWPAGE. At least one output instruction is required, either separately or within an IF statement. Multiple output instructions can be specified. The syntax is shown below:



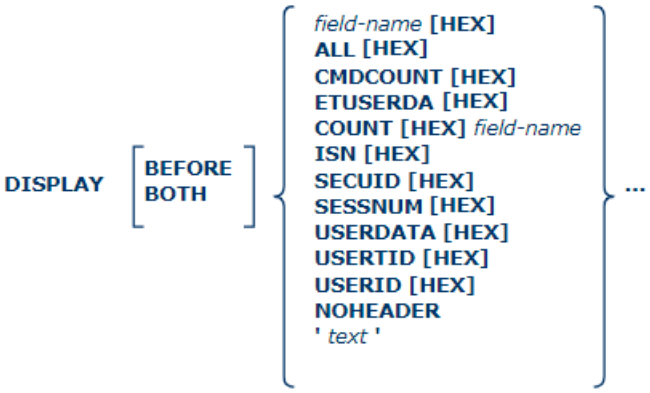
This section describes each of these output instructions.

- DISPLAY Instruction

- OUTPUT Instruction
- NEWPAGE Instruction
- SKIP Instruction

DISPLAY Instruction

DISPLAY writes the output report to DDDRUCK. The syntax specifies one or more output types. When specifying multiple output types, they are separated by at least one space:



where:

field-name	displays the contents of the specified field. The *field-name* must be the two-character Adabas field name of an elementary field in the FDT. *field-name cannot* refer to a group, periodic group (PE), superdescriptor, subdescriptor, phonetic descriptor, or hyperdescriptor. However, it can refer to a multiple-value field (MU) or a member field of a PE; indexes for MUs and PE member fields are discussed in section **MU or PE Fields**.
ALL	displays all fields of each selected record, including MU and PE group fields and their occurrence counts. For an FDT with many fields, you may need to increase the LST global parameter in a **SET GLOBALS statement** to provide sufficient space for the large number of records. This option is available for both SELECT FROM USER and SELECT FROM FILE (IN FILE) clauses in a SELECT parameter.
BEFORE	indicates that only before image of protection log data should be output. This option is valid only within IF statements containing one or more CHANGES specifications. This option is optional and is mutually exclusive with the BOTH option. If neither BOTH nor BEFORE is specified, only the after image of protection log data is output.
BOTH	indicates that both before and after images of protection log data should be output. This option is valid only within IF statements containing one or more CHANGES specifications. This option is optional and is mutually exclusive with the BEFORE option. f neither BOTH nor BEFORE is specified, only the after image of protection log data is output.
CMDCOUNT	selects and displays data from the protection log data set (PLOG) based on the command sequence number. When CMDCOUNT is specified in the DISPLAY statement, the command sequence number appears in the printed output produced by the ADASEL utility. In addition, the CMDCOUNT value is stored in any **EXTENDED** or **EXTRACT** output produced by

	ADASEL's OUTPUT statement. An example of the use of this option can be found in *Selecting Data by Command Sequence Number*, elsewhere in this section. **Note:** The command sequence number in an Adabas PLOG is not unique. There may be multiple occurrences of this number in your PLOG data set. To ensure accurate data selection and processing, we recommend that you select on the date or time values in addition to the command sequence number. The command sequence number, date, and time values are also stored in any **EXTENDED** or **EXTRACT** output produced by the OUTPUT statement.
ETUSERDA	displays records written with an ET command, and displays ET USER DATA where it is present. USER must be specified in the SELECT statement.
COUNT	displays the number of occurrences in the selected data of the MU field or PE group field specified after the COUNT option. If one or more values of the MU or PE field are to be displayed as well, they must be specified separately.
option	displays the hexadecimal value corresponding to the type of output. HEX is especially useful if the output contains unprintable characters. Leave at least one space between the type of output and the following HEX keyword.
ISN	displays the ISN of each selected record.
SECUID	displays the security system user ID of the user who added, deleted, or updated the record. If the security system user ID is not known, "UNKNOWN" displays. The SECUID is stored in the EXTRACT output produced by ADASEL's OUTPUT statement.
SESSNUM	selects and displays data from the protection log data set (PLOG) based on the Adabas session number. When SESSNUM is specified in the DISPLAY statement, the Adabas session number appears in the printed output produced by the ADASEL utility. In addition, the SESSNUM value is stored in any **EXTENDED** or **EXTRACT** output produced by ADASEL's OUTPUT statement. An example of the use of this option can be found in *Selecting Data by Adabas Session Number*, elsewhere in this section.
USERDATA	displays records written to the protection log with a C5 command. The file number of the checkpoint file must be specified in the SELECT statement.
USERID	displays the user ID of the user who added, deleted, or updated the record.
USERTID	displays the TID of the user who added, deleted, or updated the record.
NOHEADER	suppresses the header.
'*text*'	displays the text string.

When you use DISPLAY for selected records, you might see an identifier for the type of Adabas command that generated the PLOG record. The following table gives further information on these identifiers:

Identifier	Description
UPDATE – AI	Update command after-image
BCKUPD – AI	Update command after-image generated by BT
UPDATE – BI	Update command before-image
BCKUPD – BI	Update command before-image generated by BT
NEWREC – AI	New record command after-image

Identifier	Description
BKNREC – AI	New record command after-image generated by BT
DELETE – BI	Delete command before-image
BCKDEL – BI	Delete command before-image generated by BT
SPCUPD – AI	Special update command after-image
BKSPUP – AI	Special update command after-image generated by BT
SPCUPD – BI	Special update command before-image
BKSPUP – BI	Special update command before-image generated by BT

- Examples
- Default Formats
- MU or PE Fields

Examples

Select records that have been modified. Display the text string "The following records were modified:". Then display the fields AA and CC in hexadecimal format and BB in the format defined in the FDT:

```
SELECT UPDATED RECORDS FROM FILE 117
   DISPLAY 'The following records were modified:'
   DISPLAY AA HEX BB CC HEX
END
```

Display the field AA of each new record, along with the user ID and terminal ID of the user who added the record; suppress the header:

```
SELECT NEW RECORDS FROM FILE 211
   DISPLAY AA USERID USERTID NOHEADER
END
```

Select records that have been modified and display the occurrence count, followed by the hexadecimal values of the seventh through twelfth occurrences of the MU field XX:

```
SELECT UPDATED RECORDS FROM FILE 32
   DISPLAY COUNT XX XX7-12 HEX
END
```

Select records that have been modified and display the occurrence count for PE group field XX, followed by the numbers of the YY values in the first through last XX PE group occurrence, followed by all YY values in each XX PE group occurrence (in this example YY is an MU field within the XX PE group):

```
SELECT UPDATED RECORDS FROM FILE 32
   DISPLAY COUNT XX COUNT YY1-N YY1-N(1-N)
END
```

Default Formats

A field is displayed according to its default format:

Alphanumeric	is displayed as entered, with unprintable characters converted to blanks.
Binary	is displayed in unsigned decimal digits (0-9) if the value is less than X'80000000'; otherwise, the value is displayed in hexadecimal notation.
Packed/unpacked	is displayed in decimal digits (0-9), with a leading minus sign if the value is negative.

MU or PE Fields

If *field-name* refers to an MU or a member field of a PE, you can display a single occurrence or a range of occurrences by specifying the index as part of the field name:

```
DISPLAY AA5
```

If you have Adabas 8 or later installed and if extended MU and PE counts are turned on for a file, valid index values range from "1" through "65534"; otherwise the valid index values range from "1" through "191" . In addition, if you specify "N" as the upper limit of an index range, ADASEL displays all occurrences, beginning with the first occurrence in the range.



Note: The use of more than 191 MU fields or PE groups in a file must be explicitly allowed for a file (it is not allowed by default). This is accomplished using the ADADBS MUPEX function or the ADACMP COMPRESS `MUPEX` and `MUPECOUNT` parameters.

You cannot specify the PE name in a DISPLAY statement. To display the entire periodic group, you must specify the name of each field in the group.

If an MU is contained within a PE, both indexes (PE and MU) must be specified. In the index formats shown below, *i* and *j* are the MU indexes; *k* and *l* are the PE indexes. AB refers to a member field of a PE; MB refers to an MU that is a member field of a PE.

Index	Displays . . .
MU*i*	occurrence *i* of the MU
MU*i*-*j*	occurrences *i* through *j* of the MU
MU*i*-N	all occurrences of the MU, starting with occurrence *i*
AB*k*	field AB in occurrence *k* of the PE to which the field belongs
AB*k*-*l*	field AB in occurrences *k* through *l* of the PE
AB*k*-N	field AB in all occurrences of the PE, starting with occurrence *k*
MBk(*i*)	occurrence *i* of MB in occurrence *k* of the PE to which MB belongs
MB*k* - *l*(*i*)	occurrence *i* of MB in occurrences *k* through *l* of the PE
MB*k* - *l*(*i*-*j*)	occurrences *i* through *j* of MB in occurrences *k* through *l* of the PE

Index	Displays . . .
MB*k - l(i*-N)	all occurrences of MB (starting with occurrence *i*) in occurrences *k* through *l* of the PE
MB*k*-N*(i - j)*	occurrences *i* through *j* of MB in all occurrences of the PE (starting with occurrence *k* of the PE)
MB*k*-N*(i*-N)	all occurrences of MB (starting with occurrence *i*) in all occurrences of the PE (starting with occurrence *k* of the PE)

Example:

File 12 contains the following PE:

Level	Name	Descriptive Name	Format	Length	Options	Occ
1	JT	JOB TITLE	A	16	DE,MU	12
1	PA	INCOME			PE	12
2	SA	SALARY	P	6	DE,MU	7
2	BO	BONUS	P	5		

The following are valid *DISPLAY* statements for file 12:

```
SELECT NEW FROM FILE 12
   DISPLAY JT1
END
```

```
SELECT ALL FROM FILE 12
   DISPLAY JT1-5 SA1-5(1-N) BO1-5
END
```

```
SELECT ALL FROM FILE 12
   WITH JT3 ='Programmer' THRU 'Systems Analyst'
   DISPLAY JT3 SA3(1-N) BO3
END
```

```
SELECT UPDATED FROM FILE 12
   DISPLAY JT2-N SA2-N(1-N)
END
```

OUTPUT Instruction

The OUTPUT instruction is used to write the decompressed records from the protection log to an output data set.



Up to 20 output data sets are permitted. The output data set is specified in the EXPA*n* parameter and the DDEXPA*n* job control statement.

Notes:

1. If a BT has been issued for a spanned record update, the primary record will occur before the associated secondary records on the output data set. This differs from the order in which information appears on the PLOG data set, where the secondary records occur before the primary record.
2. LOB/References are not supported.

Example:

Write the before-images of all updated or deleted records to data set DDEXPA1:

```
SELECT BEFORE IMAGE FILE 2
   OUTPUT TO EXPA1
END
```

Select one of the following topics for more information about the specifics of the OUTPUT instruction:

- Output Record Format
- SPANREC Output
- LOGINFO Output
- EXTENDED Output
- EXTRACT Output

- Output Data Set Designation

Output Record Format

The format of the output record depends on whether the SPANREC, LOGINFO, EXTENDED, or EXTRACT parameter is specified. **LOGINFO**, **EXTENDED**, and **EXTRACT** are used to display additional information. The **SPANREC** parameter indicates that alternate headers should be used to handle spanned records. See the following topics , later in this section, for more information

- ***SPANREC Output***
- ***LOGINFO Output***
- ***EXTENDED Output***
- ***EXTRACT Output***



Note: The first record in each block is preceded by the two-byte block length and two bytes of nulls or blanks.

The fields of the protection log record are provided in the order, length, and format in which they are defined in the file's FDT. Alphanumeric fields that are longer than the length defined in the FDT are truncated. Numeric fields that are longer than the length defined in the FDT cause ADASEL to end abnormally.

MUs and PEs are preceded by a one-byte binary field containing the number of occurrences.

Variable-length fields have a default length of zero and are preceded by a one-byte field containing the length of the value (including the length field).

If a field defined with the NC suppression option contains a null value, the null value is decompressed by ADASEL to an empty value (blanks or zeros, depending on the field's format). This type of NC field null processing applies only to ADASEL.

SPANREC Output

When SPANREC is specified, the new spanned record SELH and SELC output headers are used for all EXPA*n* output. DSECTs for the SELH and SELC headers can be found in the Adabas source library. These new spanned record headers are used when any decompressed logical record from the PLOG exceeds the EXPA*n* physical record limitation. In this case, the SELH header will prefix every logical record written to EXPA*n*; subsequent physical records belonging to the same logical record will be prefixed by the SELC header.



Notes:

1. If a BT has been issued for a spanned record update, the primary record will occur before the associated secondary records on the output data set. This differs from the order in which inform-

ation appears on the PLOG data set, where the secondary records occur before the primary record.

2. LOB/References are not supported.

LOGINFO Output

When LOGINFO is specified, the following information is included in each record:

Bytes	Description	
0-1	protection log record length (binary)	
2-3	set to zeros (X'0000')	
4-5	record image type:	
	C'BI'	before-image
	C'AI'	after-image
	C'C5'	user data
	C'ET'	end transaction (ET) data
6-7	Adabas file number (binary)	
8-15	ID of the user (ETID) who added, deleted, or updated the record	
16-19	low-order four bytes of the terminal ID (TID) of the user who added, deleted, or updated the record (from the communications ID; TP monitor users only)	
20-23	Data Storage RABN where the record was stored (binary)	
24-27	data protection block number for the record (binary)	
28-31	timestamp of update (binary; high-order four store-clock (STCK) bytes)	
32-33	decompressed record length (including this length field and the ISN)	
34-37	ISN (binary) or user data from a C5 command	
38	beginning of the decompressed protection log data	

Note: The first record in each block is preceded by the two-byte block length and two bytes of nulls or blanks.

EXTENDED Output

When EXTENDED is specified, the following information is included in each record:

Bytes	Description	
0-1	protection log record length (binary)	
2-3	set to zeros (X'0000')	
4-5	record image type:	
	C'BI'	before-image
	C'AI'	after-image
	C'C5'	user data
	C'ET'	end transaction (ET) data
6-7	Adabas file number (binary)	
8-15	ID of the user (ETID) who added, deleted, or updated the record	
16-23	low-order eight bytes of the terminal ID (TID) of the user who added, deleted, or updated the record (from the communications ID; TP monitor users only)	
24-27	Data Storage RABN where the record was stored (binary)	
28-31	data protection block number for the record (binary)	
32-35	timestamp of update (binary; high-order four store-clock (STCK) bytes)	
36	backout indicator:	
	C'B'	record is a result of a backout
	C' '	normal record
37	Update split indicator:	
	Y	split
	00	not split
38-41	transaction number	
42-45	command counter. The first byte is either zero if a single nucleus or the nucleus ID (1-32) if a cluster nucleus.	
46-47	Adabas session number.	
48-55	date in YYYYMMDD format, displayable character.	
56-61	time in HHMMSS format, displayable character.	
62-63	reserved	
64-65	decompressed record length (including this length field and the ISN)	
66-69	ISN (binary) or user data from a C5 command	
70	beginning of the decompressed protection log data	

Note: The first record in each block is preceded by the two-byte block length and two bytes of nulls or blanks.

EXTRACT Output

When EXTRACT is specified, the following information is included in each record:

Bytes	Description	
0-1	protection log record length (binary)	
2-3	set to zeros (X'0000')	
4-5	record image type:	
	C'BI'	before-image
	C'AI'	after-image
	C'C5'	user data
	C'ET'	end transaction (ET) data
6-7	Adabas file number (binary)	
8-10	ID of the prefix header	
11-13	Adabas version used to identify this prefix header.	
14-15	length of the extract prefix header	
16-17	Adabas session number	
18-21	transaction sequence number	
22-25	command counter. The first byte is either zero if a single nucleus or the nucleus ID (1-32) if a cluster nucleus.	
26-33	ID of the user (ETID) who added, deleted, or updated the record	
34-41	low-order eight bytes of the terminal ID (TID) of the user who added, deleted, or updated the record (from the communications ID; TP monitor users only)	
42-49	security system user ID of the user who added, deleted, or updated the record, or blank if the security system user ID is not known	
50-53	Data Storage RABN where the record was stored (binary)	
54-57	data protection block number for the record (binary)	
58-65	timestamp of update (binary; eight bytes store-clock (STCK))	
66-73	date in YYYYMMDD format, displayable character.	
74-79	time in HHMMSS format, displayable character.	
80	backout indicator:	
	C'B'	record is a result of a backout
	C' '	normal record
81	output indicator:	
	C'A'	ALL - decompressed PLOG record
	C'F'	Field selection from PLOG record
82	Update split indicator:	
	Y	split
	00	not split

Bytes	Description
83-133	reserved 51 bytes
134-135	decompressed record length (including this length field and the ISN)
136-139	ISN (binary) or user data from a C5 command
140	beginning of the decompressed protection log data



Note: The first record in each block is preceded by the two-byte block length and two bytes of nulls or blanks.

Output Data Set Designation

The EXPA*n* parameter identifies the output data set. The value of *n* must match the value in the DDEXPA*n* JCL statement. Valid output data set numbers are 1-20 with no leading zeros:

Valid statement	`OUTPUT TO EXPA3`
Invalid statement	`OUTPUT TO EXPA03`

The same rule applies to the DDEXPAn JCL statement.

Example:

Select all records for file 1. Write decompressed records in which the BA field contains "SMITH" or "SMYTH" to DDEXPA1. Write all others to DDEXPA2:

```
SELECT ALL RECORDS FROM FILE 1
IF BA ='SMITH' OR BA ='SMYTH'
   THEN OUTPUT TO EXPA1
ELSE
   OUTPUT TO EXPA2
END
```

Other, more extended examples of the use of EXPA*n* data sets can be found in ***Selecting Fields for an Output File***, elsewhere in this section.

NEWPAGE Instruction

The NEWPAGE instruction and SKIP instructions control page formatting. The NEWPAGE instruction forces a page eject before displaying the next line of data. In the following example, a page eject occurs every time the value of the BA field changes:

```
SELECT ALL RECORDS FROM FILE 1
   WITH BA EQUAL 'SMITH' THRU 'SMYTH'
IF BA CHANGES THEN DO
   NEWPAGE
   DISPLAY 'NEW NAME' BA BB BC
```

```
   DOEND
END
```

SKIP Instruction

The NEWPAGE instruction and SKIP instructions control page formatting. The SKIP instruction prints the specified number of blank lines before displaying the next line of data. In the following example, two blank lines are printed every time the value of the BA field changes.

```
SELECT ALL RECORDS FROM FILE 1
   WITH BA EQUAL 'SMITH' THRU 'SMYTH'
IF BA CHANGES THEN DO
   SKIP 2 LINES
   DISPLAY 'NEW NAME' BA BB BC
   DOEND
END
```

The END Keyword

The END keyword is a required element of the SELECT parameter and must be specified last.

Examples

This section provides several examples of SELECT parameters.

The following SELECT parameter will output to data set DDEXPA1 all new data records inserted by user ETID1 for November 1st, 2008:

```
SELECT NEW RECORDS FROM USER
    STARTING FROM 20081101/000000
    ENDING AT 20081101/240000
    WITH USERID='ETID1'
    OUTPUT TO EXPA1
END
```

The following SELECT parameter will output to data set DDEXPA2 the after images of all data records updated by any users working from terminals CICS1000 through CICS9999 prior to the end of November 1st, 2008. The output will include extended LOGINFO data:

```
SET GLOBALS NU=50
SELECT NEW RECORDS FROM USER
    ENDING AT 20081101/240000
    WITH USERTID EQ 'CICS1000' THRU 'CICS9999'
    OUTPUT EXTENDED TO EXPA2
END
```

The following SELECT parameter will display fields AA and AB of all data records from file 200 that were inserted, updated, or deleted by user ETID1:

```
SELECT ALL RECORDS FROM FILE 200
    WITH USERID EQ 'ETID1'
    DISPLAY AA AB
END
```

Finally, the following SELECT parameter will display user IDs for all users who updated the data base:

```
SELECT UPDATED RECORDS FROM USER
    DISPLAY USERID
END
```

177 ADASEL Scenarios

- Selecting End Transaction (ET) Information 1310
- Selecting Data by Command Sequence Number 1311
- Selecting Data by Adabas Session Number 1312
- Selecting Fields for an Output File 1313
- Selecting Special Update Commands 1315

This chapter provides some discussion of specific ADASEL scenarios.

Selecting End Transaction (ET) Information

You can select end transaction (ET) information from the protection log (PLOG) using the Adabas ADASEL utility. The SELECT statement parameter, ETCMD, allows you to select only ET information for an ADASEL run. ET information is not included if you specify SELECT ALL.

Note: ET information is not field or transaction-specific; it is user-specific. So if you specify ETCMD in your ADASEL SELECT statement, you must also specify the FROM USER keywords.

Here is an example of the ETCMD parameter in a SELECT statement:

```
SELECT ETCMD FROM USER
DISPLAY ETUSERDA USERTID SESSNUM USERID CMDCOUNT
```

These statements would generate output similar to the following report:

```
SEL072-PROCESSING SELECTION REQUESTS --                                  ↩
                                                                         ↩
SNR   FNR       ISN       USERTID              ET         2013-08-08 10:56:07    ↩
 1                      0000000000000000
      ETUSERDA --- ETUSERDA UNAVAILABLE ---                              ↩
      USERTID                                                            ↩
      SESSNUM 61                                                         ↩
      USERID                                                             ↩
      CMDCOUNT 3                                                         ↩
. . .
```

Specifying `ETUSERDA` displays `--- ETUSERDA UNAVAILABLE ---` if no user data has been written for the ET record that is displayed. If ET user data has been written, that data is displayed for that ET command record.

USERDATA (C5 user data) and ETUSERDA (ET user data) may be requested together, which can result in a display reflecting your selection:

For C5 checkpoint records, `--- ETUSERDA UNAVAILABLE ---` is listed, and `--- USERDATA UNAVAILABLE ---` is listed for ET records.

Selecting Data by Command Sequence Number

You can use the Adabas ADASEL utility's CMDCOUNT keyword to select and display data from the protection log data set (PLOG) based on the command sequence number. The CMDCOUNT keyword can be used in ADASEL's DISPLAY statement as well as in value-criterion of ADASEL's WITH or IF statements.

- When CMDCOUNT is specified in the value criterion of the WITH or IF statements, the command sequence number can be used to select PLOG records based on their CMDCOUNT values.
- When CMDCOUNT is specified in the DISPLAY statement, the command sequence number appears in the printed output produced by the ADASEL utility.

In addition, the CMDCOUNT value is stored in any **EXTENDED** or **EXTRACT** output produced by ADASEL's OUTPUT statement. For more information about the layout of the ADASEL'S OUTPUT instruction, read ***OUTPUT Instruction***, elsewhere in this section; for more information about the EXTENDED and EXTRACT output formats of the OUTPUT instruction, read ***EXTENDED Output*** and ***EXTRACT Output***, elsewhere in this section.

