Adabas

Utilities Manual, Volume 2

software AG

Manual Order Number: ADA741-081IBB

This document applies to Adabas Version 7.4 and to all subsequent releases.

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

Readers' comments are welcomed. Comments may be addressed to the Documentation Department at the address on the back cover or to the following e-mail address:

Documentation@softwareag.com

© December 2002, Software AG
All rights reserved
Printed in the Federal Republic of Germany

Software AG and/or all Software AG products are either trademarks or registered trademarks of Software AG. Other products and company names mentioned herein may be the trademarks of their respective owners.

TABLE OF CONTENTS

GENERAL INFORMATION 1

About This Manual 1
Utility Control Statements 2
Control Statement Syntax 2
Control Statement Rules 6
Parameter Values 6

13. ADAORD : REORDER 11

Functional Overview 11
Reorder Functions 11
Restructure Functions 11
Store Function 11
Notes 12
Space Allocation 12
Function Directory 13
REORASSO : Reorder Associator 13
Optional Parameters and Subparameters 14
Examples 19
REORDATA : Reorder Data Storage 20
Optional Parameters and Their Subparameters 21
Examples 25
REORDB : Reorder Database 25
Optional Parameters and Subparameters 27
Examples 34
REORFASSO : Reorder Associator for a Single File 36
Essential Parameter 37
Optional Parameters 37
Examples 41

REORFDATA : Reorder Data Storage for a Single File 42
Essential Parameter 42
Optional Parameters 42
Examples 46
REORFILE : Reorder File 47
Essential Parameter 48
Optional Parameters 48
Examples 54
RESTRUCTUREDB : Restructure Database 55
Optional Parameters and Subparameters 56
Examples 59
RESTRUCTUREF : Restructure Single Files 60
Essential Parameter 61
Optional Parameters 61
Examples 64
STORE : Store Files 65
Optional Parameters and Subparameters 66
Examples 74
JCL/JCS Requirements and Examples 75
BS2000 76
OS/390 or z/OS 80
VM/ESA or z/VM 83
VSE/ESA 86

14. ADAPLP : PROTECTION LOG/WORK PRINT 89

Print Protection Data 89
Optional Parameters and Subparameters 90
Examples 93
JCL/JCS Requirements and Examples 93
BS2000 93
OS/390 or z/OS 97
VM/ESA or z/VM 100
VSE/ESA 102

15. ADAPRI : PRINT SELECTED ADABAS BLOCKS 105

Print Blocks 105
Essential Parameters 105
Optional Parameters 105
Examples 106
JCL/JCS Requirements and Examples 108
BS2000 108
OS/390 or z/OS 110
VM/ESA or z/VM 112
VSE/ESA 114

16. ADARAI : ADABAS RECOVERY AID 117

Concepts and Components 117
The Collection Interface 117
Recovery Log (RLOG) 117
Generation : The Unit of Recovery 118
Retaining Noncurrent Generations 119
Function Overview 119
CHKDB: Check the Database Status 120
DISABLE : Deactivate Recovery Logging 121
LIST : Display Current RLOG Generations 122
Additional LIST Information on BS2000 123
Syntax 123
Optional Parameters 124
Examples 125
PREPARE : Initialize and Start the RLOG 135
Syntax 136
Essential Parameter 136
Optional Parameters 136
Examples 137
RECOVER : Build a Recovery Job Stream 138
Recovery Processing 139
Optimized Recovery Processing 140
Requirements 141
Restrictions 141

Input Needed for Recovery 142
Output from the Recovery Operation 143
Executing the RECOVER Function 143
File-Level Recovery 145
Syntax 146
Optional Parameters and Subparameters 146
Examples 150
Skeleton Job Control 151
User Exit to Change JCL 155
Prerecovery Checking 155
Restarting the RECOVER Function or Recovery Job Stream 156
REMOVE : Remove the Recovery Aid 157
Example 157
JCL/JCS Requirements and Examples 158
BS2000 158
OS/390 or z/OS 175
VM/ESA or z/VM 181
VSE/ESA 183

17. ADAREP : REPORT 185

Functional Overview 185
Syntax 186
Optional Parameters 186
Examples 189
Processing Save Tape Input 190
Supplying Protection Log Input 191
Checking Input Tapes 192
Concurrent Parameters 192
Reports for Delta Save Tapes 192
Report Layout 193
Report Description 194
General Database Information 194
File Information 203
Checkpoint Information 212

JCL/JCS Requirements and Examples 217
BS2000 217
OS/390 or z/OS 219
VM/ESA or z/VM 221
VSE/ESA 222

18. ADARES : RESTART 225

Functional Overview 225
Information about Using ADARES 226
Using ADARES in Adabas Nucleus Cluster Environments 227
BACKOUT Functions 230
BACKOUT : Back Out Updates Using the Sequential Protection Log (SIBA) 231
Essential Parameters 232
Optional Parameters and Subparameters 232
Examples 237
BACKOUT DPLOG or MPLOG : Back Out Updates Using the Dual or Multiple Protection Log 238
Executing the Function 238
Syntax 239
Essential Parameter 240
Optional Parameters 240
Example 245
CLCOPY : Copy Dual Command Log 246
Optional Parameters 246
Examples 247
COPY : Copy a Sequential Protection Log or Save Tape 248
Optional Parameters 249
Examples 251
MERGE CLOG : Merge Nucleus Cluster Command Logs 252
Essential Parameter 252
PLCOPY : Copy Protection Log to Sequential Dataset 253
Optional Parameters 253
Examples 255

REGENERATE : Regenerate Updates 256
Syntax 257
Essential Parameters 257
Optional Parameters and Subparameters 258
Examples 264
REPAIR : Repair Data Storage Blocks 266
Notes 266
Syntax 267
Essential Parameter 267
Optional Parameters 267
Examples 268
Multithreaded Processing Statistics 268
JCL/JCS Requirements and Examples 270
BS2000 271
OS/390 or z/OS 279
VM/ESA or z/VM 288
VSE/ESA 293

19. ADASAV : SAVE/RESTORE DATABASE OR FILES 299

Functional Overview 299
RESTONL and RESTORE Functions 299
RESTPLOG and RESTONL Functions 300
Online and Offline SAVEs 300
ADASAV Function Directory 300
RESTONL : Restore Database from Online Source 301
Conditions 301
Result 301
Notes 302
Syntax 302
Optional Parameters and Subparameters 302
Examples 305
RESTONL GCB : Restore Database Incremental from Online Source 305
Conditions 306
Result 306
Notes 307
Syntax 307

Optional Parameters and Subparameters 307
Examples 310
RESTONL FILES : Restore Files to Original RABNs from Online Source . . . 311
Conditions 311
Result 312
Notes 312
Syntax 313
Optional Parameters and Subparameters 313
Examples 317
RESTONL FMOVE : Restore Files to Any RABNs from Online Source 318
Conditions 318
Result 319
Notes 319
Syntax 320
Optional Parameters 321
Examples 327
RESTORE : Restore Database from Offline Source 329
Conditions 329
Result 329
Notes 329
Syntax 330
Optional Parameters 330
Examples 332
RESTORE GCB : Restore Database Incremental from Offline Source 333
Conditions 333
Result 334
Notes 334
Syntax 334
Optional Parameters 335
Examples 337
RESTORE FILES : Restore Files to Original RABNs from Offline Source . . . 337
Conditions 338
Result 339
Notes 339
Syntax 339
Optional Parameters 340

Examples 342
RESTORE FMOVE :
Restore Files to Any RABNs from Offline Source 343
Conditions 343
Result 344
Notes 344
Syntax 344
Optional Parameters 345
Examples 351
RESTPLOG : Restore Protection Log Only 352
Essential Parameters 353
Optional Parameters 353
Example 354
SAVE : Save Database 355
Syntax 356
Optional Parameters 356
Example 358
SAVE FILES : Save Specified Files 358
Syntax 360
Optional Parameters 360
Examples 363
JCL/JCS Requirements and Examples 363
BS2000 364
OS/390 or z/OS 371
VM/ESA or z/VM 377
VSE/ESA 380

20. ADASEL : SELECT PROTECTION DATA 385

Functional Overview 385
Syntax 385
SELECT Statement 387
Additional Selection Criteria 388
date-time 389
WITH Clause 390
IF-statement 391

value-criterion 392
output-instruction 398
Overriding ADASEL Defaults with Global Parameters 407
JCL/JCS Requirements and Examples 409
BS2000 409
OS/390 or z/OS 411
VM/ESA or z/VM 412
VSE/ESA 413

21. ADAULD : UNLOAD 415

Functional Overview 415
UNLOAD FILE : Unload Specified File 416
Essential Parameter 417
Optional Parameters and Subparameters 417
Examples 423
ADAULD Input Processing 425
Processing a Save Tape as Input 425
ADAULD Output Processing 427
ADAULD User Exit 9 428
JCL/JCS Requirements and Examples 429
BS2000 429
OS/390 or z/OS 432
VM/ESA or z/VM 434
VSE/ESA 436

22. ADAVAL : VALIDATE THE DATABASE 439

Functional Overview 439
VALIDATE : Validate Data Storage and Associator 440
Essential Parameters 441
Optional Parameters 441
Example of ADAVAL Output 443

JCL/JCS Requirements and Examples 444
Collation with User Exit 444
Sorting Large Files 445
BS2000 445
OS/390 or z/OS 447
VM/ESA or z/VM 448
VSE 449

23. ADAZAP : DISPLAY OR MODIFY ASSO, DATA, AND WORK DATASETS 451

Functional Overview 451
Syntax 452
Essential Parameters 452
Optional Parameters 452
Examples 453
JCL/JCS Requirements and Examples 454
BS2000 454
OS/390 or z/OS 456
VM/ESA or z/VM 457
VSE/ESA 458

APPENDIX A : ADABAS SEQUENTIAL FILES 459

Sequential File Table 459
Operating System Dependencies 463
BS2000 Systems 463
OS/390 or MVS/ESA Systems 468
VM/ESA Systems 469
VSE/ESA Systems 470

APPENDIX B : PROCEDURES FOR VSE/ESA EXAMPLES 475

APPENDIX C : SUPPLIED UES ENCODINGS 477

Interoperable Encodings 478

Single-Byte Character Sets (Latin–1) 478

Single-Byte Character Sets (Non-Latin–1) 479

Double- and Multiple-Byte Character Sets 480

Coexistent Encodings 481

Single-Byte Character Sets 481

Double- and Multiple-Byte Character Sets 482

INDEX 485

GENERAL INFORMATION

Note:
Dataset names starting with DD are referred to in Adabas manuals with a slash separating the DD from the remainder of the dataset name to accommodate VSE/ESA dataset names that do not contain the DD prefix. The slash is not part of the dataset name.