Note: The command sequence number in an Adabas PLOG is not unique. There may be multiple occurrences of this number in your PLOG data set. To ensure accurate data selection and processing, we recommend that you select on the date or time values in addition to the command sequence number. The date and time values are also stored in the EXTENDED output produced by the OUTPUT statement. Read ***Extended Output***, elsewhere in this section, for more information.

In the following ADASEL utility statements, records are selected if the command sequence number is less than 13. Once selected, the transaction user ID (TID) and user ID of the users who added, deleted, or updated the records are displayed in addition to the command sequence number and the Adabas session number of each record.

```
SELECT ALL FROM USER IF CMDCOUNT < 13
DISPLAY USERTID SESSNUM USERID CMDCOUNT
```

Here is sample of the output produced by such ADASEL statements:

```
SEL072-PROCESSING SELECTION REQUESTS --                                    ↩
                                                                           ↩
SNR   FNR       ISN       USERTID               ET         2013-08-08 10:56:07    ↩
 1                      0000000000000000                                  ↩
      USERTID                                                              ↩
      SESSNUM 61                                                           ↩
      USERID                                                               ↩
      CMDCOUNT 3                                                           ↩
```

Selecting Data by Adabas Session Number

You can use the Adabas ADASEL utility's SESSNUM keyword to select and display data from the protection log data set (PLOG) based on the Adabas session number. The SESSNUM keyword can be used in ADASEL's DISPLAY statement as well as in value-criterion of ADASEL's WITH or IF statements.

- When SESSNUM is specified in the value criterion of the WITH or IF statements, the Adabas session number can be used to select PLOG records based on their SESSNUM values.
- When SESSNUM is specified in the DISPLAY statement, the Adabas session number appears in the printed output produced by the ADASEL utility.

In addition, the SESSNUM value is stored in any **EXTENDED** or **EXTRACT** output produced by ADASEL's OUTPUT statement. For more information about the layout of the ADASEL'S OUTPUT instruction, read ***OUTPUT Instruction***, elsewhere in this section; for more information about the EXTENDED and EXTRACT output formats of the OUTPUT instruction, read ***EXTENDED Output*** and ***EXTRACT Output***, elsewhere in this section.

In the following ADASEL utility statements, records are selected if the Adabas session numbers are less than 1000. Once selected, the transaction user ID (TID) and user ID of the users who added, deleted, or updated the records are displayed in addition to the command sequence number and the Adabas session number of each record. In addition, records written to the PLOG with a C5 command are displayed.

```
SELECT ALL FROM USER WITH SESSNUM < 1000
DISPLAY USERTID SESSNUM USERID CMDCOUNT USERDATA
```

Here is sample of the output produced by such ADASEL statements:

```
SEL072-PROCESSING SELECTION REQUESTS --
SNR   FNR       ISN       USERTID              NEWREC - AI 2013-08-08 10:56:07    ↩

 1     10       2001     E4F0F0F100000000                                        ↩

      USERTID U001                                                               ↩

      SESSNUM 61                                                                 ↩

      USERID                                                                     ↩

      CMDCOUNT 4                                                                 ↩

              --- USERDATA UNAVAILABLE ---                                       ↩
```

Selecting Fields for an Output File

The Adabas ADASEL utility can be used for:

- Selecting Fields from the PLOG for the EXPAn Data Set
- Obtaining Counts for MU, PE, and PE/MU Fields

Selecting Fields from the PLOG for the EXPAn Data Set

Field names can be specified on the OUTPUT statement with the LOGINFO format, the **EXTENDED**, the **EXTRACT** format, or with no format (the default). Field name specifications must be made in the same format as field names are specified in ADASEL's **DISPLAY instruction**. In addition, field names should be specified after the EXPA*n* parameter in the order in which you want the fields written.

Consider the following ADASEL utility statement:

```
SELECT ALL FROM 24 OUTPUT EXTENDED TO EXPA1 CA FA1 HA2 HB1(3-5) HB2(6-8)
```

This statement selects all records for file 24 and will use the EXTENDED prefix and write the output to the data set identified in the JCL as EXPA1. The output will include the ISN (by default) and each decompressed field specified in the statement. When you specify fields to be written to EXPA1, the PLOG record output is replaced by the field information requested.

In the original decompressed record, the MU, PE and PE/MU fields include count indicators. However, these count indicators are not included in the output record produced by ADASEL OUTPUT. Instead, each field is treated as a separate entry and is placed in sequential order. This means that, based on the example above, the record written to EXPA1 would have the following format:

```
common-flds extended-flds isn CA FA1 HA2 HB1(3) HB1(4) HB1(5) HB2(6) HB2(7) HB2(8)
```

The ISN is four bytes and each field has the length specified for it in the FDT.

Here is sample output from such a run:

```
SEL070-PROCESSING REQUEST STATEMENTS—
SELECT ALL FROM 24 OUTPUT EXTENDED TO EXPA1 CA FA1 HA2 HB1(3-5) HB2(6-8)
END
SEL071-SELECTION REQUEST NUMBER:   1
SEL072-PROCESSING SELECTION REQUESTS --
SEL073-PROCESSING STATISTICS --
NUM.PROTECTION BLOCK(S) READ:    58
SELECTION REQUEST -- 1
   NUM.REC(S) READ:              4
   NUM.LOGICAL REC(S) READ:      4
   NUM.DS SEG(S) REJECTED:       0
   NUM.REC(S) OUTPUT:            4
SEL074-END OF JOB --
```

Caution: If you specify an MU, PE, or PE/MU field range, and only part of that range exists in the PLOG, ADASEL will generate filler fields in the OUTPUT record for the fields missing in the PLOG. For example, if you specify HA1-99 in your ADASEL input statement, but only four occurrences of field HA exist in the PLOG, then your OUTPUT record would be "…HA1 HA2 HA3 HA4" and 95 more occurrences of the HA field that would be generated and stored, but comprised of a default padding character appropriate to the field type. In addition, if you select a PE, MU or PE/MU field with the 1-N option (thus selecting all of the member fields of the PE, MU or PE/MU field), you may exceed the length of the EXPA*n* output buffer, resulting in error SEL105.

Obtaining Counts for MU, PE, and PE/MU Fields

Specify the CNT parameter before any MU, PE, or MU/PE field in the PLOG record to obtain the count for that field. The CNT parameter is only valid after the EXPA*n* parameter and only before a valid MU, PE, or MU/PE field name.

For an MU or PE field, a two-byte binary counter is output; for a PE/MU field, a four-byte binary counter is output, with the first two bytes used as the counter for the PE and the remaining two bytes used as the counter for the MU.

Consider the following sample OUTPUT statement:

```
OUTPUT EXTENDED TO EXPA1 AA CNT AF AF1-10
```

The output data set produced by this OUTPUT statement would have the EXTENDED prefix, then field AA, a two-byte count for field AF (taken from the decompressed PLOG record), and then the values for occurrences 1 through 10 of field AF. If the PLOG only has four occurrences of field AF, ADASEL would pad the remaining six occurrences with a default padding character.

Selecting Special Update Commands

You can select special update commands (delete and insert commands split or migrated) normally reported as a delete and update. The nucleus still works as usual but sets an indicator in the PLOG records so that ADASEL can identify the split or migrated updates. Parameters are available that must be set to explicitly select this behavior. If they are not set, the standard behavior is used. The parameters SPECUPD and RECSYNC explicitly request this functionality.

- SPECUPD
- RECSYNC

SPECUPD

This is an additional specification to the selection option (ALL, AI, BI, DELETED, NEW, NEWDEL, UPDATED) which will identify any command that originated as an update and flagged as a split or migrated record.

The PLOG records that have been ‘flagged’ as split, will now show SPCUPD instead of UPDATE.

RECSYNC

RECSYNC specifies that the BI and AI images will be displayed together. When the records are split, the AI image may not follow directly after the BI on the PLOG. RECSYNC will cause ADASEL to hold off reporting the BI SPCUPD until it finds the AI counterpart that matches it. It places the record in a buffer and then processes the two records back to back. This means that data displayed or written out might actually be in a different order than these records occurred on the PLOG.

RECSYNC cannot be specified without SPECUPD being specified. The main difference is, that SPECUPD alone will give you the special updates in the exact order they are written on the PLOG. Specifying SPECUPD with RECSYNC will match the BI and AI before reporting the special update, possibly altering the original location of the records on the PLOG.

The RECSYNC process is achieved by buffering the BI until the AI is found. The buffer can retain up to 10 records originally. If ten records are being retained, and an additional record must be added, then the holding table or buffer is extended by ten records. This buffer continues to be extended if necessary.

Each extension of the table requires approximately 320K of storage. Therefore, please keep in mind that using RECSYNC can cause additional memory allocation. However, the initial GETMAIN is not done until actually needed and the first split is encountered. The RECSYNC parameter also introduces a SEL134 message indicating that a BI/AI pair was found or that an unmatched BI or AI was found, depending if the split occurred over a PLOG switch.

RECSYNC replaces RECORDS in the ADASEL syntax, when it is desired.

Below are examples using the new SPECUPD and RECSYNC parameters.

Consider the following sample SELECT statement:

```
SELECT ALL SPECUPD RECSYNC FROM FILE 100
DISPLAY ISN
END
```

Select all records from file 100 that were added, deleted, or updated and display the ISN. Special updates should be identified. Only display the special update commands when a pair (BI/AI) is found.

Example output:

```
SEL134 RECSYNC paired BI/AI follows

SNR   FNR       ISN       USERTID             SPCUPD - BI 2015-04-28 22:30:34
 1    100        6       E4F0F0F600000000
      ISN     6

SNR   FNR       ISN       USERTID             SPCUPD - AI 2015-04-28 22:30:34
 1    100        6       E4F0F0F600000000
      ISN     6

SNR   FNR       ISN       USERTID             UPDATE - BI 2015-04-28 22:30:34
 1    100        7       E4F0F0F700000000
      ISN     7

SNR   FNR       ISN       USERTID             UPDATE - AI 2015-04-28 22:30:34
 1    100        7       E4F0F0F700000000
      ISN     7                                                              ↩
```

Consider the following sample SELECT statement:

```
SELECT ALL SPECUPD RECORDS FROM FILE 3
DISPLAY AE
END
```

Select all records from file 3 that were added, deleted, or updated. Display the special updates as they occur on the PLOG. The BI and AI may have other PLOG records between them, depending on the nucleus activity and amount of thread switches.

Example output:

```
SNR   FNR       ISN       USERTID             UPDATE - BI 2015-03-26 15:55:07
 1     3         1        C014C014
      AE

SNR   FNR       ISN       USERTID             UPDATE - AI 2015-03-26 15:55:07
 1     3         1        C014C014
      AE

SNR   FNR       ISN       USERTID             UPDATE - BI 2015-03-26 15:55:07
 1     3         6        C018C018
      AE

SNR   FNR       ISN       USERTID             UPDATE - AI 2015-03-26 15:55:07
 1     3         6        C018C018
      AE

SNR   FNR       ISN       USERTID             SPCUPD - BI 2015-03-26 15:55:07
 1     3         5        C019C019
      AE

SNR   FNR       ISN       USERTID             SPCUPD - AI 2015-03-26 15:55:07
 1     3         5        C019C019
      AE

SNR   FNR       ISN       USERTID             SPCUPD - BI 2015-03-26 15:55:07
 1     3         2        C021C021
      AE

SNR   FNR       ISN       USERTID             SPCUPD - AI 2015-03-26 15:55:07
 1     3         2        C021C021
      AE                                                                    ↩
```

178 JCL Requirements and Examples

- Requirements to Run ADASEL JCL 1320

This section describes the job control information required to run ADASEL on the z/OS operating system and shows an example job stream.

Requirements to Run ADASEL JCL

Data Set	DD Name	Storage	More Information
Protection log	DDSIIN	tape/ disk	Sequential log
Optional FDT input	DDEBAND	tape/ disk	Either DDEBAND or DDSAVE can be specified, but not both.
Optional save tape with FDT input	DDSAVE	tape/ disk	Either DDEBAND or DDSAVE can be specified, but not both. This DD is required if a save tape is used in the ADASEL run.
Selected data	DDEXPAn	tape/ disk	Output by ADASEL
Associator	DDASSORn	disk	
ADARUN parameters	DDCARD	reader	*Operations*
ADASEL parameters	DDKARTE	reader	
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADASEL messages	DDDRUCK	printer	*Messages and Codes*

ADASEL JCL Example

```
//ADASEL    JOB
//*
//*   ADASEL:
//*      SELECT PROTECTION DATA
//*
//SEL       EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD          <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1    <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1    <=== DATA
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1    <=== WORK
//DDSIIN    DD   DISP=OLD,DSN=EXAMPLE.DByyyyy.PLOG5     <=== OUTPUT ADARES
//*                                                             PLCOPY
//DDEXPA1   DD   DISP=(,CATLG),DSN=EXAMPLE.DByyyyy.EXPA1, <= OUTPUT ADASEL
//                SPACE=(TRK,(10,5),RLSE),VOL=SER=vvvvvv,UNIT=uuuu
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADASEL,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
```

```
//DDKARTE   DD   *
*
*    SELECT UPDATES FOR FILE NUMBER 1
*
SELECT ALL FROM FILE 1
  OUTPUT TO EXPA1
END
/*
```

Refer to ADASEL in the JOBS data set for this example.

XX ADAULD Utility: Unload Files

This chapter covers the following topics:

Functional Overview
UNLOAD FILE: Unload Specified File
ADAULD Input Processing
ADAULD Output Processing
ADAULD User Exit 9
JCL Requirements and Examples

179 Functional Overview

The ADAULD utility unloads an Adabas file. Adabas files are unloaded to:

- permit the data to be processed by a non-Adabas program. In this case, the file must also be decompressed after unloading using the DECOMPRESS function of the ADACMP utility.
- create one or more test files, all of which contain the same data. This procedure requires that a file be unloaded, and then reloaded as a test file having a different file number.
- change the field definition table (FDT). This requires that the file be unloaded, decompressed, compressed using the modified field definitions, and reloaded. If the ADADBS utility is used to add field definitions to a file, the file does not need to be unloaded first.

The sequence in which the records are unloaded may be

physical	the order in which they are physically positioned within Data Storage.
logical	a sequence controlled by the values of a user-specified descriptor.
ISN	ascending ISN sequence.

Selection criteria (SELCRIT/SELVAL) are optionally used to indicate a subset of records to be unloaded:

- If no such criterion is provided, all records are unloaded in physical sequence.
- If a selection criterion is specified but no sort sequence (SORTSEQ), the specified records are unloaded in ISN sequence.
- If both a selection criterion and a sort sequence are provided, the selected records are sorted in the Work pool area of the nucleus and are unloaded in the specified sort sequence.
- If no records that match the selection criteria are found, ADAULD creates a file containing only the FDT and issues condition code 4 in register 15.

The unloaded record output is in compressed format. The output records have the same format as the records produced by the ADACMP utility.

When using the MODE=SHORT option, descriptor entries (which are required to create the normal index and upper index for the file) are omitted during the unload process. This reduces the time required for unloading. Note, however, that output created using MODE=SHORT has a different FDT from the same file unloaded without MODE=SHORT, since all descriptor information is removed.



Notes:

1. An interrupted ADAULD UNLOAD FILE run must be reexecuted from the beginning.
2. Logically deleted fields will appear in data unloaded by the ADAULD utility.

180 UNLOAD FILE: Unload Specified File

- Essential Parameter 1328
- Optional Parameters and Subparameters 1328
- Enhanced Temp and Sort Processing 1334
- Examples 1334

```
ADAULD  [UNLOAD]  FILE= file-number
                  [CODE = cipher-key ]
                  [DDISN]
                  [ERRLIM = { error-threshold-count | 1 } ]
                  [ETID = multiclient-file-owner-id ]
                  [LPB = prefetch-buffer-size ]
                  [LRECL = { maximum-compressed-length | 4000 } ]
                  [MODE = SHORT]
                  [NOUSERABEND]
                  [NUMOUT = { 1 | 2 } ]
                  [NUMREC = number ]
                  [PASSWORD = file-password ]
                  [SAVETAPE]
                      [PLOGNUM = plog-number , { SYN1 | SYN4 } = plog-block-number ]
                      [TEMPDEV = device-type ]
                      [TEMPSIZE = size ]
                  [SELCRIT = ' selection-criteria ', SELVAL = ' values-for-selection-criteria ' ]
                      [STARTISN = value ]
                  [ SORTSEQ = { descriptor [,MU] [,NU]
                              { ISN [, STARTISN = value ] } ]
                  [TEST]
                  [UTYPE = { EXU | EXF } ]
```

This chapter describes the syntax and parameters of the UNLOAD FILE function.

Essential Parameter

FILE

FILE specifies the number of the file to be unloaded. Neither the checkpoint file nor the security file can be unloaded.

Optional Parameters and Subparameters

CODE: Cipher Code

If the file to be unloaded is ciphered, CODE *must* supply the appropriate cipher code.

DDISN: Create DDISN Output File of Unloaded ISNs

Specifying the DDISN parameter instructs ADAULD to write the list of unloaded ISNs to the sequential output file DDISN. DDISN is structured so that it can be used as input to ADALOD UPDATE for the purpose of deleting the unloaded records.

If the DDISN keyword is specified

- but the DDISN file is missing in the JCL, ADAULD terminates with error 081.
- and SORTSEQ specifies a hyperdescriptor or descriptor that refers to a multiple-value field, ADAULD terminates with error 133 because the DDISN may contain duplicate ISNs.

ERRLIM: Error Threshold

ERRLIM sets the maximum number of nucleus response codes accepted by ADAULD before operation terminates. The default setting is one, which means that the first error terminates ADAULD with error 124.

The ERRLIM value may be set higher than one to tolerate conditions that occur intermittently such as response code 255 (ADARSP255 - all attached buffers allocated). In this case, the utility terminates with return code 8 and no user abend. The output file of ADAULD can be used, although records may be missing depending on the nucleus response code returned.

ETID: Multiclient File Owner ID

When unloading multiclient files, the ETID parameter can be used to restrict UNLOAD processing to only the records owned by the specified user. If the ETID parameter is omitted, all records are unloaded.

If the SELCRIT/SELVAL parameters are specified for a multiclient file, the ETID parameter *must* also be specified.

LPB: Prefetch Buffer Size

LPB specifies the size of the internal prefetch buffer. The maximum value is 32767 bytes.

By default, ADAULD attempts to make the prefetch buffer as large as possible to achieve the best performance. The LPB parameter gives the user the option of making the prefetch buffer smaller. This might be advisable, for example, if heavy use of prefetching causes ADAULD to consume too much nucleus resource relative to other users.

The default value depends on the length of the intermediate user buffer set by the ADARUN LU parameter. ADAULD subtracts the space required to accommodate the Adabas control information (108 bytes) and the specified maximum compressed record length (LRECL) from the LU value to determine the default LPB value. The result must be equal to or less than the maximum value allowed for LPB; that is, 32767 bytes.

The default value for LU is set to 65535 bytes, the maximum size, to accommodate the record buffer of utilities such as ADAULD that need the nucleus. If the LU value is too small, ADAULD may reduce the specified value for the LPB parameter.

LRECL: Maximum Compressed Record Length

LRECL specifies, in bytes, the maximum compressed record length (including DVT) to be returned.

This length is used as an Adabas record buffer length. If this value is too small, a response code 53 (ADARSP053) occurs. The default is 4000 bytes; the maximum allowed is 32760 bytes.



Caution: Please be sure to verify that the output data set is defined so it can store a record in this size.

MODE=SHORT: Exclude Descriptor Information

This parameter indicates whether the descriptor information used to build the normal index and upper index are to be included in the output.

If MODE=SHORT is specified, no descriptor information will be unloaded, and all descriptor information is stripped from the field definition table (FDT) when it is written to the output data set.

If the output is to be used as direct input to the ADALOD utility, the file will have no descriptors.

In the case of superdescriptors, MODE=SHORT unloads them as superfields. If the output is used as direct input to ADALOD, the loaded file will have superfields.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

NUMOUT: Number of Output Files

NUMOUT specifies the number of output files to be produced. If the number is greater than one, user exit 9 must be used to control DDOUT1 or DDOUT2 output file selection. For additional information, see the *Adabas DBA Reference* documentation. Permitted values are 1 (default) and 2.

NUMREC: Number of Records to Be Unloaded

NUMREC limits the number of records to be unloaded. No limit will be in effect if the parameter is omitted.

PASSWORD: File Password

The PASSWORD parameter must be specified if the file to be unloaded is password-protected.

PLOGNUM: Protection Log Number

When SAVETAPE is specified and an online save tape is to be used as input to ADAULD, the corresponding protection log is expected as a DDPLOG sequential input data set. You can specify PLOGNUM or SYN1 or SYN4 to override the tape information.

SAVETAPE

SAVETAPE is used to unload a file from a save tape. This is useful when moving a file from a save tape with one block size to a database with another, or when using a file from a save tape in one or another test environment.

If an online save tape is used, the TEMPDEV parameter must also be specified. PLOGNUM and SYN1 or SYN4 may be specified for online save tapes to override the information included on the tape.

For more information, see the section **Processing a Save Tape as Input**.

The SORTSEQ and SELCRIT parameters may not be used with SAVETAPE.

The ETID parameter may not be used with SAVETAPE. User exit 9 must be used to select records for a particular client of a multiclient file. For more information, see the section **ADAULD User Exit 9**.

If the file to be unloaded from the save tape is ciphered, the CODE parameter must be specified as usual.



Note: Special SAVETAPE functions are available for use with the Adabas Delta Save Facility. For more information, see the *Adabas Delta Save Facility Facility* documentation.

SELCRIT: Selection Criterion

The SELCRIT parameter may be used to restrict the unloaded records to those which meet the selection criterion provided. The selection criterion must be provided using the search buffer syntax, as described in the *Adabas Command Reference* documentation.

For multiple criteria, you can specify each criterion with a separate ADAULD SELCRIT statement, as follows:

```
ADAULD SELCRIT ='AA, 20, A, D,'
ADAULD SELCRIT ='AB, 10, A.'
```

ADAULD concatenates this to:

```
'AA, 20, A, D, AB, 10, A.'
```

The values that correspond to the selection criterion must be provided using the SELVAL parameter.



Caution: If the value buffer, SELVAL parameter value, exceeds the total length of the fields defined in the search buffer, SELCRIT parameter value, those bytes in the value buffer beyond the total length of the search buffer fields will be ignored and not used in the search. So be sure to specify any search buffer field lengths in the SELCRIT parameter that exceed any search buffer field default lengths. Failure to do this could result in the wrong records being unloaded or the records that were wanted, not being found.

SELVAL: Values for Selection Criteria

SELVAL specifies the values corresponding to the selection criteria specified with the SELCRIT parameter. The value formats are the same as those used for the Value Buffer, as described in the *Adabas Command Reference* documentation.

Values can be on multiple lines. Packed decimal or binary values can be in hexadecimal format, as shown in the following example:

```
SELVAL='PARIS   '
SELVAL=X'00149C'
SELVAL='AB100'
```

Caution: If the value buffer, SELVAL parameter value, exceeds the total length of the fields defined in the search buffer, SELCRIT parameter value, those bytes in the value buffer beyond the total length of the search buffer fields will be ignored and not used in the search. So be sure to specify any search buffer field lengths in the SELCRIT parameter that exceed any search buffer field default lengths. Failure to do this could result in the wrong records being unloaded or the records that were wanted, not being found.

SORTSEQ: Unload Sequence

SORTSEQ specifies the sorting sequence for unloaded ISNs. If SORTSEQ is not specified, ISNs are unloaded in physical sequence.

If a descriptor name is specified, the records are unloaded in the ascending logical sequence of the descriptor values. You can specify the name of a descriptor, subdescriptor, super-descriptor, or hyperdescriptor. *Do not* refer to a field in a periodic group.

- MU *must* be specified if the descriptor name refers to a multiple-value field. In this case, the same record is unloaded once for each different value for the descriptor in the record in ascending value order. If MU is not specified (the default), ADAULD rejects MU descriptors and issues an error message.
- NU *must* be specified if the descriptor name refers to a field defined with the null suppression (NU) option. In this case, records of the descriptor that contain null values are not unloaded. If NU is not specified (the default), ADAULD rejects NU descriptors.

Note: Even when the descriptor field is not null suppressed, the record is *not* represented in the inverted list if the descriptor field or a field following it has never been initialized (held a value). Therefore, the record will be dropped when the utility is executed.

If SORTSEQ=ISN is specified, the records are unloaded in ascending ISN sequence.

If both SELCRIT/SELVAL and SORTSEQ are specified, the records are sorted in the Work pool area of the nucleus. Therefore, the ADARUN LS and LWP session parameters must provide enough space; see the *Adabas Operations* documentation for descriptions of the LS and LWP parameters.

STARTISN: Starting ISN

STARTISN is used with the SELCRIT/SELVAL and SORTSEQ parameters to restrict the unloaded records according to ISN. Specifying STARTISN alone is not allowed.

- Specifying STARTISN with SELCRIT/SELVAL causes all records with ISNs equal to or greater than the STARTISN-specified value *and* with field contents satisfying the SELCRIT/SELVAL criterion to be unloaded in ascending ISN sequence by descriptor name.
- Specifying STARTISN with SORTSEQ=ISN unloads all records beginning with the STARTISN-specified record in ISN sequence.

SYN1|SYN4: Starting Block Number

When SAVETAPE is specified and an online save tape is to be used as input to ADAULD, the corresponding protection log is expected as a DDPLOG sequential input data set.

You can specify PLOGNUM or SYN1 or SYN4 to override the tape information.

TEMPDEV: Temporary Storage Device Type

When SAVETAPE is specified and an online save tape is to be used as input to ADAULD, a temp data set is used to store intermediate data during processing. The TEMPDEV parameter indicates the device type to be used for the temp data set. Refer to ***Enhancements to TEMP and SORT Processing in ADAINV, ADALOD and ADAULD*** for more details on specifying the TEMPDEV parameter.

The block size of the temp data set must be at least as large as the largest Data Storage block size of the file to be unloaded, plus 16 bytes.

TEMPSIZE: Temporary Storage Size

TEMPSIZE specifies the size of the temp data set for the file. The size can be either in cylinders or blocks (followed by a "B").

The temp data set must be large enough to store all Data Storage blocks from the protection log. In the worst case scenario, it must have as many blocks as the file has Data Storage blocks but need not be larger than the PLOG data set. If the temp data set is too small, ADAULD error-136 (temp data set too small) is returned. Refer to ***Enhancements to TEMP and SORT Processing in ADAINV, ADALOD and ADAULD*** for more details on specifying the TEMPSIZE parameter.

TEST: Test Syntax

This parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

UTYPE: User Type

The user type to be in effect for the unload process.

- If EXU (the default) is specified, the file cannot be updated, but other users can read the file.
- If EXF is specified, only ADAULD can use the file; no other users can read or write the file.

Enhanced Temp and Sort Processing

When the TEMPSIZE parameter is omitted, various enhancements to TEMP processing are available to the user. See *Enhancements to TEMP and SORT Processing in ADAINV, ADALOD and ADAULD* for details.

Examples

Example 1:

```
ADAULD FILE=6
```

File 6 is to be unloaded. The records are to be unloaded in the sequence in which they are physically positioned in Data Storage.

Example 2:

```
ADAULD FILE=6,SORTSEQ=AA
```

File 6 is to be unloaded. The values for the descriptor AA are to be used to control the sequence in which the records are to be unloaded.

Example 3:

```
ADAULD FILE=6,SORTSEQ=ISN
```

File 6 is to be unloaded. The records are to be unloaded in ascending ISN sequence.

Example 4:

```
ADAULD FILE=6,SORTSEQ=ISN,STARTISN=10000
```

File 6 is to be unloaded. The records are to be unloaded in ascending ISN sequence. Only records which have an ISN equal or greater than 10000 are to be unloaded.

Example 5:

```
ADAULD FILE=6,SORTSEQ=AB,MODE=SHORT
```

File 6 is to be unloaded. The values for the descriptor AB are to be used to control the sequence in which the records are to be unloaded. The entries used to create the normal index and upper index are not to be unloaded. All descriptor information is removed from the field definition table (FDT) in the output.

Example 6:

```
ADAULD FILE=6,SELCRIT='AA,1,S,AA,2.',SELVAL='AMM'
```

File 6 is to be unloaded. Only records with AA=A through MM are to be unloaded. The records are returned in ISN sequence.

Example 7:

```
ADAULD FILE=6,UTYPE=EXF
```

File 6 is to be unloaded. The user type is indicated as EXF which locks the file during unload processing, preventing other users from reading or writing the file.

181 ADAULD Input Processing

- Processing a Save Tape as Input 1338

ADAULD is used to unload an Adabas file from

- a database; or
- a save tape (if the SAVETAPE keyword is specified).

Processing a Save Tape as Input

If a save tape is used as input, a DDSAVE sequential file is expected. Database or file save tapes created online and offline are acceptable. The save tape must have been created using ADASAV version 5.1 or above.

The ADARUN DBID specified for the ADAULD run must match the DBID found on the save tape.

If the file has hyperdescriptors defined, the corresponding hyperdescriptor exits must be specified in the ADARUN parameters for ADAULD. If the hyperdescriptor exit routines are no longer available, the file must be unloaded with MODE=SHORT specified. See the *Adabas DBA Reference* documentation for more information about hyperdescriptor exits.

For an online save tape:

- the corresponding protection log is expected as a DDPLOG sequential input data set.
- a temp (DDTEMPR1) data set is required as intermediate storage for the Data Storage blocks on the protection log. The TEMPSIZE and TEMPDEV parameters must be specified.

If an online save tape created using ADASAV version 5.1 is to be used, the additional parameters PLOGNUM and SYN1 or SYN4 must be specified:

- PLOGNUM specifies the number of the nucleus protection log used while the ADASAV SAVE operation was active; and
- SYN1 or SYN4 specifies the block number containing the SYN1 or SYN4 checkpoint at which the corresponding ADASAV SAVE operation began.

For online save tapes created using ADASAV version 5.2 or above, this information is included on the tape. You can specify PLOGNUM or SYN1 or SYN4 to override the tape information.

The ADAULD utility protocol on DDDRUCK displays a short header indicating the kind of save tape encountered, when it was created, the version of ADASAV used to create it, the database ID found on the save tape, and for online save tapes, the session number of the corresponding protection log and the block number of the SYN1/SYN4 checkpoint:

```
A D A R E P    Vv.r   SMs     DBID = nnnnn  STARTED           yyyy-mm-dd   hh:mm:ss ↩


PARAMETERS:
-----------

ADAULD UNLOAD FILE=3, SAVETAPE

************************************************************
*                                                          *
*  UNLOAD FROM            ONLINE DATABASE SAVE             *
*  CREATED AT             yyyy-mm-dd  hh:mm:ss             *
*  BY ADASAV VERSION      V vr                             *
*  DBID                   nnnnn                            *
*  DSID                   1 / 0 /   yyyy-mm-dd  hh:mm:ss   *
*  PLOG SESSION NR        17                               *
*  SYN1 BLOCK NR          137                              *
*                                                          *
************************************************************
```

ADAULD first reads the file control block (FCB) and file definition table (FDT) from the save tape. Then:

- for offline save tapes, ADAULD scans the tape to find the file's Data Storage RABNs, extracts the Data Storage records, and for each Data Storage record, generates the descriptor values according to the FDT.
- for online save tapes, ADAULD scans the protection log and copies the latest version of each Data Storage block of the relevant file to the temp data set. The location of a Data Storage block on the temp data set is maintained in a directory in main memory. Then, ADAULD scans the save tape for Data Storage blocks of the file. If more recent versions of Data Storage blocks exist on the temp data set, they are actually unloaded to DDOUT1 or DDOUT2. Note that in this case, two parallel tape units are required: concatenating the save tape and the protection log as for ADASAV RESTONL is not possible.

After opening the DDSAVE and DDPLOG input data sets, ADAULD cross-checks to ensure that the input tapes are correct. If an invalid save tape is encountered, ADAULD terminates and displays error-134 (invalid save tape supplied). If an invalid protection log tape is encountered, ADAULD terminates with error-135 (invalid protection log supplied).

182 ADAULD Output Processing

ADAULD unloads the records in the specified sequence. The unloaded records are written to one or both of two sequential data sets: DDOUT1 and DDOUT2. Writing to these output data sets is controlled by user exit 9.

The records output are identical in format to the output produced by the ADACMP utility unless the MODE=SHORT option is used, in which case the descriptor entries required for the normal index and upper index are omitted and the descriptor information is removed from the Adabas FDT. The ISN of the record immediately precedes the compressed data record, and is provided as a four-byte binary number.

Specifying the DDISN parameter instructs ADAULD to write the list of unloaded ISNs to a sequential output file DDISN. Only one DDISN file is created, containing the superset of ISNs written to either or both of DDOUT1 and DDOUT2. ISNs that are rejected by user exit 9 are not written to DDISN.

DDISN is structured so that it can be used as input to ADALOD UPDATE for the purpose of deleting the unloaded records.

The number of ISNs written to DDISN is displayed in the ADAULD statistics on the DDDRUCK utility protocol:

```
A D A U L D  Statistics
-----------------------

Number of Output Data Sets        =  1
Number of Requested Records       =  16777215
STARTISN                          =  0
Options                           =  DVT
Unload Sequence                   =  PHYS SEQ

Number of Records Read            =         307
Number of Records Written         =         307
Number of Record Segments Read    =         777
Number of Record Segments Written =         777
Records Written to DDOUT1         =         307
Records Written to DDOUT2         =           0
Records Rejected by USEREXIT-9    =           0
Number of ISNS Written to DDISN   =         307
```

If the DDISN parameter is specified, the number of ISNs written to DDISN should always be the number of records read minus the number of records rejected by user exit 9.

The ISNs on the DDISN file are ISNs as visible to applications; that is, the internal ISN as stored in a Data Storage record plus MINISN-1.

183 ADAULD User Exit 9

User exit 9 is called (when present) for each record selected before writing the record to the output data set. The user exit is supplied with the record address, and returns an action code as follows:

1	Write record to DDOUT1;
2	Write record to DDOUT2;
3	Write record to DDOUT1 and DDOUT2;
I	Ignore this record.

The above data sets must have the same block size. See the *Adabas DBA Reference* documentation for more information about user exits.

184 JCL Requirements and Examples

- Requirements to Run ADAULD JCL 1346

This section describes the job control information required to run ADAULD on the z/OS operating system and shows examples of job streams.

Note: When you specify a dataset (such as ASSO) that is not utilized by ADAULD, it will be ignored.

Requirements to Run ADAULD JCL

Data Set	DD Name	Storage	More Information
Unloaded data	DDOUT1	tape/ disk	Output by ADAULD (see note)
Unloaded data	DDOUT2	tape/ disk	Output by ADAULD (see note)
Unloaded ISNs	DDISN	tape/ disk	Required with DDISN
Save tape	DDSAVE	tape/ disk	Required with SAVETAPE
Sequential PLOG	DDPLOG	tape/ disk	Required for online save tapes
Temp area	DDTEMPR1	disk	Required for online save tapes
Recovery log (RLOG)	DDRLOGR1	disk	Required for ADARAI
ADAULD messages	DDDRUCK	printer	*Messages and Codes*
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADARUN parameters	DDCARD	reader	*Operations*
ADAULD parameters	DDKARTE	reader	

Note: DDOUT1 and DDOUT2 must have the same block size; otherwise, an ADAULD error will occur. DDOUT2 is required only if NUMOUT=2 is specified.

ADAULD JCL Examples

Unload a File

```
//ADAULD    JOB
//*
//*   ADAULD:
//*      UNLOAD A FILE
//*
//ULD       EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD       <=== ADABAS LOAD
//*
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <=== WORK
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
```

```
//DDOUT1    DD   DISP=(,CATLG),DSN=EXAMPLE.DByyyyy.OUT1,UNIT=DISK, <===
//          VOL=SER=DISK01,SPACE=(TRK,(200,10),RLSE)
//DDCARD    DD   *
ADARUN PROG=ADAULD,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADAULD FILE=1,SORTSEQ=AA
/*
```

Refer to ADAULD in the JOBS data set for this example.

Unload a File from Save Tape Created Offline

```
//ADAULDS   JOB
//*
//*   ADAULD:
//*      UNLOAD A FILE  FROM AN OFFLINE SAVE TAPE
//*
//ULD       EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD      <=== ADABAS LOAD
//*
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <=== WORK
//DDSAVE    DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.SAVE   <=== SAVE DATASET
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDOUT1    DD   DISP=(,CATLG),DSN=EXAMPLE.DByyyyy.OUT1,UNIT=DISK, <===
//          VOL=SER=DISK01,SPACE=(TRK,(200,10),RLSE)
//DDCARD    DD   *
ADARUN PROG=ADAULD,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADAULD FILE=1,SAVETAPE
/*
```

Refer to ADAULDS in the JOBS data set for this example.

XXI ADAWRK Utility: Work Area Recovery Reports

The ADAWRK utility can be used to produce reports from records in the autorestart area of Work part 1. This information can be used when the database autostart fails and the database will not come up. The data on the ADAWRK reports can help you determine whether:

- You should run a restore/regenerate (ADASAV RESTORE utility function followed by the ADARES REGENERATE utility function) of the database, which can be time-consuming.
- Excluding specific files from the autorestart using AREXCLUDE and then restoring/regenerating only these single files would be beneficial.
- The database can be quickly repaired so it can be started and functional more quickly.

This chapter covers the following topics:

Functional Overview
Utility Syntax
Report Descriptions
JCL Requirements and Examples

185 Functional Overview

- Replication-Related Data Processing 1352
- ADAWRK EXU User Processing 1353

When a database autostart fails and the database will not start up, you need to know what can be done to get the database back up and running quickly, with a minimum amount of lost data and with enough information to retrieve any lost updates. The ADAWRK utility can help you make this determination. It can produce the following reports:

- The **Environment report** is always produced, regardless of the ADAWRK parameters specified, and it identifies the ADAWRK parameters used to produce the report as well as the Work data sets used for the report.
- The **Summary report** is produced by default and provides an overview of the data in the autorestart area of Work part 1.
- The **File report** is an optional report that provides a breakdown of the data in the autorestart area of Work part 1 by file.
- The **Transaction report** is an optional report that provides a breakdown of the data in the autorestart area of Work part 1 by transaction.
- The **Checkpoint record report** is an optional report that lists the checkpoint records and associated data within the autostart area of Work part 1.
- The **Replication-related reports** are optional reports that report on data that may need to be replicated.

Samples and more detailed descriptions of all reports and checkpoint records are provided elsewhere in this chapter.

The ADAWRK utility will only report on transactions that may need to be corrected as part of the autorestart processing logic.

You can filter all of the Work part 1 autorestart area records processed and reports produced in an ADAWRK run by communication ID, ETID, user ID, and file number.

For information on Adabas database autorestart processing, read *Recovery/Restart Design*, in *Adabas DBA Tasks Manual*

Replication-Related Data Processing

To internally support replication-related protection records, ADAWRK determines and reports the replication restart point (RRP) of the transactions on Work part 1. The RRP is the timestamp of the start of the oldest committed transaction that was replicated but for which Adabas has not yet received confirmation from the Event Replicator Server. The RRP should be taken into consideration when you are trying to identify transactions that may have been completed and replicated, or scheduled to be replicated, but for which replication processing may have been interrupted. Such transactions may need to be resent to the Event Replicator as part of autorestart processing by the nucleus.

If there is more than one Work data set (from a cluster database), the RRP is located on every Work data set. If, while searching for the RRP, ADAWRK detects that one Work data set has wrapped, then it determines that it has not found the RRP on that Work data set and that replication data may have been lost.

ADAWRK EXU User Processing

When a user runs as an exclusive-update (EXU) user, ADAWRK groups the user's updates under the associated communication ID.

Note: An EXU user is one who specifies the keyword EXU (omitting the keyword UPD) in the record buffer of the OP command to Adabas and does not issue ET commands. No other user can update the file(s) over which the EXU user has exclusive-update control. An EXU user does not have transactional support and cannot back out recent updates. For more information on EXU users and EXU processing, read *Competitive Updating* in *Adabas Concepts Manual* and read *Exclusive Control Updating* in the .

In the Transaction report, ADAWRK lists only those EXU user updates that may need to be corrected as part of the session autorestart processing. The last such update may be incomplete and subject to being backed out by Adabas.

In the Replication report, ADAWRK lists only those replicated EXU user updates that the Event Replicator server has not confirmed as successfully processed. Some or all updates may appear in both the Transaction and Replication reports.

186 Utility Syntax

This is the syntax of the ADAWRK utility. Sample JCL using this syntax is provided elsewhere in this chapter.

```
ADAWRK  [ABEND34]
        [CHECKPOINT = {YES | NO }]
        [CMID = id1 [, id2 ] ... [, id24 ] ]
        [ETID = etid1 [, etid2 ] ... [, etid32 ] ]
        [FILES = fn1 [, fn2 ] ... [, fn64 ] ]
        [FORCE = {YES | NO }]
        [LWP = nnnK | 1024K ]
        [NOPPT]
        [NOUSERABEND]
        [REPLICATION = { NO | YES | DETAIL | FULL }]
        [REPORTFILE = {YES | NO }]
        [SECUID = secuid1 [, secuid2 ] ... [, secuid24 ] ]
        [SUMMARY = {NO | YES }]
        [TEST]
        [TIMEZONE = {MACHINE | LOCAL | {+ | -} nn } ]
        [TRANSACTIONS = { NO | YES | DETAIL | FULL} ]
        [USERID = id1 [, id2 ] ... [, id24 ] ]
```

ADAWRK can be specified alone, without any parameters, to produce a summary report (SUMMARY=YES is the default). You can optionally customize the reports produced by ADAWRK by adding other parameter values. Each parameter is described here:

ABEND34

Use the ABEND34 parameter to change a user abend 35 to a user abend 34 when an ADAWRK utility error occurs. This ensures that a dump is produced when the utility terminates abnormally.

CHECKPOINT

Use the CHECKPOINT parameter to indicate whether or not checkpoints found in the autorestart area of Work part 1 should be printed in the ADAWRK report output. Valid values are "YES" and "NO". A value of "YES" indicates that checkpoint records found should be printed; a value of "NO" indicates that they should not. The default is "NO".

CMID

Use the CMID parameter to specify up to 24 32-byte communication IDs in hexadecimal format. Only Work part 1 autorestart area records with communication IDs equal to the values specified on the CMID parameter will be processed by the ADAWRK utility and printed on its reports.

ETID

Use the ETID parameter to specify up to 32 ETIDs in character format. ETIDs must be one to eight bytes long. When ETIDs are specified, only Work part 1 autorestart area records for those ETIDs are processed by the ADAWRK utility and printed on its reports.

FILES

Use the FILES parameter to specify up to 64 file numbers that should be included in the report. Only Work part 1 autorestart area records for files listed in the FILES parameter will be processed by the ADAWRK utility and printed on its reports. However, if the FILES parameter is not specified, all files in the database will be processed by default.

You can specify a range of file numbers for this parameter if needed. For example, FILES=2-20 indicates that all files with file numbers between and including 2 and 20 should be processed by the utility.

FORCE

Use the FORCE parameter to indicate how ADAWRK processing should proceed when inconsistencies in the autorestart area of Work part 1 are encountered. Valid values are "YES" and "NO". A value of "YES" indicates that the ADAWRK utility should continue to attempt to interpret the data, without abending; a value of "NO" indicates that inconsistencies in the autorestart area of Work part 1 will result in the termination of the utility with an appropriate message and abend.

LWP

Use the LWP parameter to specify the size of the work pool used internally by the ADAWRK utility. Valid values are in the range 100K - 1048576K (or 1 Gb). The default is 1024K (1Mb). The LWP must be specified in kilobyte units and if the "K" is not present in the specification, an error will result. For example, LWP=500K is a valid specification, but LWP=500 is not.

NOPPT

Use the NOPPT parameter to indicate that the Associator (ASSO) data set for the database should not be opened by the utility. If you specify this parameter, the ASSO data set will not

be opened. When this parameter is not specified, and if the ASSO data set is provided to the utility, the utility will use the PPT.

This parameter provides a workaround in situations where the Work data sets are available, but the Associator is not.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.



Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

REPLICATION

Use the REPLICATION parameter to activate the production of the Replication report, the Replication Summary report, and additional replication statistics in the other ADAWRK reports. Valid values are described in the following table:

Parameter Value	Description
NO	"NO" is the default value of the REPLICATION parameter. This value indicates that the Replication report, the Replication Summary report, and the additional replication statistics should not be produced.
YES	Specify "YES" to produce the basic Replication report and the Replication Summary report in the ADAWRK run. This report will print overall information about each transaction found on Work part 1 that was replicated but for which Adabas has not yet received confirmation from the Event Replicator Server. The information printed includes the transaction start and end times as well as the numbers and types of modifications performed.
DETAIL	Specify "DETAIL" to produce a more detailed version of the basic Replication and Replication Summary reports. The Replication report contains all of the information produced if you specify REPLICATION=YES, but also provides more detailed file information such as the ISNs in each file modified by the transaction and the type of modifications performed against each file.
FULL	Specify "FULL" to produce the most detailed version of the Replication report available to you and the Replication Summary report. The Replication report contains all of the information produced if you specify REPLICATION=DETAIL, but also includes the data storage and inverted list data associated with each modification in a transaction.

For complete information about the Replication and Replication Summary reports, read *Replication-Related Reporting*, elsewhere in this chapter.

REPORTFILE

Use the REPORTFILE parameter to indicate whether or not the File Statistics report should be printed. Valid values are "YES" and "NO". "NO" indicates that the File Statistics report should *not* be printed; "YES" indicates that the File Statistics report *should* be printed.

The default is "NO".