About This Manual

The *Adabas Utilities Manual* comprises two volumes. The utilities are ordered alphabetically by utility name over both volumes: volume 1 contains chapters for the utilities ADAACK through ADAMER; volume 2 contains chapters for the utilities ADAORD through ADAZAP. The chapters are also numbered consecutively over both volumes.

This chapter and appendixes A, B, and C are included in both volumes for your convenience.

Each Adabas utility is described in a separate chapter. For a single-function utility, the chapter begins with a syntax diagram showing the utility statement and all possible parameters. Chapters 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 BS2000, z/OS and OS/390, z/VM and VM/ESA, and VSE/ESA operating systems.

Utility Control Statements

Control Statement Syntax

Utility control statements have the following format:

```
utility   function   parameter-list
```

—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.

Parameter List

In the statement syntax descriptions in this manual, parameters are listed vertically; that is, “stacked”. 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.

Required and Optional Parameters

Parameters can be required or optional. Optional parameters (or parameter groups) are shown in square brackets [ ]. For example, the ADADBS CHANGE function has the following parameters:

```
ADADBS CHANGE  FILE=file-number
               FIELD='field-name'
               LENGTH=new-length
               [PASSWORD='password']
               [TEST]
```

The ADADBS CHANGE operation changes the standard length of an Adabas field. The FILE that contains the field to be changed must be specified, as well as the FIELD and the new standard LENGTH for the field. The PASSWORD parameter must be specified only if the specified file is password-protected. You can choose to specify the TEST parameter or not; there are no circumstances where it is required.

Some optional parameters have default values that are used if the parameter is not specified. Default values are indicated by underlined values (see the section **Default Parameter Values** on page NO TAG).

Subparameters

Indented parameter lists are **subparameters**; they cannot be specified unless the parameter above the list is also specified. The following example shows the syntax for subparameters of the FILE parameter:

```
utility function   [FILE=file-number]
                        DATAPFAC=padding-factor
                        DSRABN=startng-rabn
                        [DSSIZE=size]
                        [NU]
```

FILE is an optional parameter. If FILE= is actually specified, then DATAPFAC and DSRABN must also be specified. Note that any subparameter that is not enclosed in square brackets is required, and listed first. The remaining parameters indented under FILE and enclosed in square brackets can optionally be specified when the FILE parameter is specified.

Mutually Exclusive Parameters

Parameters that cannot be specified with each other are called **mutually exclusive** parameters, and are shown in curly brackets, otherwise known as braces { }.

When parameters are stacked and enclosed in braces, one parameter must be specified, but only one can be specified. For example, in the following parameter list:

```
ADADBS INCREASE      { ASSOSIZE=size | DATASIZE=size }
                     [ TEST ]
```

—either the parameter ASSOSIZE or DATASIZE must be specified when executing ADADBS INCREASE; but not both.

Optionally, the parameter TEST may be specified with either ASSOSIZE or DATASIZE.

The following example shows just one of the parameter options of ADAREP REPORT:

```
ADAREP [REPORT] [ { LIMCOUNT | NOCOUNT } ]
```

In this case, the braces { } within the square brackets [ ] indicate that, if you choose to specify the parameter option at all, either LIMCOUNT or NOCOUNT may be specified.

ADAREP reads the address converter to determine the value for RECORDS LOADED for a file. For very large records, 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 appears under RECORDS LOADED for any file.
- If neither LIMCOUNT nor NOCOUNT is specified, ADAREP compiles the exact value for RECORDS LOADED for each file.

Repeating Parameters and Values

Some parameters can be repeated, and some parameters can specify multiple values. An ellipsis (...) following the parameter or value indicates that it may be repeated. A comma-ellipsis (,...) following the parameter or value indicates that it may be repeated and that repetitions must be separated by a comma.

For longer parameters, parameter combinations, or other lengthy values that can be repeated, the repeatable portion of the syntax is enclosed in braces, followed immediately by a separating comma and ellipsis.

In the following example, FILECRIT and FILEVAL use the '{ },...' notation to indicate that multiple complex criteria and value entries may be specified, respectively.

```
ADALOD LOAD     FILE=file-number [ , file-type ]
                DSSIZE=size
                MAXISN=maximum-number-of-records
                SORTSIZE=size
                TEMPSIZE=size
                FILECRIT={ '{file-selection-criteria}, ...', FILEVAL='value, ... ' |
                                'ISN', MINISN=lowest-allocated-isn  }
```

Some utilities allow a function or parameter itself to be specified multiple times, usually each on separate function statements. For instance, ADACMP field definition statements (FNDEF) might be specified as follows:

```
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)'
```

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 1–6.
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 “=”.
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 1 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 1–6.

Parameter Values

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

```
parameter = value
parameter = value-list
parameter = value-range
```

—where “value” is as described in the following sections. “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

"Value" 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; see the section **Default Parameter Values** on page NO TAG for more information.

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=3380**
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.

Default Parameter Values

In the ADADBS ADD function, for example, the optional ASSODEV parameter can specify the device type where the ADADBS ADD function allocates the new dataset space; if ASSODEV is not specified, a currently effective device type (the “default”) is assumed—in this case, a 3380 disk device. Where appropriate, default values or settings are shown underlined in the syntax statement:

[**FILE=** { file-number | **1** }]

Otherwise, the defaults are explained in the accompanying parameter descriptions. If desired, the default value can be explicitly entered (in this example, FILE=1).

If the default is a value originally established by some other control statement, the parameter name is shown in underlined italics as the default:

[**TTSYN=**{ seconds | ADARUN-tt *}*]

Here, the default is the value already set by the TT parameter in the Adabas runtime control (ADARUN) statement, which is in effect if TTSYN is not specified. The value “tt” should not actually be entered.

ADAORD : REORDER

Functional Overview

Only one function may be executed during a given execution of ADAORD.

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.

Restructure Functions

The RESTRUCTUREDB function unloads an entire database to a sequential dataset; the RESTRUCTUREF function unloads one or more files to a sequential dataset. This dataset can be used as input to the STORE function.

The RESTURCTURE functions are used to relocate the database to a different physical device or a file or files to another device.

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.

Notes

1. The format of the sequential dataset 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.

2. The Associator and Data Storage are reordered as part of RESTRUCTURE/STORE processing.

3. Parts of the database are overwritten during ADAORD execution. It is therefore recommended that the database (or file) be backed up with the ADASAV utility before running ADAORD functions.

4. 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.

5. 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 either the checkpoint or security files require exclusive database control.

6. If the specified file was originally loaded with ISNREUSE=YES active, a STORE function and all ADAORD reorder functions that affect the file's Associator 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.

7. When specifying the starting RABN for Associator extents, the space needed for the FCBs, FDTs, and DSST should also be considered.

8. 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/LOD utilities to move ADAM files, instead.

Space Allocation

ADAORD allocates the amount of space required by the xxSIZE or MAXISN parameters, if specified. Otherwise, ADAORD allocates space based on the current size of the file. Note that the xxRELEASE parameters affect the amount of space required.

If possible, space is allocated on the volume specified by the xxxxVOLUME 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 xxRABN parameter overrides the associated xxxxVOLUME parameter.

Function Directory

Function	Action	Page
REORASSO	reorder the entire Associator	13
REORDATA	reorder the entire Data Storage	20
REORDB	reorder the entire Associator and Data Storage	25
REORFASSO	reorder Associator for a single file	36
REORFDATA	reorder Data Storage for a single file	42
REORFILE	reorder Associator and Data Storage for a single file	47
RESTRUCTUREDB	unload an entire database for the purpose of relocating it to a different physical device	55
RESTRUCTUREF	unload one or more files for the purpose of relocating them to another device	60
STORE	load one or more files into an existing database using the output produced by the RESTRUCTURE functions or the REORDB function	65

REORASSO : Reorder Associator

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.

```
ADAORD REORASSO    [DBINDEXCOMPRESSION={YES | NO} ]
                   [FILE=file-number ]
                        [ACRABN=starting-rabn ]
                        [ALLOCATION={FORCE | NOFORCE} ]
                        [ASSOPFAC=padding-factor ]
                        [ASSOVOLUME='Associator-extent-volume' ]
                        [INDEXCOMPRESSION={YES | NO} ]
                        [ISNSIZE= { 3|4 } ]
                        [MAXISN=highest-isn ]
                        [NIRABN=starting-rabn ]
                        [NIRELEASE]
                        [NISIZE=size ]
                        [UIRABN=starting-rabn ]
                        [UIRELEASE ]
                        [UISIZE=size ]
                   [LPB={ prefetch-buffer-size | ADARUN-lu } ]
                   [MAXFILES=maximum-number-files ]
                   [NEWDBID=database-identifier ]
                   [NEWDBNAME=database-name ]
                   [NOUSERABEND]
                   [RAID]
                   [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.

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 blocksize, whichever is lower. For example, 2003 is the highest MAXFILES value for a database whose ASSOR1 is stored on a 3380 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.

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 blocksize 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 “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 an error is encountered while the function is running, the utility 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.

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).

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 blocksize 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 "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 1,000. The normal index for file 1 is to begin with RABN 2,200. The MAXISN for file 2 is to be set to 500,000. 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.

REORDATA : Reorder Data Storage

The REORDATA function reorders Data Storage for **all** files. Files not specified are reordered according to their existing definitions.

```
ADAORD REORDATA    [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 ]
                       [SORTSEQ={ descriptor | ISN | physical-sequence } ]
                   [LIP={ isn-pool-size | 16384 } ]
                   [LPB={ prefetch-buffer-size | ADARUN-lu } ]
                   [NOUSERABEND]
                   [RAID]
                   [TEST]
```

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 blocksize 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 "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 “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).

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 DATA dataset for the file can be 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 ERROR–126 “Data Storage record too long”.

If MAXRECL is not specified, the maximum compressed record length does not change.

NOUSERABEND : Termination without ABEND

When an error is encountered while the function is running, the utility 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.

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 1,000 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.

REORDB : Reorder Database

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.

```
ADAORD REORDB  [DBINDEXCOMPRESSION={YES | NO} ]
               [FILE=file-number  ]
                    [ACRABN=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 ]
                    [MAXRECL=record-length ]
                    [NIRABN=starting-rabn ]
                    [NIRELEASE]
                    [NISIZE=size ]
                    [SORTSEQ={ descriptor | ISN | physical-sequence } ]
                    [UIRABN=starting-rabn ]
                    [UIRELEASE]
                    [UISIZE=size ]
               [LIP={ isn-pool-size | 16384 } ]
               [LPB={ prefetch-buffer-size | ADARUN-lu } ]
               [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.

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 blocksize 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 "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 “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).

LPB : Prefetch Buffer Size

LPB specifies the size, in bytes, of the internal prefetch buffer. The maximum value is 32,760 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 blocksize, whichever is lower. For example, 2003 is the highest MAXFILES value for a database whose ASSOR1 is stored on a 3380 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.