When the CMID, FILES, USERID, or ETID parameters are specified with the REPORTFILE parameter, only updates that satisfy all of the criteria specified by all of the parameters is included in the File Statistics report.

SECUID

Use the SECUID parameter to specify up to 24 security system user IDs (SECUIDs) in character format. Each SECUID must be one to eight bytes long. When SECUIDs are specified, only Work part 1 autorestart area records for those SECUIDs are processed by the ADAWRK utility and printed in its reports.

SUMMARY

Use the SUMMARY parameter to indicate whether or not the Summary report should be printed. Valid values are "YES" and "NO". "NO" indicates that the Summary report should *not* be printed; "YES" indicates that the Summary report *should* be printed.

The default is "YES".

When the CMID, FILES, USERID, or ETID parameters are specified with the SUMMARY parameter, only updates that satisfy all of the criteria specified by all of the parameters is included in the Summary report.

TEST

Use the TEST parameter to test the validity of the parameters you have specified for the ADAWRK utility. When you specify the TEST parameter, no reports are produced.

TIMEZONE

Use the TIMEZONE parameter to convert the time values to a specific time zone prior to producing the ADAWRK reports. Valid values for the TIMEZONE parameter are LOCAL, MACHINE, and an hour difference specification in the format +|- *nn* (where *nn* is a valid value from 0 to 23).

Parameter Value	Description
LOCAL	Times are adjusted by the local adjustment value found on the machine. This is the default.
MACHINE	Times are printed as they are found.
-23 to +23	Times are adjusted backward (if a minus sign is specified) or forward (if a plus sign is specified) by the number of hours specified.

TRANSACTIONS

Use the TRANSACTIONS parameter to indicate whether or not the Transaction report should be printed. Valid values are "YES", "NO", "DETAIL", and "FULL":

Parameter Value	Description
NO	The Transaction report is *not* printed. This is the default.
YES	The basic Transaction report is printed. Overall details about each transaction found on Work-part-1 that would impact the recovery process are printed. This includes ended (ET)transactions whose updates may not have been applied to the database or transactions that were not ended at the time the ADAWRK utility was run against the database. The information printed in this case will include transaction start and end time, the numbers and types of modifications and other similar information.
DETAIL	The basic Transaction report (the same as when TRANSACTIONS=YES is specified) is printed along with additional details about the files that were modified by the transaction, the ISNs modified on each file, and the modification type.
FULL	The TRANSACTION=DETAIL report is printed along with additional information about the data related to each file modification. All data storage and inverted list (descriptor value table DVT) data associated with each modification in a transaction is printed.

When the CMID, FILES, USERID, or ETID parameters are specified with the TRANSACTIONS parameter, only updates that satisfy all of the criteria specified by all of the parameters is included in the Transaction report.

USERID

Use the USERID parameter to specify up to 24 user ID values that should be used to filter the Work part 1 autorestart area records processed by the utility. User IDs are stored in the last eight bytes of the communication ID in a record. Only records with communication IDs whose last eight bytes match the user IDs listed in the USERID parameter will be processed by the ADAWRK utility and printed on its reports.

User IDs must be specified as one to eight bytes long and in character or hexadecimal format. If you specify user IDs in character format, you may use an asterisk (*) as a wildcard character; user IDs specified in hexadecimal cannot use wildcards (the specification of an asterisk in a hexadecimal user ID will be converted to X'5C' in ADAWRK processing).

The asterisk wildcard character in character user IDs must be specified at the end of a user ID. For example, USERID=ABC* would cause the ADAWRK utility to process all Work part 1 autorestart area records with user ID values beginning with the letters "ABC". However, if USERID=*ABC were specified, the asterisk wildcard would be ignored and the ADAWRK utility would process all Work part 1 autorestart are records with a user ID value of "ABC".

187 Report Descriptions

- Environment Report 1362
- Summary Report 1363
- File Report 1366
- Transaction Report 1370
- Checkpoint Record Reporting 1377
- Replication-Related Reporting 1378

This chapter describes the possible reports produced by the ADAWRK utility.

All reports are affected by the settings of the filter parameters CMID, ETID, FILES, and USERID. When these parameters are specified in an ADAWRK run, ADAWRK will only process records and produce reports from Work part 1 autorestart area records that meet the criteria specified by the filter parameters.

Environment Report

Here is a sample of the Environment report. This page is always printed -- regardless of the ADAWRK parameters specified. This report identifies the ADAWRK parameters used to produce the report as well as the Work data sets used for the report.

```
A D A W R K   V8.2  SM8   DBID = 15650  Started           2009-07-30  14:39:46



Parameters:
-----------


ADAWRK NOPPT
ADAWRK FORCE=NO
ADAWRK TRANSACTIONS=FULL
ADAWRK REPORTFILE=YES
ADAWRK REPLICATION=FULL




Work datasets provided in JCL.
Database ID on Work is 15650.

The following Work datasets were used:
    -    ADABAS.ADAWRK1.WORKR1                                                ↩
```

The Environment report is the first page of the ADAWRK report. Usually, this report looks the same, whether or not replication-related records were processed by ADAWRK. However, if the replication restart point (RRP) could not be determined on the Work data sets (when the RRP has been overwritten), the following message appears in the Environment report:

```
The Replication Restart Point was not found on the Work
data set(s) provided. Replication information may have been lost.
```

The RRP can be overwritten on the Work data set when Adabas has not received replication confirmations from the Event Replicator Server for a long time. In this case, Adabas continues to write new protection records to the Work data sets and eventually it may overwrite records that are needed for resending replication data. If the replication data associated with the overwritten protection records has not yet been processed by the Event Replicator Server and if the data is no longer available on the Event Replicator Server, it is lost and the source and target replication data are out of sync. The Adabas files that may be affected by this loss of replication data are identified in the new “REPL Data” column of the Summary report.

Summary Report

The Summary report is triggered by the ADAWRK SUMMARY parameter. When SUMMARY=YES (the default) is specified, a Summary report is produced that provides an overview of the data in the autorestart area of Work part 1. This report consists of a number of sections, as follows:

- An overview of what was processed and any environmental information that can be gleaned from the provided input
- An overview of the files that will be modified as part of an autorestart and some details about what parts of the file may be affected
- The total number of system checkpoint records in the autorestart area of Work part 1 (checkpoints with a name starting with the letters "SYN")
- The total number of user checkpoint records in the autorestart area of Work part 1
- The total number of ET transactions that may need to be redone (including a subtotal of those that were backed out)
- For ET transactions that may need to be redone, the number of insert, update, and delete operations that were performed on the file
- For ET transactions that may need to be redone, the total number of data modifications to the file (total of inserts, updates, and deletes), the total number of data record updates, the total number of update commands with associated descriptor value table (inverted list) records, and the total number of descriptor value table (inverted list) updates
- The total number of incomplete transactions for the file
- For incomplete transactions, the number of insert, update, and delete operations that were performed on the file
- For incomplete transactions, the total number of data modifications to the file (total of inserts, updates, and deletes), the total number of data record updates, the total number of update commands with associated descriptor value table (inverted list) records, and the total number of descriptor value table (inverted list) updates

The following is a sample of a Summary report produced by ADAWRK:

```
*******************************************************************************
*                          A D A W R K  Summary Report                         *
*******************************************************************************


Work Dataset: ADABAS.ADAWRK1.WORKR1                            NUCID: 12021

I Item                      I   Work RABN I      Blocks I Date/Time           I
I--------------------------I-------------I-------------I---------------------I
I LP value                  I             I       2,000 I                     I
I Last written block        I           3 I             I 2016-12-06 14:06:25 I
I First block of interest   I           3 I           1 I 2016-12-06 13:32:17 I
I Autobackout endpoint      I           3 I           1 I 2016-12-06 13:32:17 I
I Backward repair endpoint  I           0 I           4 I                     I
I--------------------------I-------------I-------------I---------------------I


WORK Part 1 extension (Part 1b):

I Item                      I   Work RABN I      Blocks I Date/Time           I
I--------------------------I-------------I-------------I---------------------I
I Part 1b allocated/used    I             I    10/    0 I                     I
I Latest buffer flush       I             I             I 2016-12-06 14:06:25 I
I Latest block written      I           0 I             I 2016-12-06 13:21:16 I
I Latest session start      I             I             I 2016-12-06 13:21:16 I
I--------------------------I-------------I-------------I---------------------I
Sign of life timestamp was used
No WORK part 1 extension blocks were used

Replication Restart Information:

I Item                      I   Work RABN I      Blocks I Date/Time           I
I--------------------------I-------------I-------------I---------------------I
I Replication restart pointI           3 I           1 I 2016-12-06 13:32:18 I
I--------------------------I-------------I-------------I---------------------I


Records from all WORKs complete until 2016-12-06 14:06:25



Summary of potential inconsistencies:
-------------------------------------
"Index Structure"....: Index structures may be inconsistent.
"Index Values".......: Index values may be incorrect.
"Address Converter"..: Address converter may be incorrect.
"Data Contents"......: Data contents may be incorrect according to application.
"REPL Data"..........: Replication data may have been lost for the file.
```

```
I  File  I   Index   I  Index    I  Address  I   Data    I   REPL    I
I Number I Structure I  Values   I Converter I Contents  I   Data    I
I--------I-----------I-----------I-----------I-----------I-----------I
I      4 I           I           I           I           I           I
I      5 I           I           I           I           I           I
I     13 I           I           I           I           I           I
I     19 I           I           I           I     *     I           I
I    414 I           I           I           I           I           I
I--------I-----------I-----------I-----------I-----------I-----------I


The following checkpoints were encountered in the area of interest:

Total number of system checkpoints:      4


Completed transactions that will be re-done during autorestart processing:
        Total transactions.........................:       1
              Backed out...........................:       0
        Total modification commands................:       1
              Inserts (N1).........................:       1
              Updates (A1).........................:       0
              Deletes (E1).........................:       0
        Total data records modified................:       1
        Total modifications with descriptor updates..:       0
        Total descriptor updates...................:       0

Incomplete transactions that must be backed out during autorestart processing:
        Total transactions.........................:       0
        Total modification commands................:       0
              Inserts (N1).........................:       0
              Updates (A1).........................:       0
              Deletes (E1).........................:       0
        Total data records modified................:       0
        Total modifications with descriptor updates..:       0
        Total descriptor updates...................:       0
```

Notes:

1. The NUCID and the sections `Work Part 1 extension` and `Records from all WORKs complete until` are printed only for Work data sets used by cluster nuclei.
2. If no replication data is found, but you specified REPLICATION as YES, DETAIL, or FULL, the replication restart point (RRP) Work RABN, blocks, and date and time are blank. Finally, if the RRP cannot be determined, but you specified REPLICATION as YES, DETAIL, or FULL, the RRP Work RABN, blocks are listed as asterisks.

File Report

The File report is triggered by the ADAWRK REPORTFILE parameter. When REPORTFILE=YES is specified, a File report is produced that breaks down the data in the autorestart area of Work part 1 by file. For each file, the following information can be provided in a File report:

- Information similar to that provided in the Summary report
- The total number of ET transactions that may need to be redone (including a subtotal of those that were backed out)
- For ET transactions that may need to be redone, the number of insert, update, and delete operations that were performed on the file
- For ET transactions that may need to be redone, the total number of data modifications to the file (total of inserts, updates, and deletes), the total number of data record updates, the total number of update commands with associated descriptor value table (inverted list) records, and the total number of descriptor value table (inverted list) updates
- The total number of incomplete transactions for the file
- For incomplete transactions, the number of insert, update, and delete operations that were performed on the file
- For incomplete transactions, the total number of data modifications to the file (total of inserts, updates, and deletes), the total number of data record updates, the total number of update commands with associated descriptor value table (inverted list) records, and the total number of descriptor value table (inverted list) updates

The following is a sample of a File report:

```
*******************************************************************************
*                          A D A W R K  File Report                           *
*******************************************************************************


I-------------------------------------------------------------------------------I
I                                File      4                                    I
I-------------------------------------------------------------------------------I

Oldest activity in Work RABN           0 at
Latest activity in Work RABN           0 at


I  File  I   Index   I  Index    I  Address  I   Data    I   REPL    I
I Number I Structure I  Values   I Converter I Contents  I   Data    I
I--------I-----------I-----------I-----------I-----------I-----------I
I      4 I           I           I           I           I           I
I--------I-----------I-----------I-----------I-----------I-----------I
```

```
Completed transactions that will be re-done during autorestart processing:
        Total transactions.........................:       0
              Backed out...........................:       0
        Total modification commands................:       0
              Inserts (N1).........................:       0
              Updates (A1).........................:       0
              Deletes (E1).........................:       0
        Total data records modified................:       0
        Total modifications with descriptor updates..:       0
        Total descriptor updates...................:       0

Incomplete transactions that must be backed out during autorestart processing:
        Total transactions.........................:       0
        Total modification commands................:       0
              Inserts (N1).........................:       0
              Updates (A1).........................:       0
              Deletes (E1).........................:       0
        Total data records modified................:       0
        Total modifications with descriptor updates..:       0
        Total descriptor updates...................:       0


I-------------------------------------------------------------------------------I
I                                File      5                                    I
I-------------------------------------------------------------------------------I

Oldest activity in Work RABN           0 at
Latest activity in Work RABN           0 at


I  File  I   Index   I  Index    I  Address  I   Data    I   REPL    I
I Number I Structure I  Values   I Converter I Contents  I   Data    I
I--------I-----------I-----------I-----------I-----------I-----------I
I      5 I           I           I           I           I           I
I--------I-----------I-----------I-----------I-----------I-----------I


Completed transactions that will be re-done during autorestart processing:
        Total transactions.........................:       0
              Backed out...........................:       0
        Total modification commands................:       0
              Inserts (N1).........................:       0
              Updates (A1).........................:       0
              Deletes (E1).........................:       0
        Total data records modified................:       0
        Total modifications with descriptor updates..:       0
        Total descriptor updates...................:       0

Incomplete transactions that must be backed out during autorestart processing:
        Total transactions.........................:       0
        Total modification commands................:       0
```

```
              Inserts (N1).....................:       0
              Updates (A1).....................:       0
              Deletes (E1).....................:       0
        Total data records modified..................:       0
        Total modifications with descriptor updates..:       0
        Total descriptor updates.....................:       0


I-------------------------------------------------------------------------------I
I                                File     13                                    I
I-------------------------------------------------------------------------------I

Oldest activity in Work RABN           0 at
Latest activity in Work RABN           0 at


I  File  I   Index   I  Index    I  Address  I   Data    I   REPL    I
I Number I Structure I  Values   I Converter I Contents  I   Data    I
I--------I-----------I-----------I-----------I-----------I-----------I
I     13 I           I           I           I           I           I
I--------I-----------I-----------I-----------I-----------I-----------I


Completed transactions that will be re-done during autorestart processing:
        Total transactions...........................:       0
              Backed out.....................:       0
        Total modification commands..................:       0
              Inserts (N1).....................:       0
              Updates (A1).....................:       0
              Deletes (E1).....................:       0
        Total data records modified..................:       0
        Total modifications with descriptor updates..:       0
        Total descriptor updates.....................:       0

Incomplete transactions that must be backed out during autorestart processing:
        Total transactions...........................:       0
        Total modification commands..................:       0
              Inserts (N1).....................:       0
              Updates (A1).....................:       0
              Deletes (E1).....................:       0
        Total data records modified..................:       0
        Total modifications with descriptor updates..:       0
        Total descriptor updates.....................:       0


I-------------------------------------------------------------------------------I
I                                File     19                                    I
I-------------------------------------------------------------------------------I

Oldest activity in Work RABN           3 at 2009-07-30 14:39:37
Latest activity in Work RABN           3 at 2009-07-30 14:39:37
```

```
I  File  I   Index   I  Index    I  Address  I   Data    I   REPL    I
I Number I Structure I  Values   I Converter I Contents  I   Data    I
I--------I-----------I-----------I-----------I-----------I-----------I
I     19 I           I           I           I     *     I           I
I--------I-----------I-----------I-----------I-----------I-----------I


Completed transactions that will be re-done during autorestart processing:
        Total transactions.........................:       1
              Backed out...........................:       0
        Total modification commands................:       1
              Inserts (N1).........................:       1
              Updates (A1).........................:       0
              Deletes (E1).........................:       0
        Total data records modified................:       1
        Total modifications with descriptor updates..:       0
        Total descriptor updates...................:       0

Incomplete transactions that must be backed out during autorestart processing:
        Total transactions.........................:       0
        Total modification commands................:       0
              Inserts (N1).........................:       0
              Updates (A1).........................:       0
              Deletes (E1).........................:       0
        Total data records modified................:       0
        Total modifications with descriptor updates..:       0
        Total descriptor updates...................:       0


I-------------------------------------------------------------------------------I
I                               File    414                                     I
I-------------------------------------------------------------------------------I

Oldest activity in Work RABN           0 at
Latest activity in Work RABN           0 at


I  File  I   Index   I  Index    I  Address  I   Data    I   REPL    I
I Number I Structure I  Values   I Converter I Contents  I   Data    I
I--------I-----------I-----------I-----------I-----------I-----------I
I    414 I           I           I           I           I           I
I--------I-----------I-----------I-----------I-----------I-----------I


Completed transactions that will be re-done during autorestart processing:
        Total transactions.........................:       0
              Backed out...........................:       0
        Total modification commands................:       0
              Inserts (N1).........................:       0
              Updates (A1).........................:       0
              Deletes (E1).........................:       0
```

```
        Total data records modified..................:       0
        Total modifications with descriptor updates..:       0
        Total descriptor updates.....................:       0

Incomplete transactions that must be backed out during autorestart processing:
        Total transactions...........................:       0
        Total modification commands..................:       0
              Inserts (N1)......................:       0
              Updates (A1)......................:       0
              Deletes (E1)......................:       0
        Total data records modified..................:       0
        Total modifications with descriptor updates..:       0
        Total descriptor updates.....................:       0                ↩
```

Transaction Report

The Transaction report is triggered by the ADAWRK TRANSACTIONS parameter. When TRANSACTIONS=YES, TRANSACTIONS=DETAIL, or TRANSACTIONS=FULL are specified, a Transaction report is produced that breaks down the data in the autorestart area of Work part 1 by transaction. For each transaction, the following information can be provided in a Transaction report:

- The communication ID for the transaction
- Whether the transaction was still open, committed, or backed out
- The number of inserts, updates, and deletes performed by the transaction
- The total number of modifications performed by the transaction (total of inserts, updates, and deletes)
- The total number of update commands with associated descriptor value table (inverted list) records
- Whether the ET data is associated with the transaction and any ETID is associated with the transaction
- If TRANSACTIONS=DETAIL is specified, each file updated by the transaction and the ISN on each file that was updated by the transaction
- If TRANSACTIONS=FULL is specified, the Data Storage image(s) associated with each update for the transaction and the descriptor value table (inverted list) data associated with each update for the transaction
- If a transaction was replicated, a message is printed as part of the report indicating this event.

The following is a sample of a Transaction report:

```
*******************************************************************************
*                          A D A W R K  Transaction Report                    *
*******************************************************************************


I-----------------------------------------------------------------------------I
I                            New Communication ID (seq nr 14)                 I
I-----------------------------------------------------------------------------I

Communication ID:  '    hhhhD  ..?..    BBBB    '
                  x'7777777788888888C48FBE18094A360700000000C2C2C2C200000000'

SECUID: 'USER1   '                                                             ↩

This transaction was committed (ET).
ETID: None
ET data was not provided.
First transaction data in Work RABN           3 written at 2009-07-30 14:39:37
Last  transaction data in Work RABN           3 written at 2009-07-30 14:39:37

        Total modification commands..................:       1
              Inserts (N1).....................:       1
              Updates (A1).....................:       0
              Deletes (E1).....................:       0
        Total data records modified..................:       1
        Total modifications with descriptor updates..:       0
        Total descriptor updates.....................:       0

--- File    19  ISN         1,967 (internal) Inserted

#1: After Image Data Storage in Work RABN           3 written <14:39:37
0000 002C0000 07AFC802 CA05C48F D8EA05C2  ' .  . H. .D Q .B'
0010 E4C6C6C1 02050207 09C4E4C1 D3404000  'UFFA.....DUAL   '
0020 00024009 E2E2E6C1 E3C1F2F1           ' . .SSWATA21'

Transaction was committed (ET).


Summary of communication IDs with recovery data:
------------------------------------------------

Communication ID:  '    hhhhD  ..?..    BBBB    '
                  x'7777777788888888C48FBE18094A360700000000C2C2C2C200000000'   ↩
```

The following sample of a Transaction Report shows how the report appears for an EXU user (U002 in the report).

```
*****************************************************************************
*                          A D A W R K  Transaction Report                  *
*****************************************************************************


I---------------------------------------------------------------------------I
I                            New Communication ID (seq nr nn1)               I
I---------------------------------------------------------------------------I

Communication ID:  '????hhhhA.Qb2.??    U001    '
                  x'7777777788888888C11FD882F238574600000000E4F0F0F100000000'

This transaction was committed (ET).
ETID: None
ET data was not provided.
First protection data in Work RABN           2 written at 2007-08-30 12:32:13
Last  protection data in Work RABN           2 written at 2007-08-30 12:32:13

        Total modification commands..................:       1
              Inserts (N1)......................:       1
              Updates (A1)......................:       0
              Deletes (E1)......................:       0
        Total data records modified..................:       1
        Total modifications with descriptor updates..:       0
        Total descriptor updates.....................:       0

--- File    19  ISN         1,067 (internal) Inserted

#1: After Image Data Storage in Work RABN           2 written <12:32:13
0000 00280000 042BC802 C905C11F A2DE05C2  ' .  ..H.I.A.s?.B'
0010 E4C6C6C3 09404040 40404000 00024009  'UFFC.        . .'
0020 E4E2C1D1 C8E3E7C7                    'USAJHTXG'

Transaction was committed (ET).


I---------------------------------------------------------------------------I
I                            New Communication ID (seq nr nn2)               I
I---------------------------------------------------------------------------I

Communication ID:  ' .~..o           9??U001    '
                  x'0005A10E2096000040404040404040000F9B880E4F0F0F100000000'

SECUID: 'USER1   '                                                          ↩

This transaction was committed (ET).
ETID: 'U001    '
ET data was not provided.
First protection data in Work RABN           2 written at 2007-08-30 12:32:13
Last  protection data in Work RABN           2 written at 2007-08-30 12:32:13

        Total modification commands..................:      10
```

```
              Inserts (N1).....................:      10
              Updates (A1).....................:       0
              Deletes (E1).....................:       0
        Total data records modified..................:      10
        Total modifications with descriptor updates..:      10
        Total descriptor updates.....................:      10

--- File     4  ISN            11 (internal) Inserted

#2: After Image Data Storage in Work RABN           2 written <12:32:13
0000 000F0000 000B09E4 D3F0F100 000000    ' .   ..UL01    '

#3: Descriptor value updates in Work RABN           2 written <12:32:13
  Value for descriptor A8 inserted:
0000 E4D3F0F1 00000000                    'UL01    '

--- File     4  ISN            12 (internal) Inserted

#4: After Image Data Storage in Work RABN           2 written <12:32:13
0000 000F0000 000C09E4 D3F0F100 000001    ' .   ..UL01   .'

#5: Descriptor value updates in Work RABN           2 written <12:32:13
  Value for descriptor A8 inserted:
0000 E4D3F0F1 00000001                    'UL01   .'

--- File     4  ISN            13 (internal) Inserted

#6: After Image Data Storage in Work RABN           2 written <12:32:13
0000 000F0000 000D09E4 D3F0F100 000002    ' .   ..UL01   .'

#7: Descriptor value updates in Work RABN           2 written <12:32:13
  Value for descriptor A8 inserted:
0000 E4D3F0F1 00000002                    'UL01   .'

--- File     4  ISN            14 (internal) Inserted

#8: After Image Data Storage in Work RABN           2 written <12:32:13
0000 000F0000 000E09E4 D3F0F100 000003    ' .   ..UL01   .'

#9: Descriptor value updates in Work RABN           2 written <12:32:13
  Value for descriptor A8 inserted:
0000 E4D3F0F1 00000003                    'UL01   .'

--- File     4  ISN            15 (internal) Inserted

#10: After Image Data Storage in Work RABN          2 written <12:32:13
0000 000F0000 000F09E4 D3F0F100 000004    ' .   ..UL01   .'

#11: Descriptor value updates in Work RABN          2 written <12:32:13
  Value for descriptor A8 inserted:
0000 E4D3F0F1 00000004                    'UL01   .'
```

```
--- File     4  ISN            16 (internal) Inserted

#12: After Image Data Storage in Work RABN           2 written <12:32:13
0000 000F0000 001009E4 D3F0F100 000005    ' .   ..UL01   .'

#13: Descriptor value updates in Work RABN           2 written <12:32:13
  Value for descriptor A8 inserted:
0000 E4D3F0F1 00000005                   'UL01   .'

--- File     4  ISN            17 (internal) Inserted

#14: After Image Data Storage in Work RABN           2 written <12:32:13
0000 000F0000 001109E4 D3F0F100 000006    ' .   ..UL01   .'

#15: Descriptor value updates in Work RABN           2 written <12:32:13
  Value for descriptor A8 inserted:
0000 E4D3F0F1 00000006                   'UL01   .'

--- File     4  ISN            18 (internal) Inserted

#16: After Image Data Storage in Work RABN           2 written <12:32:13
0000 000F0000 001209E4 D3F0F100 000007    ' .   ..UL01   .'

#17: Descriptor value updates in Work RABN           2 written <12:32:13
  Value for descriptor A8 inserted:
0000 E4D3F0F1 00000007                   'UL01   .'

--- File     4  ISN            19 (internal) Inserted

#18: After Image Data Storage in Work RABN           2 written <12:32:13
0000 000F0000 001309E4 D3F0F100 000008    ' .   ..UL01   .'

#19: Descriptor value updates in Work RABN           2 written <12:32:13
  Value for descriptor A8 inserted:
0000 E4D3F0F1 00000008                   'UL01   .'

--- File     4  ISN            20 (internal) Inserted

#20: After Image Data Storage in Work RABN           2 written <12:32:13
0000 000F0000 001409E4 D3F0F100 000009    ' .   ..UL01   .'

#21: Descriptor value updates in Work RABN           2 written <12:32:13
  Value for descriptor A8 inserted:
0000 E4D3F0F1 00000009                   'UL01   .'

Transaction was committed (ET).


I-------------------------------------------------------------------------------I
I                           New Communication ID (seq nr nn3)                   I
I-------------------------------------------------------------------------------I
```

```
Communication ID:  ' .~..o          9??U002    '
                 x'0005A10E2096000040404040404040000F9B880E4F0F0F200000000'

These updates were performed by an EXU user.
First protection data in Work RABN          1 written at 2007-08-30 12:32:08
Last  protection data in Work RABN          1 written at 2007-08-30 12:32:08

        Total modification commands.................:      10
              Inserts (N1)..................:      10
              Updates (A1)..................:       0
              Deletes (E1)..................:       0
        Total data records modified.................:      10
        Total modifications with descriptor updates..:      10
        Total descriptor updates....................:      10

--- File     5  ISN            1 (internal) Inserted

#22: After Image Data Storage in Work RABN          1 written <12:32:08
0000 000F0000 000109E4 F0F0F200 000000    ' .   ..U002    '

#23: Descriptor value updates in Work RABN          1 written <12:32:08
  Value for descriptor A8 inserted:
0000 E4F0F0F2 00000000                   'U002    '

--- File     5  ISN            2 (internal) Inserted

#24: After Image Data Storage in Work RABN          1 written <12:32:08
0000 000F0000 000209E4 F0F0F200 000001    ' .   ..U002   .'

#25: Descriptor value updates in Work RABN          1 written <12:32:08
  Value for descriptor A8 inserted:
0000 E4F0F0F2 00000001                   'U002   .'

--- File     5  ISN            3 (internal) Inserted

#26: After Image Data Storage in Work RABN          1 written <12:32:08
0000 000F0000 000309E4 F0F0F200 000002    ' .   ..U002   .'

#27: Descriptor value updates in Work RABN          1 written <12:32:08
  Value for descriptor A8 inserted:
0000 E4F0F0F2 00000002                   'U002   .'

--- File     5  ISN            4 (internal) Inserted

#28: After Image Data Storage in Work RABN          1 written <12:32:08
0000 000F0000 000409E4 F0F0F200 000003    ' .   ..U002   .'

#29: Descriptor value updates in Work RABN          1 written <12:32:08
  Value for descriptor A8 inserted:
0000 E4F0F0F2 00000003                   'U002   .'

--- File     5  ISN            5 (internal) Inserted
```

```
#30: After Image Data Storage in Work RABN           1 written <12:32:08
0000 000F0000 000509E4 F0F0F200 000004    ' .   ..U002   .'

#31: Descriptor value updates in Work RABN           1 written <12:32:08
  Value for descriptor A8 inserted:
0000 E4F0F0F2 00000004                   'U002   .'

--- File     5  ISN             6 (internal) Inserted

#32: After Image Data Storage in Work RABN           1 written <12:32:08
0000 000F0000 000609E4 F0F0F200 000005    ' .   ..U002   .'

#33: Descriptor value updates in Work RABN           1 written <12:32:08
  Value for descriptor A8 inserted:
0000 E4F0F0F2 00000005                   'U002   .'

--- File     5  ISN             7 (internal) Inserted

#34: After Image Data Storage in Work RABN           1 written <12:32:08
0000 000F0000 000709E4 F0F0F200 000006    ' .   ..U002   .'

#35: Descriptor value updates in Work RABN           1 written <12:32:08
  Value for descriptor A8 inserted:
0000 E4F0F0F2 00000006                   'U002   .'

--- File     5  ISN             8 (internal) Inserted

#36: After Image Data Storage in Work RABN           1 written <12:32:08
0000 000F0000 000809E4 F0F0F200 000007    ' .   ..U002   .'

#37: Descriptor value updates in Work RABN           1 written <12:32:08
  Value for descriptor A8 inserted:
0000 E4F0F0F2 00000007                   'U002   .'

--- File     5  ISN             9 (internal) Inserted

#38: After Image Data Storage in Work RABN           1 written <12:32:08
0000 000F0000 000909E4 F0F0F200 000008    ' .   ..U002   .'

#39: Descriptor value updates in Work RABN           1 written <12:32:08
  Value for descriptor A8 inserted:
0000 E4F0F0F2 00000008                   'U002   .'

--- File     5  ISN            10 (internal) Inserted

#40: After Image Data Storage in Work RABN           1 written <12:32:08
0000 000F0000 000A09E4 F0F0F200 000009    ' .   ..U002   .'

#41: Descriptor value updates in Work RABN           1 written <12:32:08
  Value for descriptor A8 inserted:
0000 E4F0F0F2 00000009                   'U002   .'
```

```
1
 Summary of communication IDs with recovery data:
 ------------------------------------------------

 Communication ID:  '????hhhhA.Qb2.??    U001    '
                   x'777777778888888C11FD882F23857460000000E4F0F0F100000000'

 Communication ID:  ' .~..o           9??U002    '
                   x'0005A10E2096000040404040404040004000F9B880E4F0F0F200000000'

 Communication ID:  ' .~..o           9??U001    '
                   x'0005A10E2096000040404040404040004000F9B880E4F0F0F100000000'
```

Checkpoint Record Reporting

Checkpoint records and associated data found within the autorestart area of Work part 1 can be printed by the ADAWRK utility and are triggered by the CHECKPOINT parameter. If CHECKPOINT=YES is specified, checkpoint records such as the following are printed in the output. If a Transaction report is also requested, these checkpoint records are interspersed within the Transaction report.

```
A D A W R K   V8.1  SM8   DBID = 15003  Started           2007-01-10  18:34:01
                                                                              ↩

Work RABN           1 written at hh:mm:ss  System Checkpoint
 written by jobname='cccccccc'
 Checkpoint Name=nnnn Type=tt xxxxxxxxx
                                                                              ↩

Work RABN           1 written at hh:mm:ss  System Checkpoint
 written by jobname='cccccccc'
 Checkpoint Name=nnnn Type=tt xxxxxxxxx
                                                                              ↩

Work RABN           1 written at hh:mm:ss  System Checkpoint
 written by jobname='cccccccc'
 Checkpoint Name=nnnn Type=tt xxxxxxxxx            ↩
```

Replication-Related Reporting

Replication-related reports are described in this section. In addition, replication-related information is provided on the Environment, Summary, and Transaction reports. For more information on the replication-related data appearing on those reports, read the descriptions of them.

- Replication Report
- Replication Summary Report

Replication Report

The Replication report is printed when the ADAWRK REPLICATION parameter is set to any value except NO.

If no replication data is available on the Work data set, the following Replication report is printed:

```
*******************************************************************************
*                           A D A W R K  Replication Report                   *
*******************************************************************************
 There were no transactions involving replicated files. ↩
```

If replication data is available on the Work data set, and replication confirmation records exist for all of them (there are none outstanding), the following Replication report is printed:

```
*******************************************************************************
*                           A D A W R K  Replication Report                   *
*******************************************************************************
 All transactions involving replicated files were confirmed by the Reptor.
```

However, if unconfirmed replicated transactions are found on the Work data set and the REPLICATION parameter is set to YES, a basic Replication report is printed. This is a sample of the basic Replication report.

```
*******************************************************************************
*                           A D A W R K Replication Report                    *
*******************************************************************************

These transactions involving replicated files were not confirmed by the Reptor:


I-----------------------------------------------------------------------------I
I                         Next transaction (seq nr 2)                         I
I-----------------------------------------------------------------------------I

Communication ID:  ' .~..q           8> XXXX    '
                  x'0004A10E2098000040404040404040000F86E80E7E7E7E700000000'
```

```
First transaction data in Work RABN           2 written at 2009-07-30 14:39:35
Last  transaction data in Work RABN           2 written at 2009-07-30 14:39:35

        Total modification commands..................:       5
              Inserts (N1).....................:       5
              Updates (A1).....................:       0
              Deletes (E1).....................:       0
        Total data records modified..................:       5
        Total modifications with descriptor updates..:       5
        Total descriptor updates.....................:       5
                                                                                ↩
                                                                                ↩
                                                                          ↩
```

Note the transaction sequence number listed in the report (in green). If a Transaction report is also requested in the ADAWRK run, information on the transaction is provided in that report as well, identified by the same transaction sequence number. You can then use the transaction sequence numbers to accurately match transactions in the Transaction report with transactions in the Replication report. If the transaction was executed by an EXU user, alternate text will appear on this report that reads "These updates by an EXU user were also printed in the Transaction Report".

If REPLICATION=DETAIL is selected, a detailed Replication report is printed. Here is a sample:

```
I--------------------------------------------------------------------------------I
I                        Next transaction (seq nr 5)                             I
I--------------------------------------------------------------------------------I

Communication ID:  ' .~..o           9t A007A007'
                 x'0004A10E2096000040404040404040000F9A380C1F0F0F7C1F0F0F7'

First transaction data in Work RABN           4 written at 2007-11-29 20:30:09
Last  transaction data in Work RABN           4 written at 2007-11-29 20:30:09

Transaction started at 2007-11-29 20:26:58
Transaction ended at 2007-11-29 20:26:59

This transaction was executed by NUCID 15102.

        Total modification commands..................:       5
              Inserts (N1).....................:       5
              Updates (A1).....................:       0
              Deletes (E1).....................:       0
        Total data records modified..................:       5
        Total modifications with descriptor updates..:       0
        Total descriptor updates.....................:       0

--- File   222  ISN             3 (internal) Inserted
--- File   222  ISN            20 (internal) Inserted
--- File   222  ISN            22 (internal) Inserted
--- File   222  ISN            24 (internal) Inserted
--- File   222  ISN            26 (internal) Inserted
```

```
Transaction was committed (ET).  ↩
```

If REPLICATION=FULL is selected, a complete Replication report is printed. Here is a sample of part of such a report:

```
*****************************************************************************
*                          A D A W R K Replication Report                   *
*****************************************************************************

These transactions involving replicated files were not confirmed by the Reptor:


I---------------------------------------------------------------------------I
I                        Next transaction (seq nr 2)                        I
I---------------------------------------------------------------------------I

Communication ID:  ' .~..q           8> XXXX    '
                 x'0004A10E2098000040404040404040000F86E80E7E7E7E700000000'

First transaction data in Work RABN           2 written at 2009-07-30 14:39:35
Last  transaction data in Work RABN           2 written at 2009-07-30 14:39:35

        Total modification commands..................:       5
              Inserts (N1).....................:       5
              Updates (A1).....................:       0
              Deletes (E1).....................:       0
        Total data records modified..................:       5
        Total modifications with descriptor updates..:       5
        Total descriptor updates.....................:       5

--- File     4  ISN             1 (internal) Inserted

#2: After Image Data Storage in Work RABN           2 written <14:39:35
0000 000F0000 000109D9 C3D9C400 000000    ' .   ..RCRD    '

#3: Descriptor value updates in Work RABN           2 written <14:39:35
  Value for descriptor A8 inserted:
0000 D9C3D9C4 00000000                    'RCRD    '

--- File     4  ISN             2 (internal) Inserted

#4: After Image Data Storage in Work RABN           2 written <14:39:35
0000 000F0000 000209D9 C3D9C400 000001    ' .   ..RCRD   .'

#5: Descriptor value updates in Work RABN           2 written <14:39:35
  Value for descriptor A8 inserted:
0000 D9C3D9C4 00000001                    'RCRD   .'

--- File     4  ISN             3 (internal) Inserted
```

```
#6: After Image Data Storage in Work RABN           2 written <14:39:35
0000 000F0000 000309D9 C3D9C400 000002    ' .   ..RCRD   .'

#7: Descriptor value updates in Work RABN           2 written <14:39:35
  Value for descriptor A8 inserted:
0000 D9C3D9C4 00000002                   'RCRD   .'

--- File     4  ISN             4 (internal) Inserted

#8: After Image Data Storage in Work RABN           2 written <14:39:35
0000 000F0000 000409D9 C3D9C400 000003    ' .   ..RCRD   .'

#9: Descriptor value updates in Work RABN           2 written <14:39:35
  Value for descriptor A8 inserted:
0000 D9C3D9C4 00000003                   'RCRD   .'

--- File     4  ISN             5 (internal) Inserted

#10: After Image Data Storage in Work RABN           2 written <14:39:35
0000 000F0000 000509D9 C3D9C400 000004    ' .   ..RCRD   .'

#11: Descriptor value updates in Work RABN           2 written <14:39:35
  Value for descriptor A8 inserted:
0000 D9C3D9C4 00000004                   'RCRD   .'

Transaction was committed (ET).


I-------------------------------------------------------------------------------I
I                        Next transaction (seq nr 5)                            I
I-------------------------------------------------------------------------------I

Communication ID:  ' .~..q           8> YYYY    '
                  x'0004A10E2098000040404040404040000F86E80E8E8E8E800000000'


These updates were performed by an EXU user.
First transaction data in Work RABN           2 written at 2009-07-30 14:39:35
Last  transaction data in Work RABN           2 written at 2009-07-30 14:39:35

        Total modification commands..................:       5
              Inserts (N1).....................:       5
              Updates (A1).....................:       0
              Deletes (E1).....................:       0
        Total data records modified..................:       5
        Total modifications with descriptor updates..:       5
        Total descriptor updates.....................:       5

--- File     5  ISN             1 (internal) Inserted

#12: After Image Data Storage in Work RABN           2 written <14:39:35
0000 000F0000 000109D9 C3D9C400 000000    ' .   ..RCRD    '
```

```
#13: Descriptor value updates in Work RABN           2 written <14:39:35
  Value for descriptor A8 inserted:
0000 D9C3D9C4 00000000                    'RCRD    '

--- File     5  ISN             2 (internal) Inserted

#14: After Image Data Storage in Work RABN           2 written <14:39:35
0000 000F0000 000209D9 C3D9C400 000001    ' .   ..RCRD   .'

#15: Descriptor value updates in Work RABN           2 written <14:39:35
  Value for descriptor A8 inserted:
0000 D9C3D9C4 00000001                    'RCRD   .'

--- File     5  ISN             3 (internal) Inserted

#16: After Image Data Storage in Work RABN           2 written <14:39:35
0000 000F0000 000309D9 C3D9C400 000002    ' .   ..RCRD   .'

#17: Descriptor value updates in Work RABN           2 written <14:39:35
  Value for descriptor A8 inserted:
0000 D9C3D9C4 00000002                    'RCRD   .'

--- File     5  ISN             4 (internal) Inserted

#18: After Image Data Storage in Work RABN           2 written <14:39:35
0000 000F0000 000409D9 C3D9C400 000003    ' .   ..RCRD   .'

#19: Descriptor value updates in Work RABN           2 written <14:39:35
  Value for descriptor A8 inserted:
0000 D9C3D9C4 00000003                    'RCRD   .'

--- File     5  ISN             5 (internal) Inserted

#20: After Image Data Storage in Work RABN           2 written <14:39:35
0000 000F0000 000509D9 C3D9C400 000004    ' .   ..RCRD   .'

#21: Descriptor value updates in Work RABN           2 written <14:39:35
  Value for descriptor A8 inserted:
0000 D9C3D9C4 00000004                    'RCRD   .'

Transaction was committed (ET).
                                                                              ↩
```

Replication Summary Report

The Replication Summary report is printed when the ADAWRK REPLICATION parameter is set to any value except NO. This report lists all of the communication IDs for which there are one or more transactions to be replicated on the Work data set, but for which Event Replicator Server replication confirmation has not yet been received. The following is a sample Replication Summary report:

```
*****************************************************************************
*                       A D A W R K Replication Summary Report              *
*****************************************************************************

Communication IDs for which no Reptor confirmation was received:
-----------------------------------------------------------------

Communication ID:  ' .~..q           8> XXXX    '
                  x'0004A10E2098000040404040404040000F86E80E7E7E7E700000000'


Communication ID:  ' .~..q           8> YYYY    '
                  x'0004A10E2098000040404040404040000F86E80E8E8E8E800000000'


Communication ID:  ' .~..q           8> ZZZZ    '
                  x'0004A10E2098000040404040404040000F86E80E9E9E9E900000000'


Communication ID:  ' .~..q           8> ZZZZ    '
                  x'0004A10E2098000040404040404040000F86E80E9E9E9E900000000'


Communication ID:  ' .~..q           8> ZZZZ    '
                  x'0004A10E2098000040404040404040000F86E80E9E9E9E900000000'


Communication ID:  ' .~..q           8> AAAA    '
                  x'0004A10E2098000040404040404040000F86E80C1C1C1C100000000'
                                                                             ↩
```

188 JCL Requirements and Examples

- Requirements to Run ADAWRK JCL 1386

This section describes the job control information required to run ADAWRK on z/OS systems and shows examples of job streams.

Requirements to Run ADAWRK JCL

Data Set	DD Name	Storage	More Information
Associator	DDASSOR*n*	disk	Not required if the NOPPT parameter is specified
Work	DDWORKR*n*	disk	Multiple work data sets must be provided for a cluster nucleus
ADARUN parameters	DDCARD	reader	*Operations*
ADAWRK parameters	DDKARTE	reader	
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADAWRK messages	DDDRUCK	printer	*Messages and Codes*

ADAWRK JCL Examples

The following example produces a Summary report. Refer to member ADAWRKJ in the JOBS data set for this example.

```
//ADAWRK    JOB
//*
//*   ADAWRK: ALL FUNCTIONS
//*
//WRK      EXEC  PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD      <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <=== WORK
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADAWRK,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADAWRK    SUMMARY=YES
/*
```

The following example will produce a Summary and Transaction report. The Transaction report will provide a full report of all of the transactions that are on the Work data sets provided. Refer to member ADAWRKT in the JOBS data set for this example.

```
//ADAWRK    JOB
//*
//*   ADAWRK: PRINT ALL RELEVANT TRANSACTIONS ON WORK
//*
//WRK      EXEC  PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD      <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <=== WORK
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADAWRK,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADAWRK    TRANSACTIONS=FULL
/*
```

The following example produces a Summary, Transaction, and File report, including a full report of all transactions that are on the Work data sets. However only Work data set records for files 1, 2, 3, 4, 5, and 10 will be processed and reported by the utility. In addition, note the absence of an Associator data set in the sample JCL -- this is due to the NOPPT parameter which specifies that no PPT should be used for the run, and therefore no Associator data set is required.

```
//ADAWRK    JOB
//*
//*   ADAWRK: PRINT ALL RELEVANT TRANSACTIONS ON WORK
//*
//WRK      EXEC  PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD      <=== ADABAS LOAD
//*
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <=== WORK
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADAWRK,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
 ADAWRK NOPPT
 ADAWRK FORCE=YES
 ADAWRK FILES=1-4,5,10
 ADAWRK TRANSACTIONS=FULL
 ADAWRK REPORTFILE=YES
/*
```

XXII ADAZAP Utility: Display or Modify Asso, Data, and Work Data Sets

This chapter covers the following topics:

Functional Overview
ADAZAP Syntax
JCL Requirements and Examples

189 Functional Overview

The ADAZAP utility can be executed only when the Adabas nucleus is inactive.

ADAZAP is used to display in hexadecimal format and optionally to change the contents of the ASSO, DATA, or WORK data sets.

Note: If you use ADAZAP to change the DBID of the database in the GCB in ASSO RABN 1 (2 bytes at offset 0), set the ADARUN DBID parameter for this ADAZAP execution to the new DBID that will be in effect after the change to be made.

Because a significant element of risk is involved in modifying physical database blocks, the use of this utility is restricted. Software GmbH will provide the master code necessary to run the ADAZAP utility only on written request to the individuals at each customer site who are authorized to receive it.

In addition, we strongly recommend that you use your external security system to protect ADAZAP just as you protect other ZAP programs.

we also recommends that a current save tape be available before running ADAZAP. If an error occurs during ADAZAP execution, it may be necessary to restore the affected file or database.