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 dataset 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").

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 1–65,535 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 blocksize 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 “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 an error is encountered while the function is running, the utility 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.

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 : Text 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 blocksize 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 “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 1,000. The normal index allocation for file 1 is to begin with RABN 2,200.

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 500,000. 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.

REORFASSO : Reorder Associator for a Single File

The REORFASSO function reorders the Associator for a single file. Associator information for unspecified files is not reordered.

```
ADAORD REORFASSO   FILE=file-number
                       [ACRABN=starting-rabn ]
                       [ALLOCATION={FORCE | NOFORCE} ]
                       [ASSOPFAC=padding-factor ]
                       [ASSOVOLUME='Associator-extent-volume' ]
                       [INDEXCOMPRESSION={YES | NO} ]
                       [ISNSIZE= { 3 | 4 } ]
                       [MAXISN=highest-isn ]
                       [NIRABN=starting-rabn ]
                       [NIRELEASE]
                       [NISIZE=size ]
                       [PASSWORD=password ]
                       [UIRABN=starting-rabn ]
                       [UIRELEASE]
                       [UISIZE=size ]
                   [EXCLUDE=file-list ]
                   [LPB={ prefetch-buffer-size | ADARUN-lu } ]
                   [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.

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 32,760 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.

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 blocksize 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 "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 an error is encountered while the function is running, the utility 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.

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 blocksize 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 “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.

REORFDATA : Reorder Data Storage for a Single File

The REORFDATA function reorders Data Storage for a single file. Data Storage for unspecified files is not reordered.

```
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 | physical-sequence } ]
                    [EXCLUDE=file-list ]
                    [LIP={ isn-pool-size | 16384 } ]
                    [LPB={ prefetch-buffer-size | ADARUN-lu } ]
                    [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 blocksize 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 “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 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 “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).

LPB : Prefetch Buffer Size

LPB specifies the size, in bytes, of the internal prefetch buffer. The maximum value is 32,760 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 DATA dataset for the file can be 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”).

If MAXRECL is not specified, the maximum compressed record length does not change.

NOUSERABEND : Termination without ABEND

When an error is encountered while the function is running, the utility 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.

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.

REORFILE : Reorder File

The REORFILE function reorders the Associator and Data Storage for a single file. Associator and Data Storage for other files are not affected.

```
ADAORD REORFILE     FILE=file-number
                        [ACRABN=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 ]
                        [MAXRECL=record-length ]
                        [NIRABN=starting-rabn ]
                        [NIRELEASE]
                        [NISIZE=size ]
                        [PASSWORD=password ]
                        [SORTSEQ={ descriptor | ISN | physical-sequence } ]
                        [UIRABN=starting-rabn ]
                        [UIRELEASE]
                        [UISIZE=size ]
                    [EXCLUDE=file-list ]
                    [LIP={ isn-pool-size | 16384 } ]
                    [LPB={ prefetch-buffer-size | ADARUN-lu } ]
                    [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.

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 blocksize 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 "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 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 “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).

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.

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 DATA dataset for the file can be 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 ERROR–126 “Data Storage record too long”.

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 blocksize 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 “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 an error is encountered while the function is running, the utility 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.

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 blocksize 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 “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 cylinders 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
```

The Associator for files 9 and 23 is to be reordered; other files are not changed.

The Associator padding factor is set to 5% and the Data Storage padding factor to 15% for files 9, 10, and 12. For file 23, the new starting Data Storage RABN is 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.

RESTRUCTUREDB : Restructure Database

The RESTRUCTUREDB function unloads an entire database to a sequential dataset, 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.

```
ADAORD {RESTRUCTUREDB | REDB}
            [DBASSODEV={ device-type | ADARUN-device } ]
            [DBDATADEV={ device-type | ADARUN-device } ]
            [DBINDEXCOMPRESSION={YES | NO} ]
            [FILE=file-number ]
                [ASSOPFAC=padding-factor ]
                [ASSODEV={ device-type | DBASSODEV-value } ]
                [DATADEV={ device-type | DBDATADEV-value } ]
                [DATAPFAC=padding-factor ]
                [INDEXCOMPRESSION={YES | NO} ]
                [ISNSIZE={ 3 | 4 } ]
                [SORTSEQ={ descriptor |ISN  | physical-sequence} ]
            [LPB={ prefetch-buffer-size|ADARUN-lu } ]
            [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 dataset. 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 / FILEA dataset.

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 dataset. 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 dataset. 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.

DBDATADEV : Default Data Storage Device Type

DBDATADEV specifies a default device type for the new DATA dataset. 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.

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 an error is encountered while the function is running, the utility 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.

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.

RESTRUCTUREF : Restructure Single Files

The RESTRUCTUREF function unloads one or more files to a sequential dataset, 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.

```
ADAORD {RESTRUCTUREF | REF}
            FILE=file-number
                [ASSODEV={ device-type | DBASSODEV-value } ]
                [ASSOPFAC=padding-factor ]
                [DATADEV={ device-type | DBDATADEV-value } ]
                [DATAPFAC=padding-factor ]
                [DBASSODEV={ device-type | ADARUN-device } ]
                [DBDATADEV={ device-type | ADARUN-device } ]
                [INDEXCOMPRESSION={YES | NO} ]
                [ISNSIZE={3 | 4 } ]
                [PASSWORD=password ]
                [SORTSEQ={ descriptor | ISN | physical-sequence} ]
            [LPB={ prefetch-buffer-size|ADARUN-lu } ]
            [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 dataset. 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 / FILEA dataset.

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 dataset. 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 dataset. 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.

DBDATADEV : Default Data Storage Device Type

DBDATADEV specifies a default device type for the new DATA dataset. 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.

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 32,760 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 an error is encountered while the function is running, the utility 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.

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.

STORE : Store Files

The STORE function loads one or more files into an existing database using output produced by the RESTRUCTURE functions.

If the ALLFILES parameter is specified, all files contained on the input dataset 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.

Notes:

1. *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 ADAULD and ADALOD.*
2. *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.*

```
ADAORD STORE    [ALLFILES]
                [CHECKPOINT]
                [FILE=file-number ]
                     [ACRABN=start-rabn ]
                     [ALLOCATION={FORCE | NOFORCE} ]
                     [ASSOPFAC=padding-factor ]
                     [ASSOVOLUME='Associator-extent-volume' ]
                     [CHECKPOINT]
                     [DATAPFAC=padding-factor ]
                     [DATAVOLUME='Data-Storage-extent-volume' ]
                     [DSDEV=device-type ]
                     [DSRABN=start-rabn ]
                     [DSRELEASE]
                     [DSSIZE=size ]
                     [INDEXCOMPRESSION={YES | NO} ]
                     [ISNSIZE= { 3 | 4 } ]
                     [MAXISN=highest-isn ]
                     [MAXRECL=record-length ]
                     [NIRABN=start-rabn ]
                     [NIRELEASE]
                     [NISIZE=size ]
                     [UIRABN=start-rabn ]
                     [UIRELEASE]
                     [UISIZE=size ]
                [EXCLUDE=file-list ]
                [LIP={ isn-pool-size | 16384 } ]
                [NOUSERABEND]
                [OVERWRITE]
                [SECURITY]
                [TEST]
                [TRIGGER]
```

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.

ALLFILES : Select All Files for Storing

ALLFILES causes all files in the input dataset 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 dataset contains files that are coupled or part of an expanded file and the related files are not in the dataset, ERROR–138 is returned indicating an inconsistent file list. You must add the related files before the STORE function will execute successfully.

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/FILEA 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/DDFILEA 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 blocksize 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 “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 may be specified only if ALLFILES is also specified.

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 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 “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).

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.

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 DATA dataset for the file can be 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 ERROR–126 “Data Storage record too long”.

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 blocksize 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 “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 an error is encountered while the function is running, the utility prints an error message and terminates with user ABEND 034 (with a dump) or user ABEND 035 (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.

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.

SECURITY : Store the Security File

SECURITY stores the security file from the DDFILEA / FILEA 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 FILEA / DDFILEA tape is not stored in the database, even if it is specified by a FILE parameter or the ALLFILES parameter is 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; not the validity of values and variables.

TRIGGER : Store the Trigger File

Specify the TRIGGER parameter to store the trigger file from the DDFILEA/FILEA 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/FILEA 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 blocksize 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 “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 500,000. 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 dataset 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 dataset (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 12,345. 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.

JCL/JCS Requirements and Examples

This section describes the job control information required to run ADAORD with BS2000, OS/390 or z/OS, VM/ESA or z/VM, and VSE/ESA systems and shows examples of each of the job streams.

Note:
When running with the optional Recovery Aid (ADARAI) for RESTRUCTUREXx or STORE functions, all temporary datasets must also be cataloged in the job control.

BS2000

Dataset	Link Name	Storage	More Information
Associator	DDASSORn	disk	
Data Storage	DDDATARn	disk	
Intermediate storage	DDFILEA	tape/disk	
Recovery log (RLOG)	DDRLOGR1	disk	Required when using the ADARAI option
ADARUN parameters	SYSDTA/DDCARD		*Operations Manual*
ADAORD parameters	SYS/DTA/DDKARTE		
ADARUN messages	SYSOUT/DDPRINT		*Messages and Codes*
ADAORD messages	SYSLST/DDDRUCK		*Messages and Codes*

ADAORD JCL Examples (BS2000)

Reorder File Data Storage, Reorder File, Reorder Data, Reorder Database

In SDF Format:

```
/.ADAORD LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK * A D A O R D REORDER FILE DATA, REORDER FILE, REORDER DATA
/REMARK * REORDER DATABASE
/REMARK *
/DELETE-FILE ADAyyyyy.FILEA
/SET-JOB-STEP
/CREATE-FILE ADAyyyyy.FILEA,PUB(SPACE=(4800,480))
/SET-JOB-STEP
/ASS-SYSLST L.ORD.DATA
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDASSOR1,ADAyyyyy.ASSO,SHARE-UPD=YES
/SET-FILE-LINK DDDATAR1,ADAyyyyy.DATA,SHARE-UPD=YES
/SET-FILE-LINK DDWORKR1,ADAyyyyy.WORK,SHARE-UPD=YES
/SET-FILE-LINK DDFILEA,ADAyyyyy.FILEA
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADAORD,DB=yyyyy,IDTNAME=ADABAS5B
ADAORD REORDATA FILE=1,DSSIZE=80,DATAPFAC=30
/LOGOFF SYS-OUTPUT=DEL
```