If the data is successfully altered, a SYNP 3F checkpoint record is written containing the REP and VER data to provide an audit trail of any changes that have been made.

190 ADAZAP Syntax

- Essential Parameters 1394
- Optional Parameters 1394
- Examples 1395

```
ADAZAP  MCODE = master-code
        { ASSO | DATA | WORK }
        [LENGTH = length-of-data ]
        [OFFSET = { offset-from-RABN-start | 0 } ]
        [RABN = { rabn-number | 1 } ]
        [REP = replace-data ]
        [VER = verify-data ]
```

This chapter describes the syntax and parameters of the ADAZAP utility.

Essential Parameters

MCODE

For security purposes, a mastercode is required to run the ADAZAP utility. We provide the 8-byte mastercode on written request to authorized individuals.

ASSO | DATA | WORK

It is necessary to specify the physical data set you wish to display or modify.

Optional Parameters

LENGTH

The length of the data to be displayed. LENGTH cannot be specified if VER is specified, and the reverse.

The minimum number of bytes displayed is 16 since the lower address is rounded down to a 16-byte boundary and the upper address is rounded up to a 16-byte boundary.

OFFSET

This is the offset from the start of the block. The value must be smaller than or equal to the length of a block; that is, it must fall within the block. The default value is zero.

RABN

The relative Adabas block number (RABN) that is to be displayed or altered. The default is '1'.

REP

The replace data, which must be less than or equal to the verify data specified in the VER parameter. Up to 128 bytes of hexadecimal data may be specified.

VER

The verify data, which must be at least as long as the replace data. Up to 128 bytes of hexadecimal data may be specified.

Examples

Example 1:

```
ADAZAP MCODE=master-code
       ADAZAP ASSO OFFSET=X'10',LENGTH=16
```

The default RABN=1 is used. ADAZAP displays the database name.

Example 2:

```
ADAZAP MCODE=master-code
ADAZAP WORK OFFSET=X'10'
ADAZAP      VER=X'C1C2'
ADAZAP      REP=X'C2C1'
```

The default RABN=1 is used. ADAZAP alters data in the Work data set.

191 JCL Requirements and Examples

- Requirements to Run ADAZAP JCL 1398

Below are sample jobs to use the ADAZAP utility. They can be used to change the contents of a specific Adabas RABN in DATA, ASSO, or WORK.

- Specify the RABN, the offset, and the values to be replaced in hexadecimal.
- To obtain the master password, contact your local support center.

Important: This utility must be used carefully. Any misuse may lead to serious problems.

Requirements to Run ADAZAP JCL

Data Set	DD Name	Storage	More Information
Associator	DDASSOR*n*	disk	required if ASSO is being zapped and for all other operations. Must always be specified.
Data Storage	DDDATAR*n*	disk	required if DATA is being zapped
Work	DDWORKR*n*	disk	required if WORK is being zapped
ADARUN parameters	DDCARD	reader	*Operations*
ADAZAP parameters	DDKARTE	reader	
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADAZAP messages	DDDRUCK	printer	*Messages and Codes*

ADAZAP JCL Example

```
//ADAZAP    JOB
//*
//ZAP       EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADAvrs.LOAD       <=== ADABAS LOAD
//*
//DDASSOR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDDATAR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <=== WORK
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADAZAP,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
//DDKARTE  DD   *
  ADAZAP MCODE=mmmmmmmm                              <<--- MASTER PASSWORD        ↩

  ADAZAP ASSO RABN=1,OFFSET=X'10',LENGTH=16          <=== DISPLAY ASSO RABN 1
  ADAZAP ASSO RABN=1,OFFSET=X'10',VER=X'C1C2',REP=X'C2C1'
  ADAZAP ASSO RABN=1,OFFSET=X'10',LENGTH=16          <=== DISPLAY ASSO RABN 1
/*                                                                              ↩
```

Refer to ADAZAP in the JOBS data set for this example.

XXIII ADAZIN Utility: Print Adabas Maintenance and SVC Information

This chapter covers the following topics:

Functional Overview
ADAZIN Syntax
JCL Requirements and Examples
Sample ADAZIN Report

192 Functional Overview

- Usage Notes and Processing 1404

The ADAZIN utility can be used to print maintenance information about Adabas load modules and status information about the Adabas SVC on the system in which ADAZIN is run. Names of target load modules and SVC numbers can be specified to the utility to limit the range of the printed report.

Module information in the ADAZIN report includes:

- The load module name.

  Note: Some delivered load modules such as non-executable data modules, non-assembly language programs, subroutines, and sample programs that cannot have regular maintenance applied to them are excluded from the report.

- CSECT names (if appropriate)
- The date the module was last compiled
- The Adabas version and release of the module
- The number of the library from which the module was loaded within the concatenation list
- A list of zap numbers applied to the module for the zap base level.

Usage Notes and Processing

The Adabas modules that ADAZIN reviews for the report reside in a load library (or concatenation of load libraries) defined through one of the following job control statements in the ADAZIN batch job:

- DDZIN
- STEPLIB (if no DDZIN job control statement exists in the job)

It is important that the library where ADAZIN itself resides is APF-authorized, because ADAZIN processing behaves differently if it is not:

- If the ADAZIN library *is* APF-authorized, then ADAZIN can load into memory any module, regardless of whether the module is already loaded in memory (possibly another version from a different load library). z/OS allows this only for APF-authorized programs.

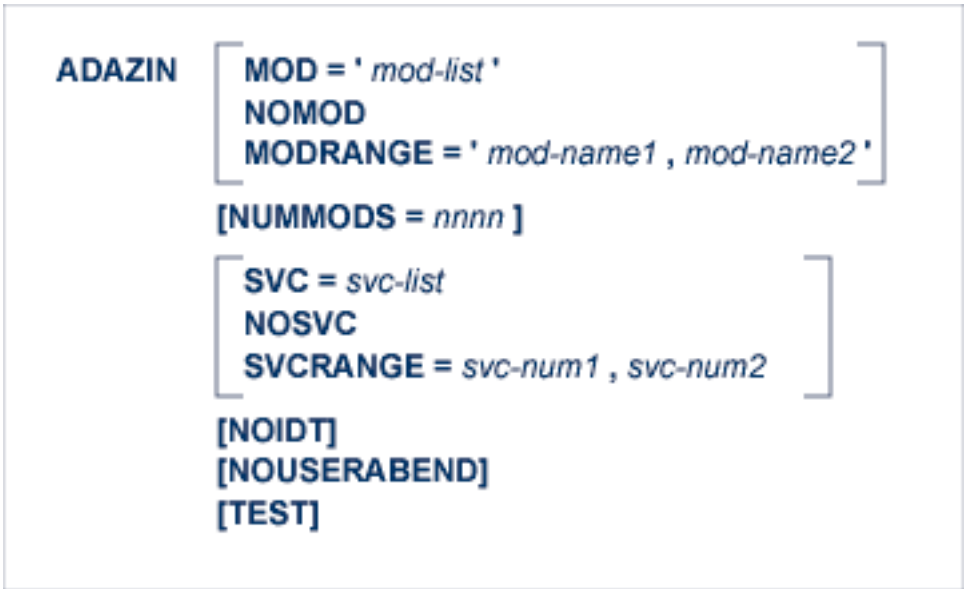
  For example, suppose that module ADAIOR is in library X referred by DDZIN and we want to check its version and zap status. In addition, module ADAIOR is in library Y referred by the STEPLIB (where ADAZIN itself also resides). If library Y is APF-authorized, then ADAZIN will load ADAIOR from library X, regardless of the fact that ADAIOR already exists in memory, loaded from library Y. ADAZIN, in this case, will report the status of module ADAIOR from library X.

- If ADAZIN library is *not* APF-authorized, then z/OS cannot load into memory modules with names that match the names of modules it has already loaded into memory. So, in the example in the previous bullet, ADAZIN can never report on the ADAIOR module from library X, because it already has ADAIOR loaded from library Y. In this case, it will always report the ADAIOR status from library Y.

In addition to the module information, ADAZIN provides similar status information for the Adabas SVCs, according to the SVC or SVCRANGE parameters.

193 ADAZIN Syntax

- Optional Parameters 1408



This chapter describes the syntax and parameters of the ADAZIN utility. All parameters are optional.

Optional Parameters

MOD: Specify Module List

Use the MOD parameter to specify a list of load modules for which maintenance information should be printed. Module names should be separated with commas. In the following example, ADAZIN would print maintenance information for the ADAREP and ADASEL load modules only.

```
ADAZIN MOD='ADAREP,ADASEL'
```

A maximum of 255 module names can be listed in the MOD parameter.

The default is to print maintenance information about all of the load modules in the load module library. In other words, the following coding would print maintenance information about all of the load modules in the load module library:

```
ADAZIN
```

The MOD, MODRANGE, and NOMOD parameters are mutually exclusive. Only one of them may be specified in an ADAZIN run, although none of them are required.

MODRANGE: Specify Module Range

Use the MODRANGE parameter to specify the names of the first and last load modules for which maintenance information should be printed. In addition to printing maintenance information about the load modules listed in the MODRANGE parameter, ADAZIN will print maintenance information for all of the load modules that fall alphabetically (by module name)

between the two specified load modules. In the following example, ADAZIN would print maintenance information for the ADAREP and ADASEL load modules as well as for every other load module in the load library with a module name that falls alphabetically between ADAREP and ADASEL (for example, ADASAV would also be included in the report).

```
ADAZIN MODRANGE='ADAREP,ADASEL'
```

The MOD, MODRANGE, and NOMOD parameters are mutually exclusive. Only one of them may be specified in an ADAZIN run, although none of them are required.

NOMOD: Specify No Modules

Specify the NOMOD parameter to indicate that maintenance information should not be printed for load modules in the ADAZIN run.

The MOD, MODRANGE, and NOMOD parameters are mutually exclusive. Only one of them may be specified in an ADAZIN run, although none of them are required.

NOSVC: Specify No SVCs

Specify the NOSVC parameter to indicate that status information should not be printed for any SVCs in the ADAZIN run.

The SVC, SVCRANGE, and NOSVC parameters are mutually exclusive. Only one of them may be specified in an ADAZIN run, although none of them are required.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note: When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

NUMMODS: Specify Number of Modules

Use the NUMMODS parameter to estimate the number of members in the target load libraries. Usually this parameter is optional. However, if the target load libraries contain more than 5000 members, the NUMMODS parameter must be specified.

ADAZIN uses the NUMMODS parameter to estimate the space it requires to build the library member list. If this number is too small, the list will be truncated and all modules may not be processed.

SVC: Specify SVC List

Use the SVC parameter to specify a list of SVCs for which status information should be printed. SVC numbers should be separated with commas. In the following example, ADAZIN would print status information for SVC 225 and 255 only.

```
ADAZIN SVC=225,255
```

SVC numbers must lie in the range 200 through 255, inclusive. A maximum of 56 SVC numbers can be listed in the SVC parameter.

The default is to print status information about all of the SVCs in use by Adabas unless parameter NOSVC is specified.

The SVC, SVCRANGE, and NOSVC parameters are mutually exclusive. Only one of them may be specified in an ADAZIN run, although none of them are required.

SVCRANGE: Specify SVC Range

Use the SVCRANGE parameter to specify the first and last SVC numbers for which status information should be printed. In addition to printing status information about the SVCs listed in the SVCRANGE parameter, ADAZIN will print status information for all of the SVCs that fall numerically between the two specified SVC numbers. In the following example, ADAZIN would print status information for SVC 225 and 255 load modules as well as for every other SVC number that falls between 225 and 255 (for example, SVC 240 would also be included in the report).

```
ADAZIN SVCRANGE=225,255
```

SVC numbers must lie in the range 200 through 255, inclusive.

The SVC, SVCRANGE, and NOSVC parameters are mutually exclusive. Only one of them may be specified in an ADAZIN run, although none of them are required.

TEST: Test Syntax

This parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

194 JCL Requirements and Examples

- Requirements to Run ADAZIN JCL 1412

This section describes the job control information required to run ADAZIN on z/OS systems and shows an example job stream.

Requirements to Run ADAZIN JCL

Data Set	DD Name	Storage	More Information
Load	DDZIN	disk	not required. If not specified, the load modules will be listed from the STEPLIB, JOBLIB, LPA, and LNKLST. If specified, one or more load libraries should be supplied.
Load	STEPLIB	disk	required to supply code for ADAZIN utility and as default if no DDZIN DD is specified.
ADARUN parameters	DDCARD	reader	*Operations*
ADAZIN parameters	DDKARTE	reader	
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADAZIN messages	DDDRUCK	printer	*Messages and Codes*

ADAZIN JCL Example

```
//ADAZIN    JOB . . .
//*
//ZIN       EXEC PGM=ADARUN,REGION=0M
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.ADA822.LOAD
//*
//DDZIN     DD   DISP=SHR,DSN=ADABAS.ACF822.LOAD
//          DD   DISP=SHR,DSN=ADABAS.ADA822.LOAD
//*
//DDDRUCK  DD SYSOUT=*
//DDPRINT  DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//DDCARD   DD *
ADARUN PROG=ADAZIN,SVC=235,DEVICE=3390,DBID=123
/*
//DDKARTE  DD   *
 ADAZIN                                  ↩
```

Refer to ADAZIN in the JOBS data set for this example.

195 Sample ADAZIN Report

The ADAZIN report lists maintenance and status information about Adabas load modules and status information about the Adabas SVC on the system in which ADAZIN is run. Module information is listed in messages with the following format:

```
module Date date, Version a.i, SM s, Level LXfff member nn
```

The italicized pieces of this format are explained in the following table:

Message Part	Description
module	The module name.
date	The date of the module in *yyyy-mm-dd* format.
a.i	The version and release of the module.
s	The system maintenance (SM) level of the module.
fff	The solution level of the module (omitted for the base SM level).
member	The data set member in which the module can be found. The data set is identified in message ZIN135 of the ADAZIN report.
nn	The data set concatenation number.

The following represents a sample of a report output by ADAZIN:

```
A D A Z I N   V8.5  SP3   DBID = 00123  Started      2021-08-16  14:11:20

Parameters:
-----------

ADAZIN SVC=254
ADAZIN MODRANGE='ADANCA,ADANC9'

ZIN135 +00 SAG.ADA853.LOAD
```

```
ADANCA             Date 2019-08-25, Version 8.5, SP 3            ADANCA   00
                   Zaps AN853010

ADANCB             Date 2019-08-25, Version 8.5, SP 3            ADANCB   00

ADANCC             Date 2019-08-25, Version 8.5, SP 3            ADANCC   00

ADANCX             Date 2019-08-25, Version 8.5, SP 3            ADANCX   00
                   Zaps AN853002

ADANCZ             Date 2019-08-25, Version 8.5, SP 3            ADANCZ   00

ADANC0             Date 2019-08-25, Version 8.5, SP 3            ADANC0   00

ADANC1             Date 2019-08-25, Version 8.5, SP 3            ADANC1   00

ADANC2             Date 2019-08-25, Version 8.5, SP 3            ADANC2   00

ADANC3             Date 2019-08-25, Version 8.5, SP 3            ADANC3   00

ADANC4             Date 2019-08-25, Version 8.5, SP 3            ADANC4   00
                   Zaps AN853011

ADANC5             Date 2019-08-25, Version 8.5, SP 3             ADANC5  00
                   Zaps AN853003

ADANC6             Date 2019-08-29, Version 8.5, SP 3             ADANC6  00

ADANC7             Date 2019-09-06, Version 8.5, SP 3             ADANC7  00
                   Zaps AN853002 AN853007 AN853008

ADANC8             Date 2019-09-10, Version 8.5, SP 3             ADANC8  00

ADANC9             Date 2019-08-25, Version 8.5, SP 3             ADANC9  00
                   Zaps AN853011

SVC 254  SVCMVS    Date 2019-08-25, Version 8.5, SP 3

         SVCCLU    Date 2019-08-25, Version 8.5, SP 3
```

XXIV

- 196 Adabas Sequential Files 1417
- 197 Adabas Personnel Demo File 1425

196 Adabas Sequential Files

- Sequential File Table 1418
- z/OS Dependencies 1421
- Enhancements to TEMP and SORT Processing in ADAINV, ADALOD and ADAULD 1422

This chapter describes Adabas sequential files.

Sequential File Table

This section summarizes the sequential files used by the Adabas utilities. Explanations of the table heading and contents are in the text following the table.

Utility	File Name	Out	In	BLKSIZE by device	Concatenation
ADACDC	DDSIIN		x		Yes
ADACHK	DDFEHL	x		Yes	
ADACMP	DDAUSBA	x	x		Yes
	DDEBAND	x			
	DDFEHL				
ADACNV	DDFILEA	x			
ADALOD	DDEBAND	x	x	Yes	Yes
	DDFILEA	x	x		Yes
	DDISN		x		
	DD/OLD				
ADAMER	DDEBAND		x		
ADAORD	DDFILEA	x	x	Yes	
ADAPLP	DD/PLOG		x		Yes
ADARAI	DDOUT	x			
ADAREP	DDSAVE		x		Yes
	DDPLOG		x		Yes
ADARES	DDBACK	x	x		Yes
	DDSIAUS1	x	x		Yes
	DDSIAUS2				
	DDSIIN				
ADASAV	DDDEL1		x		Yes
	DDDEL2		x		Yes
	DDDEL3		x		Yes
	DDDEL4		x		Yes
	DDDEL5		x		Yes

Utility	File Name	Out	In	BLKSIZE by device	Concatenation
	DDDEL6		x		Yes
	DDDEL7		x		Yes
	DDDEL8		x		Yes
	DDDUAL1	x			
	DDDUAL2	x			
	DDDUAL3	x			
	DDDUAL4	x			
	DDDUAL5	x			
	DDDUAL6	x			
	DDDUAL7	x			
	DDDUAL8	x			
	DDFULL		x		Yes
	DDPLOG		x		Yes
	DDREST1		x		Yes
	DDREST2		x		
	DDREST3		x		
	DDREST4		x		
	DDREST5		x		
	DDREST6		x		
	DDREST7		x		
	DDREST8		x		
	DDSAVE1	x			
	DDSAVE2	x			
	DDSAVE3	x			
	DDSAVE4	x			
	DDSAVE5	x			
	DDSAVE6	x			
	DDSAVE7	x			
	DDSAVE8	x			

Utility	File Name	Out	In	BLKSIZE by device	Concatenation
ADASEL	DDEXPA1	x			
	DDEXPA2	x			
	DDEXPA3	x			
	DDEXPA4	x			
	DDEXPA5	x			
	DDEXPA6	x			
	DDEXPA7	x			
	DDEXPA8	x			
	DDEXPA9	x			
	DDEXPA10	x			
	DDEXPA11	x			
	DDEXPA12	x			
	DDEXPA13	x			
	DDEXPA14	x			
	DDEXPA15	x			
	DDEXPA16	x			
	DDEXPA17	x			
	DDEXPA18	x			
	DDEXPA19	x			
	DDEXPA20	x			
	DDSIIN		x		Yes
	DDEBAND		x		No
	DDSAVE		x		No
ADAULD	DDOUT1	x	x	Yes	Yes
	DDOUT2	x	x	Yes	Yes
	DDISN	x	x	Yes	Yes
	DDSAVE		x		Yes
	DDPLOG				
	DDFULL				

Utility	File Name	Out	In	BLKSIZE by device	Concatenation
	DDDEL1-8				

Files that are both output and input are first written and then read by the indicated program.

z/OS Dependencies

To allow utilities to access data set information after closing, the DD statement for sequential data sets used in utilities should not contain FREE=CLOSE.

The BLKSIZE of a sequential file is determined as follows:

- If the column, "BLKSIZE by device" specifies "Yes" for a file, the default BLKSIZE depends on the device type as follows:

Tape:	up to 256K
3380 disk:	23476
3390 disk:	27998

- If the column "BLKSIZE by device" does *not* specify "Yes" for a file, the file's BLKSIZE is obtained from the DD statement or data set label, if present. It must be present for any input file.
- If the column "BLKSIZE by device" does *not* specify "Yes" for a file *and* the BLKSIZE cannot be obtained from the DD statement or data set label, the value of the ADARUN QBLKSIZE parameter is used, if specified.

Except for ADACMP EBAND, the RECFM and LRECL of all sequential files are VB and BLKSIZE-4, respectively. For ADACMP EBAND, RECFM and LRECL must be available from the DD statement and/or data set label.

If the DCB BUFNO parameter is provided on the DD statement, it will be used.

If the DCB BUFNO parameter is not provided on the DD statement, 40 buffers are allocated for disk and tape datasets with a block size of less than 64K, and 10 buffers are allocated for tape datasets with a block size of 64K or larger.

Enhancements to TEMP and SORT Processing in ADAINV, ADALOD and ADAULD

Various enhancements to the processing of TEMP and SORT datasets are now available in utilities ADAINV, ADALOD and ADAULD. Similar enhancements are already available for use by the ADACHK utility.

- Allow SORT and TEMP to be dynamically extended when full.
- Support writing to SORT and TEMP even though the datasets have not been pre-formatted (e.g. via ADAFRM).
- Support temporary dataset usage for TEMP and SORT, e.g.:

```
//DDTEMPR1 DD  DSN=&&TEMP,DISP=(NEW,DELETE),
//             SPACE=(CYL,(100,100)),VOL=(,,,10)
//DDSORTR1 DD  DSN=&&SORT,DISP=NEW,
//             SPACE=(CYL,(100,100)),RECFM=F,BLKSIZE=8904
```

The TEMPSIZE and SORTSIZE parameters, which were previously mandatory, are now optional parameters.

If TEMPSIZE/SORTSIZE is specified, the utility will process TEMP/SORT in the same way as in previous versions, i.e. pre-formatting of TEMP/SORT is required and dynamically extending the dataset is not available.

By omitting the TEMPSIZE/SORTSIZE parameter, users can take advantage of the enhancements listed above.

When TEMPSIZE/SORTSIZE is omitted, the user can either let the utility choose the TEMP/SORT device type (by omitting the TEMPDEV/SORTDEV parameter), or else set the TEMP/SORT device type by specifying TEMPDEV/SORTDEV.

When both TEMPSIZE/SORTSIZE and TEMPDEV/SORTDEV are omitted and TEMP/SORT is unformatted, users can specify the block size to be used for TEMP/SORT via JCL. The sample JCL for DDSORTR1 above provides an example of this. In this case, the block size in the JCL will be used if it matches an Adabas device type. A JCL block size which does not match an Adabas device type is treated as an error. For the list of Adabas device types and corresponding block sizes, see *Supported Adabas Device Types*.

For processing TEMP/SORT datasets which are not pre-formatted, we recommend omitting both TEMPSIZE/SORTSIZE and TEMPDEV/SORTDEV, and not specifying a block size via JCL. The utility will then select an optimum block size.

When both TEMPSIZE/SORTSIZE and TEMPDEV/SORTDEV are omitted and TEMP/SORT is pre-formatted, the utility retains the block size of the existing dataset.

Note that it is still possible for TEMP/SORT to fill up. Users should define TEMP and SORT datasets with secondary extents large enough to hold all required data.

In general, TEMP must be large enough to accommodate all descriptor values to be loaded into the Normal Index, and SORT must be large enough for twice the largest size of all values for any one descriptor to be sorted.

See the **ADAINV**, **ADALOD** and **ADAULD** Utilities documentation for further guidance on the size requirements for TEMP/SORT datasets.

197 Adabas Personnel Demo File

Here is the FDT of the Personnel demo file distributed with Adabas. Other demo files distributed with Adabas include an Employees demo file, a Vehicle demo file, and a Miscellaneous demo file. The Personnel demo file includes data that makes use of the expanded features provided as of Adabas 8, such as large object support. Associated with the Personnel demo file is a new LOB demo file, containing the LOB data referenced by the Personnel demo file. During installation, the LOB demo file is loaded in the same job used to load the Personnel demo file.



Note: The Personnel demo file must be installed on a UES-enabled database because it includes wide-character format (W) fields.

```
1,A0                          personnel-data
2,AA,8,A,DE,UQ                personnel-id
2,AB                          id-data
3,AC,4,F,DE                   personnel-no !UQ taken!
3,AD,8,B,NU                   id-card
3,AE,0,A,LA,NU,NV,NB          signature
1,B0                          full-name
2,BA,40,W,NU                  first-name
2,BB,40,W,NU                  middle-name
2,BC,50,W,NU,DE               name
1,CA,1,A,FI                   mar-stat
1,DA,1,A,FI                   sex
1,ES,2,B,NU                   birth
1,EA,4,P,DE,NC                birth
1,F0,PE                       full-address
2,FA,60,W,NU,MU               address-line
2,FB,40,W,DE,NU               city
2,FC,10,A,NU                  post-code
2,FD,3,A,NU                   country
2,F1                          phone-numbers
3,FE,6,A,NU                   area-code
3,FF,15,A,NU                  home-phone
3,FG,15,A,NU                  home-fax
3,FH,15,A,NU                  private-mobile
```

```
3,FI,80,A,NU,MU,DE            private-email
1,IO,PE                       business-address
2,IA,40,W,NU,MU               address-line
2,IB,40,W,DE,NU               city
2,IC,10,A,NU                  post-code
2,ID,3,A,NU                   country
2,IE,5,A,NU                   room number
2,I1                          telephone
3,IF,6,A,NU                   area-code
3,IG,15,A,NU                  business-phone
3,IH,15,A,NU                  business-fax
3,II,15,A,NU                  business-mobile
3,IJ,80,A,NU,MU,DE            business-email
1,JA,6,A,DE                   dept
1,KA,66,W,DE,NU               job-title
1,LO,PE                       income
2,LA,3,A,NU                   curr-code
2,LB,6,P,NU                   salary P9.2
2,LC,6,P,NU,MU,DE             bonus P9.2
1,MA,4,G,NU                   total income (EUR)
1,NO                          leave-date
2,NA,2,U                      leave-due
2,NB,3,U,NU                   leave-taken N2.1
1,OO,PE                       leave-booked
2,OA,8,U,NU                   leave-start
2,OB,8,U,NU                   leave-end
1,PA,3,A,DE,NU,MU             language
1,QA,7,P                      last update (*TIMX)
1,RA,0,A,LB,NU,NV,NB          picture
1,SO,PE                       documents
2,SA,80,W,NU                  document-description
2,SB,3,A,NU                   document-type
2,SC,0,A,LB,NU,NV,NB,MU       document


H1=NA(1,2),NB(1,3)
S1=JA(1,2)
S2=JA(1,6),BC(1,40)
S3=LA(1,3),LB(1,6)
```

Index

, 1362

A

ABEND34 parameter, 1356
ACODE parameter, 1026
ACTIVATE parameter, 532, 536
Adabas
  ADASAV release support, 1155
Adabas 8
  ADAORD considerations, 876
  ADASAV support, 1155
  valid index ranges for ADASEL IF statements, 1294, 1299
Adabas control block
  start logging using utility, 485
  stop logging using utility, 489
Adabas Delta Save Facility Facility
  change flags, 1061
  database status report, 1061
  display status
    using utility, 469
Adabas event log
  displaying entries, 383
Adabas file
  compressing using ADACMP COMPRESS, 203
Adabas files
  personnel file FDT, 1425
  sequential list by utility, 1417
Adabas Review
  deactivate
    using utility, 491
  local mode
    switch to using utility, 491
  setting or modifying the hub ID, 491
Adabas Statistics Facility
  using ADADBS REFRESHSTATS with, 516
ADACDC utility
  errors with ADADBS or ADALOD ISNREUSE, 29
  examples, 39
  extract file, 19
  functional overview, 17
  input data, 21
  JCL requirements and examples, 41
    z/OS, 42
  operating system factors, 31
  operation, 19
  parameters, 26
  phases of operation, 19
  primary output file, 20, 22
    checkpoints, 20
  running, 25
  syntax, 26
  transaction file, 23
  user exit, 33
    calls, 36
    installing, 34
    interface description, 34
    using to update or add records, 37
ADACHK utility
  ACCHECK function, 61
  ASSOPRINT function, 73
  CHECK function, 77
  checking syntax, 56
  cipher file considerations, 56
  CLOGPRINT function, 85
  DATAPRINT function, 89
  DSCHECK function, 93
  DSIMPRINT function, 101
  DSSTPRINT function, 105
  FCBPRINT function, 109
  FDTPRINT function, 113
  FSTPRINT function, 117
  GCBPRINT function, 121
  ICHECK function, 125
  JCL requirements and examples
    z/OS, 172
  NIPRINT function, 129
  PLOGPRINT function, 133
  PPTPRINT function, 137
  print function formatted output, 58
  RLOGPRINT function, 141
  sample jobs, 60
  Sort data set considerations, 56
  SORTPRINT function, 145
  SORTTYPE=EXTERNAL operating system considerations, 57
  TEMPPRINT function, 149
  UIPRINT function, 153
  VALIDATE function, 157
  WORKPRINT function, 167
ADACHK VALIDATE utility function
  output example, 164
ADACMP utility
  collation with user exits, 277
  COMPRESS examples, 214
  COMPRESS function, 180, 203
  COMPRESS function compressed data records output, 284
  COMPRESS function output, 283
  COMPRESS function rejected data records output, 284

COMPRESS function storage requirements report, 290
compression with user exit 6, 276
data compression, 198
data verification, 198
DECOMPRESS examples, 227
DECOMPRESS function, 181, 219
DECOMPRESS function output, 295
DECOMPRESS function rejected data records output, 296
DECOMPRESS function with multiclient files, 226
input data requirements, 183
input data structure, 184
JCL requirements and examples, 275
multiple-value field counts, 184
periodic group count input requirements, 186
processing, 191
restart considerations, 201
system field requirements, 189
user exit 6, 201
variable-length field size, 190
z/OS JCL requirements and examples, 277
ADACNV utility
CONVERT function, 307
functional overview, 303
JCL requirements and examples, 317
z/OS, 318
REVERT function, 311
ADADBS ADD utility function
Associator or Data Storage data set, 326
essential parameters, 327
examples, 327
optional parameters, 327
ADADBS ISNREUSE
errors with ADACDC utility, 29
ADADBS REPLICATION function
ACTIVATE parameter, 532
DEACTIVATE parameter, 532
DSBI parameter, 532
FILE parameter, 532
KEY parameter, 533
MODIFY parameter, 532
NOKEY parameter, 533
OFF parameter, 532
ON parameter, 532
TARGET parameter, 534
ADADBS REPTOR function
ACTIVATE parameter, 536
CLOSE parameter, 536
DBID parameter, 537
DEACTIVATE parameter, 536
description, 535
DESTINATION parameter, 537
examples, 538
FILE parameter, 537
IQUEUE parameter, 537
OPEN parameter, 536
SUBSCRIPTION parameter, 538
ADADBS utility, 535
ADD data set function, 325
ADDCLOG data set function, 329
ADDPLOG data set function, 333
ALLOCATE function, 337
CHANGE function, 341
checking syntax, 324
CVOLSER function, 347
DEALLOCATE function, 351
DECREASE function, 355
DELCLOG data set function, 359
DELCP function, 363
DELDE function, 367
DELETE function, 371
DELFN function, 375
DELPLOG data set function, 379
DEVENTLOG function, 383
DSREUSE function, 387
ENCODEF function, 391
EXPFILE function, 395
functional overview, 323
INCREASE function, 399
general procedure, 401
ISNREUSE function, 405
JCL requirements and examples
z/OS, 570
MODFCB function, 409
MUPEX function, 415
NEWFIELD function, 419
ONLADD function, 423
ONLINCREASE function, 429
ONLINVERT function, 435
ONLREORFASSO function, 439
ONLREORFDATA function, 443
ONLREORFILE function, 447
OPERCOM function, 451
PRIORITY function, 499
REACTLOG function, 503
RECORDSPANNING function, 505
RECOVER function, 509
REFRESH function, 511
REFRESHSTATS function, 515
RELEASE function, 519
RENAME function, 523
RENUMBER function, 527
REPLICATION function, 531
RESETDIB function, 541
RESETPPT function, 545
SPANCOUNT function, 549
TRANSACTIONS function, 553
UNCOUPLE function, 557
UNDELDE function, 561
UNDELFN function, 565
ADADEF utility
DEFINE function, 577
functional overview, 575
checkpoint file, 576
database components, 576
JCL requirements and examples
z/OS, 596
MODIFY function, 587
NEWWORK function, 591
ADADRU SPACEUSAGE utility function
essential parameters, 612
examples, 619
file group processing, 608
optional parameters, 613
overview, 607
sample report, 619
syntax, 612
ADADRU utility
checking syntax, 604

JCL requirements and examples
z/OS, 752
JCL/JCS requirements and examples, 751
overview, 601
SPACEUSAGE function, 607
spanned record handling, 602
ADAEND operator command
using utility, 454
ADAFRM utility
all functions, 761
format new RABNs, 758
formatting database components, 758
functional overview, 757
JCL requirements and examples
z/OS, 768
reset data set blocks/cylinders to zeros, 758
ADAINV utility
COUPLE function, 775
examples, 788
functional overview, 773
INVERT function, 783
JCL requirements and examples
z/OS, 790
space allocation during execution, 787
user exits
collation, 570, 790
ADALOD ISNREUSE
errors with ADACDC utility, 29
ADALOD LOAD function
RPLERRORDEACTFILE parameter, 815
RPLINITERROR parameter, 815
ADALOD utility
JCL requirements and examples
z/OS, 854
LOAD function, 799
examples, 820
input data for, 821
space allocation for file, 822
space/statistics report, 851
storage requirements and use, 845
Temp data set requirements, 847
UPDATE function, 829
Associator updating with, 841
descriptor information generation, 841
examples, 838
input requirements, 841
mass updates to expanded files, 842
space allocation, 841
updating Associator using the LOAD function, 825
user exits
collation, 854
ADAM
load files with
using utility, 804
ADAMER utility
examples, 866
JCL requirements and examples
z/OS, 870
output report, 867
syntax, 863
ADAORD utility
Adabas 8 considerations, 876
JCL requirements and examples
z/OS, 952
REORASSO function, 877
REORDATA function, 885
REORDB function, 891
REORFASSO function, 903
REORFDATA function, 911
REORFILE function, 917
RESTRUCTUREDB function, 927
RESTRUCTUREF function, 933
STORE function, 939
ADAPLP utility
examples, 964
JCL requirements and examples
z/OS, 968
PLOGPRI function, 959
SPLOGPRI function, 959
WORKPRI function, 959
ADARAI utility
CHKDB function, 977
concepts, 974
DISABLE function, 979
file-level recovery, 1002
function directory, 971
LIST function, 981
LIST function z/OS output examples, 984
PREPARE function, 991
RECOVER function, 995
RECOVER function input data sets, 1000
RECOVER function output job stream, 1000
REMOVE function, 1011
restarting RECOVER function after interruption, 1010
running RECOVER function, 1001
skeleton job control input for RECOVER function, 1006
z/OS JCL requirements and examples, 1014
z/OS skeleton job control input for RECOVER function, 1015
ADAREP utility
examples, 1033
z/OS JCL requirements and examples, 1076
ADARES utility
BACKOUT DPLOG function, 1097
BACKOUT function, 1089
BACKOUT MPLOG function, 1097
CLCOPY function, 1107
COPY function, 1111
function directory, 1082
MERGE CLOG function, 1115
PLCOPY function, 1117
REGENERATE function, 1123
REPAIR function, 1135
z/OS JCL requirements and examples, 1142
ADASAV utility
Adabas release support, 1155
RESTONL FILES function, 1163
RESTONL FMOVE function, 1175
RESTONL function, 1157
overview, 1154
RESTONL function overview, 1154
RESTONL GCB function, 1191
RESTORE FILES function, 1207
RESTORE FMOVE function, 1219
RESTORE function, 1201
RESTORE function overview, 1154
RESTORE GCB function, 1235
RESTPLOG function, 1245
RESTPLOG function overview, 1154

SAVE FILES function, 1255
SAVE function, 1249
SAVE function overview, 1155
z/OS JCL requirements and examples, 1262
ADASEL utility
date and time syntax, 1286
DISPLAY instruction syntax, 1296
DST parameter, 1282
FDTINPUT parameter, 1277
global parameters, 1280
IF statement syntax, 1288
job requirements and examples, 1319
NEWPAGE instruction syntax, 1306
obtaining MU and PE field counts, 1314
output instruction syntax, 1295
OUTPUT instruction syntax, 1300
overview, 1271
RECORDS keyword, 1285
RECSYNC keyword, 1285
recsync update command, 1315
scenarios, 1309
SELECT parameter syntax, 1283
SET DST settings, 1282
SET GLOBALS settings, 1280
SET TZ settings, 1282
setting global parameters, 1280
SKIP instruction syntax, 1307
special update command, 1315
specifying an alternate FDT, 1277
specifying fields for output file, 1313
syntax, 1275
TEST parameter, 1277
testing syntax, 1277
TZ parameter, 1282
valid index ranges for Adabas 8 IF statements, 1294, 1299
value criterion syntax, 1289
WITH clause syntax, 1287
z/OS JCL requirements and examples, 1320
ADAULD utility
input processing, 1337
LOAD FILE function DDISN parameter, 1328
output processing, 1341
UNLOAD FILE function, 1327
user exit 9 processing, 1343
z/OS JCL requirements and examples, 1346
ADAWRK utility
exclusive user (EXU) processing, 1353
overview, 1351
replication-related data processing, 1352
replication-related reports, 1378
reports, 1361
syntax, 1355
z/OS JCL requirements and examples, 1386
ADAZAP utility
z/OS JCL requirements and examples, 1398
ADAZIN utility
functional overview, 1403
job requirements, 1411
parameters, 1408
sample report, 1413
syntax, 1407
z/OS JCL requirements and examples, 1412
address converter
allocate an extent using ADADBS utility, 338
checking, 61
deallocate an extent using the ADADBS utility, 352
space allocation
using utility, 823, 841
advance-lock
obtaining for a file, 454
alert messages
PLOG and CLOG, 486
ALL option, 1296
allocating
appended ASSO or DATA space dynamically, 429
new ASSO or DATA space dynamically, 423
ALLOCATION parameter, 1126
ALOCKF operator command
using in utility, 454
alphanumeric fields
no conversion option (NV), 243
Associator
add data set to
using utility, 326
automatic increase size command, 459
automatic increase threshold setting, 460
coupling lists
creating, 779
decrease size of data set
using utility, 355
formatting using ADAFRM, 761
increase size of data set
using utility, 399
reorder
using utility, 874
reordering for a file using ADAORD REORFASSO, 903
reordering for database using ADAORD REORASSO, 877
reset blocks/cylinders to zeros
using utility, 762
resetting PPT, 545
setting maximum size, 460
space notifications, 457-458
updating
using utility, 841
updating using ADALOD LOAD, 825
Associator (ASSO)
allocating appended data sets dynamically, 429
allocating new data sets dynamically, 423
determining space defined, 476
printing, 73
ASSOSPACEWARN
operator command, 457
ATM
loading system files, 802
attached buffers
command to display usage, 475
AUDCONNECT
operator command, 458
Audit log
close/switch dual
using utility, 483
AUDITLOG
operator command, 458
AUTOBACKOUT parameter, 1126
AUTOINCASSOSIZE
operator command, 459
AUTOINCASSOTHRESHOLD
operator command, 460

AUTOINCASSOTOTAL
operator command, 460
AUTOINCDATASIZE
operator command, 461
AUTOINCDATATHRESHOLD
operator command, 462
AUTOINCDATATOTAL
operator command, 463

B

BEFORE option, 1296
blank compression
NB field option, 242
bold, 9
BOTH option, 1296
braces ({}), 11
brackets ([]), 10

C

cache structure
statistics display, 480-481
CANCEL operator command
using utility, 464
checkpoint file
define, 577
CHECKPOINT parameter, 1356
checkpoint record reporting, 1377
checkpoints
database status report, 1067
deleting using utility, 363
written by Adabas nucleus/utilities, 1069
choices in syntax, 10
cipher file considerations
ADACHK utility, 56
CLOG
alert messages, 486
reactivating command logging, 503
CLOGDEV parameter
ADADBS ADDCLOG function, 331
CLOGs (see command logs (CLOGs))
CLOSE parameter, 536
cluster environments
dynamic CLOG addition, 331
dynamic CLOG deletion, 360
dynamic PLOG addition, 335
dynamic PLOG deletion, 380
CMDCOUNT option, 1296
CMID parameter, 1356
collation descriptor
define
using ADAINV, 784
defining in ADACMP COMPRESS, 253
Command log
add data set to
using utility, 326
close/switch dual
using utility, 483
copy a dual data set to a sequential data set using ADARES CLCOPY, 1107
formatting using ADAFRM, 761
merging multiple, 1115, 1146
start logging using utility, 485
command log
stop logging using utility, 489
command log (CLOG)
printing, 85
command logging
reactivating, 503
command logs (CLOGs)
data set device, 331
data set number, 331, 360
dynamically add data sets, 329
dynamically deleting data sets, 359
command queue
command to display usage, 475
command queue element
display posted
using utility, 468
command queue elements
display
using utility, 472
commands
displaying distribution statistics, 467
log audit data, 458
compressing files, 203
contents of PPT table, 1044
CONTINUE parameter, 1092, 1100, 1126
convert database
to higher version, 307
to lower version, 311
COUNT option, 1297
CP parameter, 1028
CPEXLIST parameter, 1027
CPLIST parameter, 1027
CT
ADARUN parameter
command to override setting, 466
CT operator command
using utility, 466

D

data compression
ADACMP utility, 180, 198
fields with NC option, 248
data decompression
ADACMP utility, 181, 219
fields with NC option, 248
data definition
COLDE statement
of ADAINV INVERT, 785
COLDE statement in ADACMP COMPRESS, 253
CR (insert-only system field) field option, 236
DE - descriptor field option, 236
DT (date-time edit mask) field option, 236
FI - fixed storage field options, 238
field options, 235
FIELD statement
of ADAINV INVERT, 785
FNDEF statement in ADACMP COMPRESS, 231
HYPDE statement
of ADAINV INVERT, 785
HYPDE statement in ADACMP COMPRESS, 256
LA - long alphanumeric field options, 238
LB - large object field options, 239
MU - multiple-value field option, 240

NB - blank compression field options, 242
NC - null not counted in compression/decompression, 248
NC - null not counted SQL field options, 247
NN - not null SQL field option, 250
NV - no conversion field options, 243
PE - periodic group field option, 252
PHONDE statement
of ADAINV INVERT, 785
PHONDE statement in ADACMP COMPRESS, 259
SUBDE statement
of ADAINV INVERT, 785
SUBDE statement in ADACMP COMPRESS, 261
SUBFN statement, 263
SUPDE statement
of ADAINV INVERT, 785
SUPDE statement in ADACMP COMPRESS, 264
SUPFN statement, 273
SY (system field) field option, 244
syntax using ADACMP, 229
TZ (time zone) field option, 245
U - null value suppression field options, 243
UQ - unique descriptor field options, 246
XI - exclude PE instance from UQ field option, 246
data integrity block
display entries
using utility, 468
reset entries in
using utility, 541
data sets
intermediate coupling storage
calculate using utility, 778
Data Storage
add data set to
using utility, 326
allocate an extent using ADADBS utility, 338
automatic increase size command, 461
automatic increase threshold setting, 462
checking, 93
deallocate an extent using the ADADBS utility, 352
decrease size of data set
using utility, 355
formatting using ADAFRM, 761
increase size of data set
using utility, 399
reorder
using utility, 874
reordering for a file using the ADAORD REORFDATA, 911
reordering for database using ADAORD REORDATA, 885
repair blocks
using utility, 1135
resetting blocks/cylinders to zeros using ADAFRM, 762
reuse blocks
using utility, 387
setting maximum size, 463
space allocation
using utility, 823
space notifications, 466
data storage
controlling index crosschecking, 484
Data Storage (DATA)
allocating appended data sets dynamically, 429
allocating new data sets dynamically, 423
determining space defined, 476
printing, 89
Data Storage Space Table (DSST)
printing, 105
Data Storage space table (DSST)
checking, 93
database
automate and optimize recovery using ADARAI, 974
build job stream for restore, 995
change name assigned to
using utility, 523
check active/inactive status
using utility, 977
define, 577
delete file from
using utility, 371
loading files into using ADAORD STORE, 875
modify physical blocks using ADAZAP utility, 1391
produce status report using ADAREP REPORT, 1023
quiesce, 553
recovering using ADARAI RECOVER, 995
reordering Associator using ADAORD REORASSO, 877
reordering Data Storage and Associator using ADAORD REORDB, 891
reordering Data Storage using ADAORD REORDATA, 885
restore from offline source using ADASAV RESTORE, 1201
restore from online source using ADASAV RESTONL, 1157
restore incremental from offline source using ADASAV RESTORE GCB, 1235
restore incremental from online source using ADASAV RESTONL GCB, 1191
restore updates performed between two checkpoints using ADARES REGENERATE, 1123
save using ADASAV SAVE, 1249
unload to sequential data set using ADAORD RESTRUCTUREDB, 875
unloading to sequential data set using ADAORD RESTRUCTUREDB, 927
database files
checking, 157
database options
change
using utility, 587
DATASPACEWARN
operator command, 466
date-time edit mask (DT) field option
description, 236
DAUQ operator command
using utility, 467
DBID parameter, 537
DCMDSTAT operator command
using utility, 467
DCQ operator command
using utility, 468
DD/JCLIN
requirement for ADARAI, 1000
DDIB operator command
using utility, 468
DDSF operator command
using utility, 469
DEACTIVATE parameter, 532, 536
default parameter values, 10
Delta Save image (DSIM) data set
printing blocks, 101
descriptor
collation, 253

define
using ADAINV utility, 783
field option (DE), 236
hyperdescriptor, 256
phonetic, 259
release from descriptor status
using utility, 519
specify for file coupling
using utility, 775
subdescriptor, 261
superdescriptor, 264
unique, 246
descriptors
logically deleting using utility, 367
undeleting logically deleting using utility, 561
DESTINATION parameter, 537
determining ASSO and DATA defined space, 476
DFILES operator command
using utility, 469
DFILESTAT operator command
using utility, 470
DFILUSE operator command
using utility, 470
DHQA operator command
using utility, 470
disk volume
print Adabas extents located on
using utility, 347
DISPLAY instruction, 1296
displaying Adabas Event Log entries, 383
DLOCKF operator command
using utility, 471
DNC operator command
using utility, 472
DNH operator command
using utility, 472
DNU operator command
using utility, 472
DONLSTAT operator command
using utility, 472
DPARM operator command
using utility, 473
DPLOG parameter, 1100
DRES operator command
using utility, 475
DSBI parameter, 532
DSPACE operator command
using utility, 476
DSTAT operator command
using utility, 477
DTH operator command
using utility, 477
DUALPLD parameter, 1101
dump
terminate online status
using utility, 491
DUQ operator command
using utility, 477
DUQA operator command
using utility, 478
DUQE operator command
utility, 478
DUUQE operator command
using utility, 479
DVOLIO operator command
using utility, 479
DXCACHE command
display lock statistics, 480
DXFILE command
display file statistics, 480
DXLOCK command
display lock statistics, 481
DXSTAT command
display cache and lock statistics, 481
dynamic allocation
appended ASSO or DATA space, 429
new ASSO or DATA space, 423
dynamically adding CLOG data sets, 329
dynamically adding PLOG data sets, 333
dynamically change MXCANCEL setting, 486
dynamically change MXCANCELWARN setting, 487
dynamically change MXMSGWARN setting, 487
dynamically change MXWTOR setting, 488
dynamically deleting CLOG data sets, 359
dynamically deleting PLOG data sets, 379