In ISP Format:

```
/.ADAORD LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK * A D A O R D REORDER FILE DATA, REORDER FILE, REORDER DATA
/REMARK * REORDER DATABASE
/REMARK *
/SYSFILE SYSLST=L.ORD.DATA
/FILE ADA.MOD,LINK=DDLIB
/FILE ADAyyyyy.ASSO ,LINK=DDASSOR1,SHARUPD=YES
/FILE ADAyyyyy.DATA ,LINK=DDDATAR1,SHARUPD=YES
/FILE ADAyyyyy.WORK ,LINK=DDWORKR1,SHARUPD=YES
/FILE ADAyyyyy.FILEA ,LINK=DDFILEA ,SPACE=(4800,480)
/EXEC (ADARUN,ADA.MOD)
ADARUN PROG=ADAORD,DB=yyyyy,IDTNAME=ADABAS5B
ADAORD REORDATA FILE=1,DSSIZE=80,DATAPFAC=30
/LOGOFF NOSPOOL
```

Reorder Associator

In SDF Format:

```
/.ADAORD LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK * A D A O R D REORDER FILE ASSO, REORDER ASSO
/REMARK *
/DELETE-FILE ADAyyyyy.FILEA
/SET-JOB-STEP
/CREATE-FILE ADAyyyyy.FILEA,PUB(SPACE=(4800,480))
/SET-JOB-STEP
/ASS-SYSLST L.ORD.REOR
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDASSOR1,ADAyyyyy.ASSO,SHARE-UPD=YES
/SET-FILE-LINK DDDATAR1,ADAyyyyy.DATA,SHARE-UPD=YES
/SET-FILE-LINK DDFILEA,ADAyyyyy.FILEA
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADAORD,DB=yyyyy,IDTNAME=ADABAS5B
ADAORD REORFASSO
ADAORD FILE=1,MAXISN=20000,NISIZE=300B
ADAORD FILE=3,NISIZE=400B,ASSOPFAC=2
/LOGOFF SYS-OUTPUT=DEL
```

In ISP Format:

```
/.ADAORD LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK * A D A O R D REORDER FILE ASSO, REORDER ASSO
/REMARK *
/SYSFILE SYSLST=L.ORD.REOR
/FILE ADA.MOD,LINK=DDLIB
/FILE ADAyyyyy.ASSO ,LINK=DDASSOR1,SHARUPD=YES
/FILE ADAyyyyy.DATA ,LINK=DDDATAR1,SHARUPD=YES
/FILE ADAyyyyy.FILEA ,LINK=DDFILEA ,SPACE=(4800,480)
/EXEC (ADARUN,ADA.MOD)
ADARUN PROG=ADAORD,DB=yyyyy,IDTNAME=ADABAS5B
ADAORD REORFASSO
ADAORD FILE=1,MAXISN=20000,NISIZE=300B
ADAORD FILE=3,NISIZE=400B,ASSOPFAC=2
/LOGOFF NOSPOOL
```

Restructure

In SDF Format:

```
/.ADAORD LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK * A D A O R D RESTRUCTURE
/REMARK *
/DELETE-FILE ADAyyyyy.FILEA
/SET-JOB-STEP
/CREATE-FILE ADAyyyyy.FILEA,PUB(SPACE=(4800,480))
/SET-JOB-STEP
/ASS-SYSLST L.ORD.REST
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDASSOR1,ADAyyyyy.ASSO,SHARE-UPD=YES
/SET-FILE-LINK DDRLOGR1,ADAyyyyy.RLOGR1,SHARE-UPD=YES
/SET-FILE-LINK DDFILEA,ADAyyyyy.FILEA
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADAORD,DB=yyyyy,IDTNAME=ADABAS5B
ADAORD RESTRUCTUREF
ADAORD FILE=1,DATADEV=dddd
/LOGOFF SYS-OUTPUT=DEL
```

In ISP Format:

```
/.ADAORD LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK * A D A O R D RESTRUCTURE
/REMARK *
/SYSFILE SYSLST=L.ORD.REST
/FILE ADA.MOD,LINK=DDLIB
/FILE ADAyyyyy.ASSO ,LINK=DDASSOR1,SHARUPD=YES
/FILE ADAyyyyy.RLOGR1 ,LINK=DDRLOGR1,SHARUPD=YES
/FILE ADAyyyyy.FILEA ,LINK=DDFILEA ,SPACE=(4800,480)
/EXEC (ADARUN,ADA.MOD)
ADARUN PROG=ADAORD,DB=yyyyy,IDTNAME=ADABAS5B
ADAORD RESTRUCTUREF
ADAORD FILE=1,DATADEV=dddd
/LOGOFF NOSPOOL
```

Store

In SDF Format:

```
/.ADAORD LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK * A D A O R D STORE
/REMARK *
/DELETE-FILE ADAyyyyy.FILEA
/SET-JOB-STEP
/CREATE-FILE ADAyyyyy.FILEA,PUB(SPACE=(4800,480))
/SET-JOB-STEP
/ASS-SYSLST L.ORD.STOR
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDASSOR1,ADAyyyyy.ASSO,SHARE-UPD=YES
/SET-FILE-LINK DDDATAR1,ADAyyyyy.DATA,SHARE-UPD=YES
/SET-FILE-LINK DDRLOGR1,ADAyyyyy.RLOGR1,SHARE-UPD=YES
/SET-FILE-LINK DDFILEA,ADAyyyyy.FILEA
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADAORD,DB=yyyyy,IDTNAME=ADABAS5B
ADAORD STORE
ADAORD FILE=1,DSSIZE=80,DATAPFAC=30,DSRABN=1234
ADAORD MAXISN=200000
/LOGOFF SYS-OUTPUT=DEL
```

In ISP Format:

```
/.ADAORD LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK * A D A O R D STORE
/REMARK *
/SYSFILE SYSLST=L.ORD.STOR
/FILE ADA.MOD,LINK=DDLIB
/FILE ADAyyyyy.ASSO ,LINK=DDASSOR1,SHARUPD=YES
/FILE ADAyyyyy.DATA ,LINK=DDDATAR1,SHARUPD=YES
/FILE ADAyyyyy.RLOGR1 ,LINK=DDRLOGR1,SHARUPD=YES
/FILE ADAyyyyy.FILEA ,LINK=DDFILEA ,SPACE=(4800,480)
/EXEC (ADARUN,ADA.MOD)
ADARUN PROG=ADAORD,DB=yyyyy,IDTNAME=ADABAS5B
ADAORD STORE
ADAORD FILE=1,DSSIZE=80,DATAPFAC=30,DSRABN=1234
ADAORD MAXISN=200000
/LOGOFF NOSPOOL
```

OS/390 or z/OS

Dataset	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 Manual*
ADAORD parameters	DDKARTE	reader	
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADAORD messages	DDDRUCK	printer	*Messages and Codes*

ADAORD JCL Examples (OS/390 or z/OS)

Reorder File Associator

```
//ADAORDA   JOB
//*
//*     ADAORD : REORDER FILE ASSO,
//*              REORDER ASSO
//*
//ORD      EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.Vvrs.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 MVSJOBS dataset 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.Vvrs.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 MVSJOBS dataset for this example.

Restructure

```
//ADAORDR   JOB
//*
//*     ADAORD : RESTRUCTURE
//*
//ORD      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
//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 MVSJOBS dataset 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.Vvrs.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 MVSJOBS dataset for this example.

VM/ESA or z/VM

Dataset	DD Name	Storage	More Information
Associator	DDASSORn	disk	
Data Storage	DDDATARn	disk	
Intermediate storage	DDFILEA	tape/disk	
Recovery log (RLOG)	DDRLOGR1	disk	Required when using the ADARAI option
ADARUN parameters	DDCARD	disk/terminal/ reader	*Operations Manual*
ADAORD parameters	DDKARTE	disk/terminal/ reader	
ADARUN messages	DDPRINT	disk/terminal/ printer	*Messages and Codes*
ADAORD messages	DDDRUCK	disk/terminal/ printer	*Messages and Codes*

ADAORD JCL Examples (VM/ESA or z/VM)

Reorder Associator

```
DATADEF DDDATAR1,DSN=ADABASVv.DATA,VOL=DATAV1
DATADEF DDFILEA,DSN=ADAORD.FILEA,MODE=A
DATADEF DDPRINT,DSN=ADAORD.DDPRINT,MODE=A
DATADEF DUMP,DUMMY
DATADEF DDDRUCK,DSN=ADAORD.DDDRUCK,MODE=A
DATADEF DDCARD,DSN=RUNORD.CONTROL,MODE=A
DATADEF DDKARTE,DSN=REORASSO.CONTROL,MODE=A
ADARUN
```

Contents of RUNORD CONTROL A1

```
ADARUN PROG=ADAORD,DEVICE=dddd,DB=yyyyy
```

Contents of REORASSO CONTROL A1

```
ADAORD REORFASSO
ADAORD    FILE=1,MAXISN=20000,ASSOPFAC=20
ADAORD    FILE=3,NISIZE=400B,ASSOPFAC=20
```

Reorder Data Storage

```
DATADEF DDASSOR1,DSN=ADABASVv.ASSO,VOL=ASSOV1
DATADEF DDDATAR1,DSN=ADABASVv.DATA,VOL=DATAV1
DATADEF DDFILEA,DSN=ADAORD.FILEA,MODE=A
DATADEF DDPRINT,DSN=ADAORD.DDPRINT,MODE=A
DATADEF DUMP,DUMMY
DATADEF DDDRUCK,DSN=ADAORD.DDDRUCK,MODE=A
DATADEF DDCARD,DSN=RUNORD.CONTROL,MODE=A
DATADEF DDKARTE,DSN=REORDATA.CONTROL,MODE=A
ADARUN
```

Contents of RUNORD CONTROL A1

```
ADARUN PROG=ADAORD,DEVICE=dddd,DB=yyyyy
```

Contents of REORDATA CONTROL A1

```
ADAORD REORDATA
ADAORD    FILE=1,DSSIZE=80,DATAPFAC=30
```

Restructure

```
DATADEF DDASSOR1,DSN=ADABASVv.ASSO,VOL=ASSOV1
DATADEF DDFILEA,DSN=ADAORD.FILEA,MODE=A
DATADEF DDRLOGR1,DSN=ADABVv.RLOGR1,VOL=RLOGV1
DATADEF DDPRINT,DSN=ADAORD.DDPRINT,MODE=A
DATADEF DUMP,DUMMY
DATADEF DDDRUCK,DSN=ADAORD.DDDRUCK,MODE=A
DATADEF DDCARD,DSN=RUNORD.CONTROL,MODE=A
DATADEF DDKARTE,DSN=RESTRUCT.CONTROL,MODE=A
ADARUN
```

Contents of RUNORD CONTROL A1

```
ADARUN PROG=ADAORD,DEVICE=dddd,DB=yyyyy
```

Contents of RESTRUCT CONTROL A1

```
ADAORD RESTRUCTREF
ADAORD    FILE=1,DATADEV=eeee
```

Store

```
DATADEF DDASSOR1,DSN=ADABASVv.ASSO,VOL=ASSOV1
DATADEF DDDATAR1,DSN=ADABASVv.DATA,VOL=DATAV1
DATADEF DDRLOGR1,DSN=ADABVv.RLOGR1,VOL=RLOGV1
DATADEF DDFILEA,DSN=ADAORD.FILEA,MODE=A
DATADEF DDPRINT,DSN=ADAORD.DDPRINT,MODE=A
DATADEF DUMP,DUMMY
DATADEF DDDRUCK,DSN=ADAORD.DDDRUCK,MODE=A
DATADEF DDCARD,DSN=RUNORD.CONTROL,MODE=A
DATADEF DDKARTE,DSN=ORDSTORE.CONTROL,MODE=A
ADARUN
```

Contents of RUNORD CONTROL A1

```
ADARUN PROG=ADAORD,DEVICE=dddd,DB=yyyyy
```

Contents of ORDSTORE CONTROL A1

```
ADAORD STORE
ADAORD    FILE=1,DSSIZE=80,DATAPFAC=30,DSRABN=1234
ADAORD    MAXISN=200000
```

VSE/ESA

File	Symbolic Name	Storage	Logical Unit	More Information
Associator	ASSORn	disk	*	
Data Storage	DATARn	disk	*	
Intermediate Storage	FILEA	tape disk	SYS010 *	
Recovery log (RLOG)	RLOGR1	disk		Required when using the ADARAI option
ADARUN parameters	— CARD CARD	reader tape disk	SYSRDR SYS000 *	
ADAORD parameters		reader	SYSIPT	
ADARUN messages		printer	SYSLST	*Messages and Codes*
ADAORD messages		printer	SYS009	*Messages and Codes*

* Any programmer logical unit may be used.