E

ellipsis (...), 11
end transaction (ET) data
selecting, 1310
ENDING AT clause, 1286
ETID parameter, 1356
ETUSERDA option, 1297
Event Replicator for Adabas
performing replication cross-check function using ADADBS OPERCOM operator command, 492
RPLCHECK ADADBS OPERCOM operator command, 492
Event Replicator Server, performing replication cross-check, 492
EXCLUDE parameter, 1092, 1101, 1127
exclusive control
obtaining for a file, 454
expanded file chain
checking uniqueness of descriptor within, 246
expanded file chains
inserting or removing files, 395
expanded files
load anchor file, 805
loading, 825
mass updates to, 842
EXPFILE utility function, 395
extended I/O list
start logging using utility, 486
stop logging using utility, 489
EXU users
ADAWRK utility reporting, 1353

F

FDTINPUT parameter, 1277, 1286
FEOFAL operator command
using utility, 483
FEOFCL operator command
using utility, 483
FEOFPL operator command
using utility, 483
field definition

syntax using ADACMP, 229
Field Definition Table
database status report, 1063
Field Definition Table (FDT)
add a field to
using utility, 419
field definition table (FDT)
printing, 113
field definitions, 231, 250
field options
DT (date-time edit mask), 236
TZ (time zone), 245
fields
add
using utility, 419
change standard length of
using utility, 341
logically deleting from a file, 375
logically undeleting using utility, 565
subfield, 263
superfield, 273
file
statistics display, 480
file control block (FCB)
printing, 109
file coupling
coupling lists, 780
lists
create using utility, 779
temporary space for
calculate using utility, 778
uncouple
using utility, 557
using utility, 775
file encoding
modify, 391
file extents
allocate
using utility, 337
deallocate
using utility, 351
print on given disk volume
using utility, 347
file information, 1054
file options, 1047, 1058
FILE parameter, 532, 537, 1027, 1127
File report, 1366
file space allocations, 1050
Files
restore updates performed between two checkpoints using ADARES REGENERATE, 1123
files
activating or deactivating replication, 531
Adabas Delta Save Facility change flags, 1061
Adabas sequential list by utility, 1417
allocate space for, 822
build job stream for restore, 995
change file number of
using utility, 527
change name assigned to
using utility, 523
compressing using ADACMP COMPRESS, 203
counting spanned records, 549
decompressing multiclient files, 226
delete
using utility, 371
determining names and blocks sizes of Adabas sequential, 1418
display locked, 471
display quantity of user types, 469
display total commands processed, 470
enabling or disabling record spanning, 505
Field Definition Table report, 1063
information about, 1054
inserting or removing in an expanded file chain, 395
load
using utility, 799
load into existing database using ADAORD STORE, 939
load with ADAM option, 804
loading into existing database using ADAORD STORE, 875
locking at all security levels, 484
locking for all non-utility use, 485
locking for all users except EXU or EXF users, 485
modify parameters of
using utility, 409
obtaining an advance-lock, 454
option settings report, 1058
personnel file FDT, 1425
recovering with ADARAI RECOVER, 1002
remove advance lock
on all files, 491
on specified file, 490
reordering Associator and Data Storage using ADAORD REORFILE, 917
reordering Associator using ADAORD REORFASSO function, 903
reordering Data Storage using ADAORD REORFDATA, 911
reset to empty status
using utility, 511
restore to any RABNs from offline source using ADASAV RESTORE FMOVE, 1219
restore to any RABNs from online source using ADASAV RESTONL FMOVE, 1175
restore to original RABNs from offline source using ADASAV RESTORE FILES, 1207
restore to original RABNs from online source using ADASAV RESTONL FILES, 1163
save using ADASAV FILES, 1255
space allocation report, 1061
special descriptors report, 1066
specifying MU or PE limits, 415
stop users of specified
using utility, 492
types, 802
unloading to sequential data set using ADAORD RESTRUCTUREDB, 875
unloading to sequential data set using ADAORD RESTRUCTUREF, 933
unlock specified
using utility, 497
unlocking for utility use, 497
FILES parameter, 1092, 1101, 1356
fixed storage (FI)
use in ADACMP, 238
FORCE parameter, 1356
format buffer
start logging using utility, 485

stop logging using utility, 489
format pool
command to display usage, 475
free space table (FST)
printing, 117
FROM FILE clause, 1285
FROM USER clause, 1285
FROMBLK parameter, 1092, 1101, 1127
FROMCP parameter, 1093, 1101, 1127
FROMDATE parameter, 1027
FROMPLOG parameter, 1091, 1125
FROMSESSION parameter, 1027

G

general control blocks (GCB)
printing, 121
general database information, 1040
general file status, 1046
generation, 975
Global lock area
statistics display, 481
global parameters
setting for in ADASEL runs, 1280
group definitions, 231

H

HALT operator command
using utility, 484
HEX keyword, 1297
hold queue
command to display usage, 475
display count of ISNs
using utility, 472
hold queue elements
display
using utility, 470
hyperdescriptor
define
using ADAINV, 784
defining in ADACMP COMPRESS, 256

I

I/O activity
start logging using utility, 485
stop logging using utility, 489
IF statements
valid index ranges, 1294, 1299
IGNORECOUPLE parameter, 1093, 1102, 1128
IGNOREEXP parameter, 1093, 1102, 1128
IN FILE clause, 1285
INCLUDE parameter, 1128
indentation, 11
index
space allocation
by ADAINV, 787
using utility, 822
space allocation for coupling lists, 781
index crosschecking controls, 484
index physical structure
checking, 125
INDEXCROSSCHECK operator command
using utility, 484
invert
start online process
using utility, 435
IQUEUE parameter, 537
ISN buffer
start logging using utility, 485
stop logging using utility, 489
ISN option, 1297
ISNs
file of unloaded, 1328
format for specifying, 839
set to reuse
using utility, 405
italic, 10

K

KEY parameter, 533

L

large object (LB)
field option, 239
LAYOUT parameter, 1029
LIMCOUNT parameter, 1030
lock structure
statistics display, 481
LOCKF operator command
using utility, 484
locking files
in advance, 454
LOCKU operator command
using utility, 485
LOCKX operator command
using utility, 485
LOGCB
using utility operator command, 485
LOGFB
using utility operator command, 485
LOGGING
operator command using utility, 485
LOGIB
using utility operator command, 485
LOGIO
using utility operator command, 485
LOGRB
using utility operator command, 485
LOGSB
using utility operator command, 485
LOGUX
using utility operator command, 485
LOGVB
using utility operator command, 485
LOGVOLIO
using utility operator command, 486
LOGWARN
operator command, 486
long alphanumeric (LA)
field option, 238
lowercase, 10
LWP parameter, 1356

M

minimum keywords, 10
mixed case, 10
MODIFY parameter, 532
MPLOG parameter, 1100
MTR parameter, 1093, 1102, 1128
MU fields
  ADACMP utility requirements, 184
  setting maximum for a file, 415
multiclient files
  assigning a new owner ID using ADALOD LOAD, 808
  decompressing, 226
  loading, 827
  specify length of owner ID using ADALOD LOAD, 810
  unloading using ADAULD, 1329
multiple-value fields
  field option (MU), 240
MXCANCEL operator command, 486
MXCANCELWARN operator command, 487
MXMSGWARN operator command, 487
MXWTOR operator command, 488

N

NEWPAGE instruction, 1306
NOAUTOBACKOUT parameter, 1094, 1102, 1129
NOCOUNT parameter, 1030
NOFDT parameter, 1030
NOFILE parameter, 1027
NOHEADER option, 1297
NOKEY parameter, 533
NOLGLIST parameter, 1030
NOLOGCB
  using utility operator command, 489
NOLOGFB
  using utility operator command, 489
NOLOGGING
  operator command
    using utility, 489
NOLOGIB
  using utility operator command, 489
NOLOGIO
  using utility operator command, 489
NOLOGRB
  using utility operator command, 489
NOLOGSB
  using utility operator command, 489
NOLOGUX
  using utility operator command, 489
NOLOGVB
  using utility operator command, 489
NOLOGVOLIO
  using utility operator command, 489
NOPHLIST parameter, 1030
NOPPT parameter, 1030, 1356
normal font, 10
normal index
  allocate an extent using ADADBS utility, 338
  deallocate an extent using the ADADBS utility, 352
  printing, 129
NOSTD parameter, 1030
NOTENH parameter, 764
NPCALLS parameter, 1094, 1103, 1129
NUCID parameter
  ADADBS ADDCLOG function, 331
  ADADBS ADDPLOG function, 335
  ADADBS DELCLOG function, 360
  ADADBS DELPLOG function, 380
  ADADBS DEVENTLOG function, 384
nucleus
  display current operating status
    using utility, 477
null value
  not allowed (NN) field option, 250
  not counted (NC) field option, 247
  SQL support, 246
  suppression (NU) field option, 243
null value indicator
  specified in record buffer, 249
NUMBER parameter
  ADADBS ADDCLOG function, 331
  ADADBS ADDPLOG function, 335
  ADADBS DELCLOG function, 360
  ADADBS DELPLOG function, 380

O

OFF parameter, 532
OFFSET parameter, 1031
ON parameter, 532
online invert
  start
    using utility, 435
online process
  display status of
    using utility, 472
  resume a suspended process
    using utility, 489
  stop cleanly
    using utility, 490
  suspend
    using utility, 490
online reorder
  Associator, 439
  Data Storage, 443
  file, 447
ONLRESUME
  operator command
    using utility, 489
ONLSTOP
  operator command
    using utility, 490
ONLSUSPEND
  operator command
    using utility, 490
OPEN parameter, 536
Operator commands
  DXLOCK
    display lock statistics, 481
operator commands
  ADADBS OPERCOM function, 451
  controlling ASSO space use notifications, 457
  controlling DATA space use notifications, 466
  DXCACHE
    display lock statistics, 480
  DXFILE
    display file statistics, 480

DXSTAT
display cache and lock statistics, 481
MXCANCEL, 486
MXCANCELWARN, 487
MXMSGWARN, 487
MXWTOR, 488
setting automatic increase size for ASSO, 459
setting automatic increase size for DATA, 461
setting automatic increase threshold for ASSO, 460
setting automatic increase threshold for DATA, 462
setting maximum size for ASSO, 460
setting maximum size for DATA, 463
optional syntax elements, 10
output files
obtaining MU and PE field counts, 1314
specifying fields, 1313
OUTPUT instruction, 1300
output instruction syntax, 1295
OUTPUT parameter, 1031

P

parallel participant table (PPT)
printing, 137
PARALLELREAD parameter, 1095, 1103, 1130
parameters
positional values
specifying, 13
PE fields
ADACMP utility requirements, 184, 186
PE groups
setting maximum for a file, 415
periodic group definition
FNDEF statement in ADACMP COMPRESS, 250
periodic group definitions, 250
periodic groups
data definition option (PE), 252
phonetic descriptor
define
using ADAINV, 784
defining using ADACMP COMPRESS, 259
physical database layout, 1052
PLOG
alert messages, 486
PLOGDBID parameter, 1095, 1104, 1130
PLOGDEV parameter, 1101
ADADBS ADDPLOG function, 335
PLOGNUM parameter, 1032, 1091, 1126
PLOGs (see protection logs (PLOGs))
PPT
resetting on Associator data set, 545
PRINT parameter, 1031
Protection log
close/switch dual
using utility, 483
dual backout updates between two checkpoints using ADARES BACKOUT DPLOG, 1097
dual copy to sequential data set using ADARES PLCOPY, 1117
formatting using ADAFRM, 761
merging multiple, 1144
multiple backout updates between two checkpoints using ADARES BACKOUT MPLOG, 1097
print records from multiple PLOGs using ADAPLP, 960
print records from sequential intermediate PLOGs using ADAPLP, 960
print records from sequential PLOGs using ADAPLP, 960
restore using ADASAV RESTPLOG, 1245
sequential backout updates between two checkpoints using ADARES BACKOUT, 1089
protection log
data set types, 1087
information described, 1087
sequential copy using ADARES COPY
copy using utility, 1111
use of compressed data, 1087
protection log (PLOG)
printing, 133
protection logs (PLOGs)
data set device, 335
data set number, 335, 380
dynamically add data sets, 333
dynamically deleting data sets, 379
punctuation and symbols in syntax, 11

Q

quiesce database
ADADBS function, 553

R

RAID parameter, 1130
RALOCKF operator command
using utility, 490
RALOCKFA operator command
using utility, 491
RDUMPST operator command
using utility, 491
reactivating
command logging, 503
read-only status
switch on/off
using utility, 464, 491
READONLY
operator command
using utility, 491
record buffer
null value indicator value, 249
start logging using utility, 485
stop logging using utility, 489
records
adddelete
using utility, 829
counting spanned, 549
enabling or disabling spanned, 505
RECORDS keyword, 1285
recovery
building the job stream, 1001
check before starting, 1009
restarting after interruption, 1010
restarting job stream, 1010
skeleton job stream, 1006
using ADARAI RECOVER, 995
recovery aid (see ADARAI utility)
Recovery log
checking content with ADARAI LIST, 982
deactivate using ADARAI DISABLE, 979

description, 975
disabling using ADARAI REMOVE, 1011
display contents using ADARAI LIST, 981
formatting using ADAFRM, 762
initialize and start using ADARAI PREPARE, 991
preparing using ADARAI, 974
preparing with ADARAI PREPARE, 992
units of recovery, 975
recovery log (RLOG)
printing, 141
RECSYNC keyword, 1285
redo pool
command to display usage, 475
reorder Associator
start online process
using utility, 439
reorder Data Storage
start online process
using utility, 443
reorder file
start online process
using utility, 447
repeated syntax elements, 11
replication
activating or deactivating, 531
replication cross-check, using ADADBS OPERCOM RPLCHECK command, 492
REPLICATION parameter, 1357
replication pool
command to display usage, 475
Replication report, 1378
Replication Summary report, 1383
replication-related reporting, 1378
REPLICATOR parameter, 814, 947
Report
database status checkpoint information, 1067
REPORTFILE parameter, 1358
Reports
database status description, 1039
reports
ADAWRK utility, 1361
checkpoint record, 1377
database status file information, 1054
File, 1366
replication-related, 1378
Summary, 1363
Transaction, 1370
work part 1 recovery report description, 1361
REPTOR parameter, 583, 589
required syntax elements, 11
resources
display current usage
using utility, 475
statistics
command to display, 475
RESTONL function, 1154
RESTPLOG function, 1154
resume normal processing
ADADBS function, 553
revert database
to lower version, 311
REVIEW
operator command
using utility, 491
RLOG (see Recovery log)
RPLCHECK
operator command, 492
RPLDATA parameter, 1095, 1104, 1131
RPLDBSI parameter, 814
RPLERRORDEACTFILE parameter, 815
RPLINITERROR parameter, 815
RPLKEY parameter, 816
RPLLOAD parameter, 816, 836
RPLTARGETID parameter, 817
RPLUPDATEONLY parameter, 817, 882

S

save tape
copy using ADARES COPY utility, 1111
SAVETAPE parameter, 1032
search buffer
start logging using utility, 485
stop logging using utility, 489
secondary address converter
allocate an extent using ADADBS utility, 338
deallocate an extent using the ADADBS utility, 352
SECUID
operator command
using utility, 492
SECUID option, 1297
SECUID parameter
ADAWRK utility, 1358
security pool
command to display usage, 475
security user ID
stop user with specified ID, 493-494
session
cancel immediately
using utility, 464
display current parameters
using utility, 473
reset statistical values for
using utility, 515
stop
using utility, 484
terminate normally
using utility, 454
SESSNUM option, 1297
SET DST settings, 1282
SET GLOBALS settings, 1280
SET TZ settings, 1282
SKIP instruction, 1307
SLOG parameter, 817, 948
Sort data set
formatting using ADAFRM, 761
sort data set(SORT)
printing, 145
space
calculating for file coupling lists using ADAINV, 780
estimation report (ADACMP), 290
for temporary file coupling, 778
recover
using utility, 509
space allocated to database components, 1042
space notifications
Associator, 457
Data Storage, 466

spanned records
counting, 549
enabling or disabling for a file, 505
special descriptors, 1066
SQL
null representation support, 246
STARTING FROM clause, 1286
STOPF
operator command
using utility, 492
STOPI
operator command
using utility, 493
STOPSU
operator command, 493
STOPSUR
operator command, 494
STOPU
operator command
using utility, 494
STOPUR
operator command
using utility, 495
storage
fixed (FI), 238
subdescriptor
define
using ADAINV, 784
defining using ADACMP COMPRESS, 261
subfield
defining using ADACMP COMPRESS, 263
subparameter syntax, 11
subparameters
specifying, 13
SUBSCRIPTION parameter, 538
SUMMARY parameter, 1358
Summary report, 1363
superdescriptor
define
using ADAINV, 784
defining using ADACMP COMPRESS, 264
superfield
defining using ADACMP COMPRESS, 273
suspend normal processing
ADADBS function, 553
SYN1 parameter, 1032
SYN4 parameter, 1032
SYNCC
operator command
using utility, 496
Syntax conventions
indentation, 11
syntax conventions
bold, 9
braces ({}), 11
brackets ([]), 10
defaults, 10
ellipsis (...), 11
italic, 10
lowercase, 10
minimum keywords, 10
mixed case, 10
mutually exclusive choices, 10
normal font, 10
optional elements, 10
punctuation and symbols, 11
repeated elements, 11
required elements, 11
subparameters, 11
underlining, 10
uppercase, 9
vertical bars (|), 10
system fields
ADACMP utility requirements, 189
CR field option, 236
SY field option, 244

T

table of ISNs pool
command to display usage, 475
table of sequential commands pool
command to display usage, 475
TARGET parameter, 534
Temp data set
ADALOD requirements for, 847
formatting using ADAFRM, 762
space allocation
using utility, 824
temporary data set(TEMP)
printing, 149
TEST parameter, 1095, 1104, 1131, 1277, 1358
threads
display status
using utility, 477
time zone (TZ) field option
description, 245
timeout control
interregion communication limit
command to override setting, 466
non-activity limit set for access-only users, 496
non-activity limit set for ET logic users, 496
non-activity limit set for exclusive control users, 496
TIMEZONE parameter, 1358
TNAA
operator command
using utility, 496
TNAE
operator command
using utility, 496
TNAX
operator command
using utility, 496
TOBLK parameter, 1095, 1104, 1131
TOCP parameter, 1096, 1104, 1131
TODATE parameter, 1027
TOPLOG parameter, 1096, 1131
TOSESSION parameter, 1027
transaction
set time limit for ET logic users
using utility, 496
transaction ID (XID) pool
command to display usage, 475
transaction processing
suspend/resume, 553
Transaction report, 1370
TRANSACTIONS parameter, 1358
TT

operator command
using utility, 496

U

underlining, 10
unique descriptor
define
using ADALOD, 819
exclude PE instance, 246
use in ADACMP, 246
unique descriptor pool
command to display usage, 475
Universal Encoding Support (UES)
no conversion field option (NV), 243
UNLOCKF
operator command
using utility, 497
UNLOCKU
operator command
using utility, 497
UNLOCKX
operator command
using utility, 497
update command
recsync, 1315
special, 1315
upper index
allocate an extent using ADADBS utility, 338
deallocate an extent using the ADADBS utility, 352
printing, 153
uppercase, 9
user data
start logging using utility, 485
stop logging using utility, 489
user exits
6, used with ADACMP utility, 201
9, used with ADAULD, 1343
ADACDU, 33
hyperdescriptor, 256
user queue
command to display usage, 475
user queue element
display
using utility, 467
remove stopped, 492
using utility, 493
user queue elements
display all
using utility, 478
display for specified user
using utility, 478
display up to five
using operator command, 477
display utility
using utility, 479
user queue file list pool
command to display usage, 475
USERDATA option, 1297
USERID option, 1297
USERID parameter, 1359
users
change priority
using utility, 499
display count of
using utility, 472
resynchronize all ET logic users, 496
set non-activity time limit
using utility, 496
stop those timed out
using utility, 493
stop those using a specified file
using utility, 492
stop those with a specified job name
using utility, 494-495
stop those with a specified security user ID, 493-494
stop user with specified ID
using utility, 494-495
USERTID option, 1297
utilities
ADADBS function, 535
sequential list of files, 1417
utility control statement
parameter values, 12
default, 10
value, 12
value list, 13
value range, 13
rules, 11
syntax, 8
parameter list, 9
syntax symbols, 9
utility-only status
switch on/off
using utility, 497
UTIONLY
operator command
using utility, 497

V

value buffer
start logging using utility, 485
stop logging using utility, 489
value-criterion syntax, 1289
variable-length field size
ADACMP utility requirements, 190
vertical bars (|), 10

W

wide-character fields
no conversion option (NV), 243
Work data set
formatting using ADAFRM, 761
print data protection records using ADAPLP, 960
resetting blocks/cylinders to zeros using ADAFRM, 762
work data set(WORK)
printing, 167
Work file
define
using utility, 591
Work part 1
command to display usage, 475
Work part 2
command to display usage, 475
Work part 3
command to display usage, 475

work pool
command to display usage, 475

X

XID pool
command to display usage, 475