ADAORD JCS Examples (VSE/ESA)

See appendix B for a description of the VSE/ESA procedures (PROCs).

Refer to the following members for these examples:

Example	Member
Reorder File Associator	ADAORDA.X
Reorder File Data Storage	ADAORDD.X
Restructure	ADAORDR.X
Store Files	ADAORDS.X

Reorder File Associator

```
* $$ JOB JNM=ADAORDA,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADAORDA
*       REORDER THE ASSOCIATOR.
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// ASSGN SYS010,DISK,VOL=DISK01,SHR
// DLBL FILEA,'ADABAS.Vvr.TEMP'
// EXTENT SYS010,DISK01,1,0,sssss,nnnnn
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADAORD,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADAORD REORFASSO
ADAORD FILE=1,ASSOPFAC=15,MAXISN=10000
/*
/&
* $$ EOJ
```

Reorder File Data Storage, Reorder File, Reorder Data, Reorder Database

```
* $$ JOB JNM=ADAORDD,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADAORDD
*       REORDER DATA STORAGE
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// ASSGN SYS010,TAPE,D0
// PAUSE MOUNT SCRATCH TAPE ON TAPE cuu
// MTC REW,SYS010
// MTC WTM,SYS010,5
// MTC REW,SYS010
// TLBL FILEA,'ADABAS.Vvr.TEMP'
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADAORD,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADAORD REORDATA
ADAORD FILE=1,DSSIZE=80,DATAPFAC=30
/*
/&
* $$ EOJ
```

Restructure

```
* $$ JOB JNM=ADAORDR,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADAORDR
*       RESTRUCTURE
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// ASSGN SYS010,TAPE,D0
// PAUSE MOUNT SCRATCH TAPE ON TAPE cuu
// MTC REW,SYS010
// MTC WTM,SYS010,5
// MTC REW,SYS010
// TLBL FILEA,'ADABAS.Vvr.TEMP'
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADAORD,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADAORD RESTRUCTUREF
ADAORD FILE=1,DATADEV=eeee
/*
/&
* $$ EOJ
```

Store Files

```
* $$ JOB JNM=ADAORDS,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADAORDS
*       STORE
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// ASSGN SYS010,TAPE,D0
// PAUSE MOUNT SCRATCH TAPE ON TAPE cuu
// MTC REW,SYS010
// MTC WTM,SYS010,5
// MTC REW,SYS010
// TLBL FILEA,'ADABAS.Vvr.TEMP'
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADAORD,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADAORD STORE
ADAORD FILE=1,DSSIZE=80,DATAPFAC=30,DSRABN=1234,MAXISN=200000
/*
/&
* $$ EOJ
```

ADAPLP : PROTECTION LOG/WORK PRINT

The ADAPLP utility prints data protection records contained on the Adabas Work dataset or the Adabas data protection log.

Print Protection Data

The following diagram shows the ADAPLP syntax for specifying sequential intermediate (IPLOGPRI), multiple (PLOG..) or sequential (SPLOG..) protection logs, or Work printing:

```
ADAPLP {IPLOGPRI | PLOGPRI | SPLOGPRI | WORKPRI }
                [ {TYPE=type | FILE=file-number [ISN=isn] | RABN=data-storage-rabn} ]
                [DEVICE={multiple-PLOG-device-type|ADARUN-device } ]
                [LAYOUT={1 | 2 | 3} ]
                [NOUSERABEND]
                [NUMBER={1 | n} ]
                [PRINT]
                [SKIPRABN={block-count | 0} ]
                [STOPRABN=block-count ]
```

—where “type” is one of the following:

```
[ {ALL | ASSO | DATA} [,FILE=file-number] [ISN=isn]
| {ALL | DATA} [RABN=data-storage-rabn]
| {C1 | C5 | ET | EEKZ | SAVO | VEKZ}
| REPR [,FILE=file-number] ]
```

The IPLOGPRI function is used to print the sequential intermediate datasets created from the PLOG merge process. Input to ADAPLP IPLOGPRI must be a MERGINTI/O dataset 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 dataset 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)

```
EBCDIC              HEXADECIMAL                                OFFSET
  ↓                   ↓                                          ↓

*      I   1   SYN*   00250000000BC9059C4DF1CD05E2E8D5          0000
*5A     ADA003    *   F5C10204020307C1C4C1F0F0F3020206          0010
*GAOS1   /       *   C7C1F0E2F1006200610000000300044O          0020
```

LAYOUT=2/3

```
OFFSET  HEXADECIMAL                          EBCDIC
↓       ↓                                    ↓

0000    00250000000BC9059C4DF1CD05E2E8D5     *      I   1   SYN*
0010    F5C10204020307C1C4C1F0F0F3020206     *5A     ADA003    *
0020    C7C1F0E2F1006200610000000300044O     *GAOS1   /        *
```

LAYOUT=3 presents the same format as LAYOUT=2, and also includes an explanation of each PLOG record type.

NOUSERABEND : Termination without ABEND

When an error is encountered while the function is running, the utility 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.

NUMBER : Multiple Protection Log Dataset Number

NUMBER specifies the one of multiple (two through eight) protection log (PLOG) datasets to be printed. Specifying “2” selects the DD/PLOGR2 dataset; specifying “3” selects the DD/PLOGR3 dataset, etc. The default of “1” selects DD/PLOGR1.

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 be specified for TYPE=ALL (the default) or TYPE=DATA functions; that is, those that select data storage protection records.

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
EEKZ	records written at completion of a nucleus buffer flush
ET	records resulting from Adabas ET commands
REPR	Work dataset records used by autorestart to repair the index
SAVO	online SAVE database/file records
VEKZ	records written at completion of update commands

Note:
The number of protection records is reduced further by specifying the FILE, ISN, or RABN parameters.

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 datasets are to be printed.

JCL/JCS Requirements and Examples

This section describes the job control information required to run ADAPLP with BS2000, OS/390 or z/OS, VM/ESA or z/VM, and VSE/ESA systems and shows examples of each of the job streams.

BS2000

Dataset	Link Name	Storage	More Information
Associator	DDASSORn	disk	Required for WORKPRI.
Work	DDWORKR1	disk	Required for WORKPRI.
Sequential protection log	DDPLOG	tape/disk	Required for SPLOGPRI.

Dataset	Link Name	Storage	More Information
Multiple protection log	DDPLOGR1	disk	Required for PLOGPRI if NUMBER=1 (the default).
Multiple protection log	DDPLOGRn	disk	Required for PLOGPRI if NUMBER=n.
ADARUN parameters	SYSDTA/ DDCARD		*Operations Manual*
ADAPLP parameters	SYSDTA/ DDKARTE		
ADARUN messages	SYSOUT DDPRINT		*Messages and Codes*
ADAPLP messages	SYSLST DDDRUCK		*Messages and Codes*

ADAPLP JCL Examples (BS2000)

Print Sequential Protection Log

In SDF Format:

```
/.ADAPLP LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK * A D A P L P PRINT SEQUENTIAL PROTECTION LOG
/REMARK *
/ASS-SYSLST L.PLP.SPLO
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDPLOG,ADAyyyyy.PLOG
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADAPLP,DB=yyyyy,IDTNAME=ADABAS5B
ADAPLP SPLOGPRI PRINT
/LOGOFF SYS-OUTPUT=DEL
```

In ISP Format:

```
/.ADAPLP LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK * A D A P L P PRINT SEQUENTIAL PROTECTION LOG
/REMARK *
/SYSFILE SYSLST=L.PLP.SPLO
/FILE ADA.MOD,LINK=DDLIB
/FILE ADAyyyyy.PLOG,LINK=DDPLOG
/EXEC (ADARUN,ADA.MOD)
ADARUN PROG=ADAPLP,DB=yyyyy,IDTNAME=ADABAS5B
ADAPLP SPLOGPRI PRINT
/LOGOFF NOSPOOL
```

Print One of Multiple Protection Log Datasets

In SDF Format:

```
/.ADAPLP LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK * A D A P L P PRINT MULTIPLE PROTECTION LOG
/REMARK *
/ASS-SYSLST L.PLP.PLOG
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDPLOGR1,ADAyyyyy.PLOG
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADAPLP,DB=yyyyy,IDTNAME=ADABAS5B
ADAPLP PLOGPRI PRINT
/LOGOFF SYS-OUTPUT=DEL
```

In ISP Format:

```
/.ADAPLP LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK * A D A P L P PRINT MULTIPLE PROTECTION LOG
/REMARK *
/SYSFILE SYSLST=L.PLP.PLOG
/FILE ADA.MOD,LINK=DDLIB
/FILE ADAyyyyy.PLOG,LINK=DDPLOGR1
/EXEC (ADARUN,ADA.MOD)
ADARUN PROG=ADAPLP,DB=yyyyy,IDTNAME=ADABAS5B
ADAPLP PLOGPRI PRINT
/LOGOFF NOSPOOL
```

Print Work

In SDF Format:

```
/.ADAPLP LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK * A D A P L P PRINT ADABAS WORK
/REMARK *
/ASS-SYSLST L.PLP.WORK
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDASSOR1,ADAyyyyy.ASSO,SHARE-UPD=YES
/SET-FILE-LINK DDWORKR1,ADAyyyyy.WORK,SHARE-UPD=YES
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADAPLP,DB=yyyyy,IDTNAME=ADABAS5B
ADAPLP WORKPRI PRINT,TYPE=ASSO
/LOGOFF SYS-OUTPUT=DEL
```

In ISP Format:

```
/.ADAPLP LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK * A D A P L P PRINT ADABAS WORK
/REMARK *
/SYSFILE SYSLST=L.PLP.WORK
/FILE ADA.MOD,LINK=DDLIB
/FILE ADAyyyyy.ASSO ,LINK=DDASSOR1,SHARUPD=YES
/FILE ADAyyyyy.WORK ,LINK=DDWORKR1,SHARUPD=YES
/EXEC (ADARUN,ADA.MOD)
ADARUN PROG=ADAPLP,DB=yyyyy,IDTNAME=ADABAS5B
ADAPLP WORKPRI PRINT,TYPE=ASSO
/LOGOFF NOSPOOL
```

OS/390 or z/OS

Dataset	DD Name	Storage	More Information
Associator	DDASSORn	disk	Required for WORKPRI.
Work	DDWORKR1	disk	Required for WORKPRI.
Sequential protection log	DDPLOG	tape/disk	Required for SPLOGPRI.
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 Manual*
ADAPLP parameters	DDKARTE	reader	

ADAPLP JCL Examples (OS/390 or z/OS)

Refer to the MVSJOBS dataset for the following example jobs:

Job Member	Description
ADAPLP	Print protection log (from multiple dataset 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 Datasets

```
//ADAPLP    JOB
//*
//*   ADAPLP: PROTECTION LOG PRINT (FROM MULTIPLE PLOG)
//*
//PLP       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
//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.Vvrs.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 Adabas Work

```
//ADAPLPW    JOB
//*
//*   ADAPLP: PRINT ADABAS WORK
//*
//PLP       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
//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
/*
//
```

VM/ESA or z/VM

Dataset	DD Name	Storage	More Information
Associator	DDASSORn	disk	Required for WORKPRI.
Work	DDWORKR1	disk	Required for WORKPRI.
Sequential protection log	DDPLOG	tape/disk	Required for SPLOGPRI.
Multiple protection log	DDPLOGR1	disk	Required for PLOGPRI if NUMBER=1 (default).
Multiple protection log	DDPLOGRn	disk	Required for PLOGPRI if NUMBER=n.
ADARUN parameters	DDCARD	disk/terminal/reader	*Operations Manual*
ADAPLP parameters	DDKARTE	disk/terminal/reader	
ADARUN messages	DDPRINT	disk/terminal/printer	*Messages and Codes*
ADAPLP messages	DDDRUCK	disk/terminal/printer	*Messages and Codes*

ADAPLP Examples (VM/ESA or z/VM)

Print Sequential Protection Log

```
DATADEF DDPLOG,DSN=ADABASVv.SIBA,MODE=A
DATADEF DDPRINT,DSN=ADAPLP.DDPRINT,MODE=A
DATADEF DUMP,DUMMY
DATADEF DDDRUCK,DSN=ADAPLP.DDDRUCK,MODE=A
DATADEF DDCARD,DSN=RUNPLP.CONTROL,MODE=A
DATADEF DDKARTE,DSN=PLPSPLOG.CONTROL,MODE=A
ADARUN
```

Contents of RUNPLP CONTROL A1:

```
ADARUN PROG=ADAPLP,DEVICE=dddd,DB=yyyyy
```

Contents of PLPSPLOG CONTROL A1:

```
ADAPLP SPLOGPRI PRINT
```

Print One of Multiple Protection Log Datasets

```
DATADEF DDPLOG,DSN=ADABASVv.PLOG2,VOL=PLOGV2
DATADEF DDPRINT,DSN=ADAPLP.DDPRINT,MODE=A
DATADEF DUMP,DUMMY
DATADEF DDPLOGR1,DSN=ADAPLP.DDPLOG,MODE=A
DATADEF DDDRUCK,DSN=ADAPLP.DDDRUCK,MODE=A
DATADEF DDCARD,DSN=RUNPLP.CONTROL,MODE=A
DATADEF DDKARTE,DSN=PLPDPLOG.CONTROL,MODE=A
ADARUN
```

Contents of RUNPLP CONTROL A1:

```
ADARUN PROG=ADAPLP,DEVICE=dddd,DB=yyyyy
```

Contents of PLPDPLOG CONTROL A1:

```
ADAPLP PLOGPRI PRINT
```

Print Work

```
DATADEF DDASSOR1,DSN=ADABASVv.ASSO,VOL=ASSOV1
DATADEF DDWORKR1,DSN=ADABASVv.WORK,VOL=WORKV1
DATADEF DDPRINT,DSN=ADAPLP.DDPRINT,MODE=A
DATADEF DUMP,DUMMY
DATADEF DDDRUCK,DSN=ADAPLP.DDDRUCK,MODE=A
DATADEF DDCARD,DSN=RUNPLP.CONTROL,MODE=A
DATADEF DDKARTE,DSN=PLPWORK.CONTROL,MODE=A
ADARUN
```

Contents of RUNPLP CONTROL A1:

```
ADARUN PROG=ADAPLP,DEVICE=dddd,DB=yyyyy
```

Contents of PLPWORK CONTROL A1:

```
ADAPLP WORKPRI PRINT,TYPE=ASSO
```

VSE/ESA

File	Symbolic Name	Storage	Logical Unit	More Information
Associator	ASSORn	disk	*	Required for WORKPRI.
Work	WORKR1	disk	*	Required for WORKPRI.
Sequential protection log	PLOG	tape disk	SYS014 *	Required for SPLOGPRI.
Multiple protection log	PLOGR1	disk	*	Required for PLOGPRI if NUMBER=1 (default).
Multiple protection log	PLOGRn	disk	*	Required for PLOGPRI if NUMBER=n.
ADAPLP report		printer	SYS009	
ADARUN messages		printer	SYSLST	
ADARUN parameters	SYSRDR CARD	reader/ tape/disk		*Operations Manual*
ADAPLP parameters	SYSIPT	reader		

***** Any programmer logical unit may be used.

ADAPLP JCS Examples (VSE/ESA)

See appendix B for a description of the VSE/ESA procedures (PROCs).

Refer to the following members for these examples:

Example	Member
Print sequential protection log	ADAPLPS.X
Print multiple protection log	ADAPLP.X
Print Adabas Work	ADAPLPW.X

Print Sequential Protection Log

```
* $$ JOB JNM=ADAPLPS,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADAPLPS
*       PROTECTION LOG PRINT (FROM SEQUENTIAL PLOG)
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// ASSGN SYS014,cuu
// PAUSE MOUNT LOAD INPUT FILE ON TAPE cuu
// TLBL PLOG,'EXAMPLE.DByyyyy.PLOG'
// MTC REW,SYS014
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADAPLP,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADAPLP SPLOGPRI
/*
/&
* $$ EOJ
```

Print One of Multiple Protection Log Datasets

```
* $$ JOB JNM=ADAPLP,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADAPLP
*       PROTECTION LOG PRINT (FROM MULTIPLE PLOG)
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADAPLP,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADAPLP PLOGPRI
/*
/&
* $$ EOJ
```

Print Adabas Work

```
* $$ JOB JNM=ADAPLPW,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADAPLPW
*       PRINT ADABAS WORK
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADAPLP,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADAPLP WORKPRI PRINT
/*
/&
* $$ EOJ
```

ADAPRI : PRINT SELECTED ADABAS BLOCKS

The ADAPRI utility prints the contents of a block (or range of blocks) contained in the Associator (ASSO..), Data Storage (DATA..), Work (WORK..), temp (TEMP..), sort (SORT..), multiple dataset command log (CLOG), multiple dataset protection log (PLOG), or the recovery log (RLOG)) dataset. More than one dataset may be printed during a single ADAPRI execution.

Print Blocks

```
ADAPRI   {ASSOPRI | DATAPRI | WORKPRI | TEMPPRI | SORTPRI | RLOGPRI |
              CLOGPRI [NUMBER={1 | n} ] | PLOGPRI [NUMBER={1 |n} ]  }
         FROMRABN=block-number
         TORABN=block-number
         [BATCH]
         [DEVICE={device-type | ADARUN-device} ]
         [NOUSERABEND]
```

Essential Parameters

FROMRABN / TORABN : Range of Blocks to Be Printed

The beginning and ending numbers of the RABNs to be printed. Both values must be specified; there are no defaults.

Printing begins with the block number specified with the FROMRABN parameter and ends with the block number specified with the TORABN parameter. Each block in the range is printed in hexadecimal format.

Optional Parameters

BATCH : Output Format

Controls the line length of the printed output. If BATCH is not specified, the default line size is 80 characters. If BATCH is specified, the output line size is 120 characters.

DEVICE : Device Type

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

NOUSERABEND : Termination without ABEND

When an error is encountered while the function is running, the utility 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.

NUMBER : Command / Protection Log Dataset Number

The number of the multiple (two through eight) command log (CLOG) or protection log (PLOG) dataset from which the blocks are to be printed. NUMBER can only be specified when CLOGPRI or PLOGPRI is specified. When NUMBER=2 is specified, DD/CLOGR2 blocks are printed; if the CLOGPRI or PLOGPRI function is specified without NUMBER, the blocks are taken from DD/CLOGR1 (the default).

Examples

Example 1:

ADAPRI ASSOPRI FROMRABN=1,TORABN=1

Block 1 of the Associator (which contains the general control block) is printed.

Example 2:

ADAPRI DATAPRI FROMRABN=8000,TORABN=8120

Blocks 8000 to 8120, inclusively, of Data Storage are printed.

Example 3:

ADAPRI WORKPRI FROMRABN=1,TORABN=100,BATCH

Blocks 1 to 100 of the Adabas Work are to be printed. The output line size to be used is 120.

Example 4:

ADAPRI CLOGPRI FROMRABN=1,TORABN=80,BATCH,NUMBER=2

Blocks 1 to 100 of the command log dataset DD/CLOGR2 are printed in 120-character-wide format.

Example 5:

ADAPRI DSIMPRI FROMRABN=1,TORABN=1

Block 1 only of the DSIM dataset is printed. The DSIM dataset is only used if Adabas Delta Save Facility is installed.

JCL/JCS Requirements and Examples

This section describes the job control information required to run ADAPRI with BS2000, OS/390 or z/OS, VM/ESA or z/VM, and VSE/ESA systems and shows examples of each of the job streams.

BS2000

Dataset	Link Name	Storage	More Information
Dataset containing blocks to print	DDASSORn	disk	Associator
	DDDATARn	disk	Data Storage*
	DDWORKR1	disk	Work*
	DDTEMPR1	disk	Temp
	DDSORTRn	disk	Sort
	DDCLOGRn	disk	Multiple command log
	DDPLOGRn	disk	Multiple protection log
	DDDSIMR1	disk	DSIM dataset
Recovery log (RLOG)	DDRLOGR1	disk	Required when using ADARAI.
ADARUN parameters	SYSDTA/ DDCARD	reader	*Operations Manual*
ADAPRI parameters	SYSIPT/ DDKARTE	reader	
ADARUN messages	SYSOUT	printer	*Messages and Codes*
ADAPRI messages	SYSLST	printer	*Messages and Codes*

* When printing blocks from Data Storage or Work, the link name for the Associator must also be present.

ADAPRI JCL Example (BS2000)

In SDF Format:

```
/.ADAPRI LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK * A D A P R I ALL FUNCTIONS
/REMARK *
/ASS-SYSLST L.PRI
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDASSOR1,ADAyyyyy.ASSO,SHARE-UPD=YES
/SET-FILE-LINK DDDATAR1,ADAyyyyy.DATA,SHARE-UPD=YES
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADAPRI,DB=yyyyy,IDTNAME=ADABAS5B
ADAPRI DATAPRI FROMRABN=27,TORABN=34
/LOGOFF SYS-OUTPUT=DEL
```

In ISP Format:

```
/.ADAPRI LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK * A D A P R I ALL FUNCTIONS
/REMARK *
/SYSFILE SYSLST=L.PRI
/FILE ADA.MOD,LINK=DDLIB
/FILE ADAyyyyy.ASSO ,LINK=DDASSOR1,SHARUPD=YES
/FILE ADAyyyyy.DATA ,LINK=DDDATAR1,SHARUPD=YES
/EXEC (ADARUN,ADA.MOD)
ADARUN PROG=ADAPRI,DB=yyyyy,IDTNAME=ADABAS5B
ADAPRI DATAPRI FROMRABN=27,TORABN=34
/LOGOFF NOSPOOL
```

OS/390 or z/OS

Dataset	DD Name	Storage	More Information
Dataset containing the blocks print	DDASSORn	disk	Associator
	DDDATARn	disk	Data Storage*
	DDWORKR1	disk	Work*
	DDTEMPR1	disk	Temp
	DDSORTR1	disk	Sort
	DDCLOGRn	disk	Multiple command log
	DDPLOGRn	disk	Multiple protection log
	DDDSIMR1	disk	DSIM dataset
Recovery log (RLOG)	DDRLOGR1	disk	Required when using ADARAI.
ADARUN parameters	DDCARD	reader	*Operations Manual*
ADAPRI parameters	DDKARTE	reader	
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADAPRI messages	DDDRUCK	printer	*Messages and Codes*

***** When printing blocks from Data Storage or Work, the DD statement for the Associator must also be present.

ADAPRI JCL Example (OS/390 or z/OS)

```
//ADAPRI    JOB
//*
//*   ADAPRI:
//*      MAINTENANCE PRINT
//*
//PRI      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
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <=== WORK
//DDTEMPR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.TEMPR1 <=== TEMP
//DDSORTR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.SORTR1 <=== SORT
//DDPLOGR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.PLOGR1 <=== PLOG 1
//DDPLOGR2  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.PLOGR2 <=== PLOG 2
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADAPRI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
//DDKARTE   DD   *
ADAPRI ASSOPRI DEVICE=eeee,FROMRABN=1,TORABN=1
/*
//
```

Refer to ADAPRI in the MVSJOBS dataset for this example.

VM/ESA or z/VM

Dataset	DD Name	Storage	More Information
Dataset containing the blocks to print	DDASSORn	disk	Associator
	DDDATARn	disk	Data Storage*
	DDWORKR1	disk	Work*
	DDTEMPR1	disk	Temp
	DDSORTR1	disk	Sort
	DDCLOGRn	disk	Multiple command log
	DDPLOGRn	disk	Multiple protection log
	DDDSIMR1	disk	DSIM dataset
Recovery log (RLOG)	DDRLOGR1	disk	Required when using ADARAI.
ADARUN parameters	DDCARD	disk/terminal/ reader	*Operations Manual*
ADAPRI parameters	DDKARTE	disk/terminal/ reader	
ADARUN messages	DDPRINT	disk/terminal/ printer	*Messages and Codes*
ADAPRI messages	DDDRUCK	disk/terminal/ printer	*Messages and Codes*

***** When printing blocks from Data Storage or Work, the DD statement for the Associator must also be present.

ADAPRI Example (VM/ESA or z/VM)

```
DATADEF DDASSOR1,DSN=ADABASVv.ASSO,VOL=ASSOV1
DATADEF DDDATAR1,DSN=ADABASVv.DATA,VOL=DATAV1
DATADEF DDPRINT,DSN=ADAPRI.DDPRINT,MODE=A
DATADEF DUMP,DUMMY
DATADEF DDDRUCK,DSN=ADAPRI.DDDRUCK,MODE=A
DATADEF DDCARD,DSN=RUNPRI.CONTROL,MODE=A
DATADEF DDKARTE,DSN=ADAPRI.CONTROL,MODE=A
ADARUN
```

Contents of RUNPRI CONTROL A1:

```
ADARUN PROG=ADAPRI,DEVICE=dddd,DB=yyyyy
```

Contents of ADAPRI CONTROL A1:

```
ADAPRI DATAPRI FROMRABN=27,TORABN=34
```

VSE/ESA

File	Symbolic Name	Storage	More Information
Files containing the blocks to be printed	ASSORn	disk	Associator
	DATARn	disk	Data Storage*
	WORKR1	disk	Work*
	TEMPR1	disk	Temp
	SORTR1	disk	Sort
	CLOGRn	disk	Multiple command log
	PLOGRn	disk	Multiple protection log
	SIMR1	disk	DSIM dataset
Recovery log (RLOG)	RLOGR1	disk	Required when using ADARAI
ADARUN messages	SYSLST	printer	*Messages and Codes*
ADAPRI messages	SYS009	printer	*Messages and Codes*
ADARUN parameters	SYSRDR	reader	*Operations Manual*
	CARD	tape/disk	
ADAPRI parameters	SYSIPT	reader	

* When printing blocks from Data Storage or Work, the JCS statement for the Associator must also be present.

ADAPRI JCS Example (VSE/ESA)

See appendix B for descriptions of the VSE/ESA procedures (PROCs).

```
* $$ JOB JNM=ADAPRI,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADAPRI
*       MAINTENANCE PRINT
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADAPRI,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADAPRI ASSOPRI DEVICE=eeee,FROMRABN=1,TORABN=1
/*
/&
* $$ EOJ
```

Refer to member ADAPRI.X for this example.

16

ADARAI : ADABAS RECOVERY AID

"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 Manual*.

However, when a database dataset (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/JCS), which is the basis for reexecuting the jobs performed from the time of the last SAVE to the point of failure and error.

Note:
The job stream generation function is not yet available under VSE/ESA or VM/ESA.

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.

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 datasets, utility parameters, and protection logs needed to build the recovery job control. The RLOG dataset is DD/RLOGR1.

In a nucleus cluster environment, all nuclei use the same RLOG. Concurrent updates to the RLOG are controlled by a lock.

Notes:

1. *Sequential datasets used by the utilities whose runs are logged on the RLOG must be kept and available for any recovery operation; for example, the DD/EBAND input to an ADALOD LOAD operation.*
2. *ADADBS file changes are now recorded on the RLOG dataset.*
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, and/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 labelled “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 labelled “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 dataset is decreased in size, the user must create a Work dataset with the original size so that the recovery job can run correctly up to the point where the Work dataset size was decreased.
- A generation is labelled “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, Software AG recommends that you start a new generation as soon as possible by performing an on- or off-line save of the database. If the 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.

Function Overview

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.

The following functions are available for use with ADARAI:

Function	Action	Page
CHKDB	check the database status	120
DISABLE	disable the recovery log (RLOG)	121
LIST	list the information contained in the recovery log (RLOG)	122
PREPARE	prepare and start the recovery log file (RLOG)	135
RECOVER	create (rebuild) the job control stream to recover the database and check the database status	138
REMOVE	remove the RLOG	157

CHKDB: Check the Database Status

ADARAI CHKDB [{ **ACTIVE**|**INACTIVE** }]

The ADARAI CHKDB function checks for the specified recovery nucleus status (active—the default, or inactive) by issuing a command to the nucleus and testing 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.

DISABLE : Deactivate Recovery Logging

ADARAI DISABLE

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 page 118.

Example:

ADARAI DISABLE

Deactivates all Recovery Aid logging.

LIST : Display Current RLOG Generations

Note:
Adabas version 6 RLOGs cannot be listed; only version 7 and above RLOGs are supported.

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 DD/CARD and DD/KARTE statements; and
- details of any files written or read during the logged event.

In a nucleus cluster environment, the PLOG datasets 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
```

Additional LIST Information on BS2000

On BS2000 systems, LIST also provides the following information:

- file or file generation group (FGG) characteristics and physical location (tape, disk, etc.);
- existence and condition of each dataset (written and erased; overwritten; written as a temporary file) needed by the ADARAI RECOVER function; and
- error or warning message for incompatible coded file ID (CFID) comparison.

Since nonmatching coded file IDs (CFIDs) are a reliable indication of overwritten datasets in BS2000, LIST compares CFIDs to determine whether any datasets have been overwritten. For lost or overwritten datasets, LIST provides an error or warning indication for the following conditions:

- An “ERROR” is indicated when the dataset was written to
  - disk, and was then erased;
  - disk, and then was overwritten (CFIDs do not match). Both the original and the catalog entry are reported;
  - a temporary disk dataset.
- A “WARNING” is indicated when the dataset was written to
  - tape, and the catalog entry has been erased;
  - tape, and then was overwritten (CFIDs do not match). Both the original and the catalog entry are reported;
  - a temporary tape dataset.

The section **Output Examples** starting on page 126 provides examples of the operating-system-dependent results provided by the LIST function.

Syntax

```
ADARAI LIST   [GENS={NO|YES}]
              [RELGEN={gen-number|gen-number – gen-number}]
              [RLOGDEV={device|ADARUN-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, ADRAI 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

BS2000

The following example shows LIST output for a single BS2000 disk dataset:

```
LINK=DDSAVE1 PATHNAME=:A:$GEB.RAI.vv.SAVE.012
SIZE=6387 SEC-ALLO=96 LPP=6336
FCBTYPE=SAM RECFORM=V BLKSIZE=(STD,16) RECSIZE=32748 BLKCNTRL=PAMKEY
VSN/DEV PUBA00/D3480  /AC PUBA01/D3480  /AC
VSN/DEV PUBA02/D3480  /AC
```

The following example shows LIST output for a BS2000 file-generation group (FGG):

```
FGG INDEX BASE=10 CURRENT=10 FIRST=1 MAX=255 DISP=DEL
LINK=DDSAVE1 PATHNAME=:A:$GEB.RAI.vv.SAVE.TAPE.01(*0010)
TAPE  DEVICE=TAPE=C1 (B5) FSEQ=1 BLKCOUNT=4000
FCBTYPE=SAM RECFORM=V BLKSIZE=32760 RECSIZE=32756 BLKCNTRL=NO
VOLUMES GEBR11 GEBR12 GEBR13 GEBR17 GEBR19
```

The following examples are of ADARAI LIST error/warning output. The first is for a lost disk dataset, and the second for an overwritten tape dataset:

```
LINK=DDSAVE1 DISC DATASET NOT PRESENT - E R R O R
ORIGINAL CATALOG ENTRY:
LINK=DDSAVE1 PATHNAME=:A:$GEB.RAI.vv.SAVE.012
SIZE=6387 SEC-ALLO=96 LPP=6336
FCBTYPE=SAM RECFORM=V BLKSIZE=(STD,16) RECSIZE=32748 BLKCNTRL=PAMKEY
VSN/DEV PUBA00/D3480  /AC PUBA01/D3480  /AC
VSN/DEV PUBA02/D3480  /AC
```

```
LINK=DDSAVE1 CFID MISMATCH - W A R N I N G
ORIGINAL 379949EE  NOW : 379972F0
ORIGINAL CATALOG ENTRY:
LINK=DDSAVE1 PATHNAME=:A:$GEB.RAI.vv.SAVE.012
TAPE  DEVICE=TAPE=C1 (B5) FSEQ=1 BLKCOUNT=4000
FCBTYPE=SAM RECFORM=V BLKSIZE=32760 RECSIZE=32756 BLKCNTRL=NO
CREATION DATE yyyy-mm-dd 11:44:35
VOLUMES GEBR11 GEBR12 GEBR13 GEBR17 GEBR19
ACTUAL CATALOG ENTRY:
LINK=DDSAVE1 PATHNAME=:A:$GEB.RAI.vv.SAVE.012
TAPE  DEVICE=TAPE=C1 (B5) FSEQ=1 BLKCOUNT=3900
FCBTYPE=SAM RECFORM=V BLKSIZE=32760 RECSIZE=32756 BLKCNTRL=NO
CREATION DATE yyyy-mm-dd 12:34:56
VOLUMES GEBR23 GEBR65 GEBR66 GEBR67 GEBR68
```

OS/390 or z/OS

```
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.Vvrs.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.Vvrs.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
```

VSE/ESA

```
A D A R A I   Vv.v  SMv   DBID = 00059  STARTED     yyyy-mm-dd  hh:mm:ss

PARAMETERS:

ADARAI LIST

18:45:04 ADAI64  FILE RLOGR1  HAS BEEN OPENED IN ECKD MODE
18:45:04 ADAI64  FILE RLOGM1  HAS BEEN OPENED IN ECKD MODE

RECOVERY LOG FILE FOR DATABASE 59

START RABN FOR LOG DATA AREA IS     26
HIGHEST LOG AREA RABN IS            633
CURRENT VALUE FOR ROTATING RABN IS  26

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      1 I N I     26       26 I yyyy-08-30  17:06:51   yyyy-08-30  18:44:35 I
I      0 I R I     25       25 I yyyy-08-30  17:01:02   yyyy-08-30  17:05:05 I
I--------I---I-----------------I-------------------------------------------I

*** yyyy-08-30  17:06:51
*** SAVE DATABASE OFFLINE NON INCREMENTAL

SAVE DATASET PLOG NUMBER         = 1966
ADASAV SAVE
// TLBL SAVE1,'PMIG.ADAvrs.SAVE1',0,ADES01

DDSAVE1    VOLSER=XXXXXX    FROM BLOCK=1
                            TO BLOCK  =6192
         VOLUME IS ASSOCIATED TO PLOG NO. 1966
FILE=001,002,003,004,005,006,008,009,010,011,012,013,014,015
FILE=016,017,019,021,022,023,025,027

DDSAVE1    VOLSER=XXXXXX    FROM BLOCK=1
                            TO BLOCK  =31961
         VOLUME IS ASSOCIATED TO PLOG NO. 1966
FILE=001,002,003,004,005,006,008,009,010,011,012,013,014,015
FILE=016,017,019,021,022,023,025,027

*** yyyy-08-30  17:08:12  NUCLEUS PLOG NUMBER=1967
*** START NUCLEUS SESSION

*** yyyy-08-30  17:10:15  NUCLEUS PLOG NUMBER=1967
*** ADABAS UTILITY RUN
```

```
SYNP-CHECKPOINT ID IS 35 (UNLOAD FILE)
SYNP-CHECKPOINT IS FOUND ON PLOG 1967 IN BLOCK NO. 5
FILE=001
ADAULD FILE=1 NUMRECS=100
// DLBL OUT1,'VSESP.SAPLB.ULD2',7,SD
// EXTENT SYS034,SYSWK1,1,0,16365,30

*** yyyy-08-30  17:14:28  NUCLEUS PLOG NUMBER=1967
*** ADABAS UTILITY RUN

SYNP-CHECKPOINT ID IS 35 (UNLOAD FILE)
SYNP-CHECKPOINT IS FOUND ON PLOG 1967 IN BLOCK NO. 8
FILE=001
ADAULD FILE=1 NUMRECS=100
// DLBL OUT1,'VSESP.SAPLB.ULD2',7,SD
// EXTENT SYS034,SYSWK1,1,0,16365,30

*** yyyy-08-30  18:44:35  NUCLEUS PLOG NUMBER=1967
*** ADABAS UTILITY RUN

SYNP-CHECKPOINT ID IS 30 (LOAD FILE)
SYNP-CHECKPOINT IS FOUND ON PLOG 1967 IN BLOCK NO. 12
FILE=004
ADALOD LOAD FILE=4,ISNREUSE=YES,SORTSIZE=5,TEMPSIZE=5,DSSIZE=50B
ADALOD MAXISN=10,NAME='TESTFILE',DSREUSE=YES,LWP=1000000,LIP=500
ADALOD NUMREC=5,NISIZE=5B,UISIZE=5B
// DLBL EBAND,'VSESP.SAPLB.ULD2',7,SD
// EXTENT SYS011,SYSWK1,1,0,16365,30

ADAI03 RLOGR1           3 READS         0 WRITES
ADAI03 RLOGM1           1 READS         0 WRITES

A D A R A I  TERMINATED NORMALLY                        yyyy-08-30  18:45:03
```

PREPARE : Initialize and Start the RLOG

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 dataset 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 Manual* for more information about generation statuses.

Syntax

```
ADARAI PREPARE  RLOGSIZE=size
                [RLOGDEV={device|ADARUN-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 dataset is limited to 16,777,215 (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 dataset 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.

RECOVER : Build a Recovery Job Stream

Note:
The RECOVER function is currently available for BS2000 and OS/390 or z/OS systems only. Support for VM/ESA, z/VM, and VSE/ESA systems is planned.

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 DD/JCLOUT dataset.

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.

For BS2000 systems, RECOVER additionally

- performs, when generating the job control, the same checks performed by the LIST function for BS2000; and
- includes BS2000 /REMARK statements in the created job control for checks that produce errors.

Note:
When such errors occur, the job control must be corrected manually.

Recovery Processing

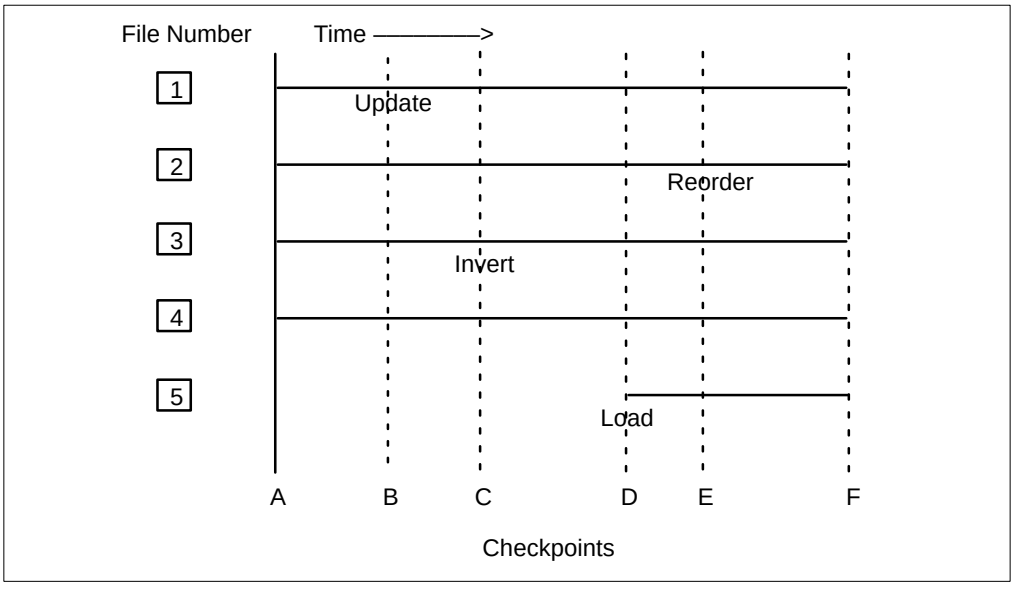
The ADARAI RECOVER function builds a job based on the exact sequence it finds in the generation to be recovered:

- it restores the database from the datasets 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, it 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 datasets input to utility functions that update files in the database must be retained.
- sequential output datasets 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 datasets must keep their original names; ADARAI cannot track copies with different names.

Software AG recommends that retained datasets be cataloged.

Restrictions

Shadow Databases

If “shadow” databases or copies of normal production databases are built by restoring the delta save output and DSIM dataset 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 dataset if a restore operation becomes necessary on the shadow database.

If, however, the DSIM dataset and the delta save dataset are merged to create a new “off-line” delta save dataset and the new merged dataset 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 Dataset

In general, ADARAI handles RESTORE DELTA processing without problems. However, if the RESTORE DELTA uses a DSIM dataset (which is essentially a “working” dataset), the DSIM dataset may not be intact if an ADARAI RECOVER becomes necessary. ADARAI therefore records the COPY or PLCOPY requests used to create a DSIM dataset and emits a job step to rebuild the dataset 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 dataset prior to the RESTORE step and therefore cannot replay the RESTORE DELTA.

DD/FILEA File

In a generated recovery job, ADARAI writes the DD/FILEA file of the ADAORD utility. This cannot be avoided because the REORDER functions must be replayed and they require that the DD/FILEA file be written.

In this case, the following restrictions apply:

- ADAORD STORE processing simply reads the same DD/FILEA 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 DD/FILEA because the recovery job creates and deletes the file again when it is executed.
- An existing file (DISP=OLD) can be used for DD/FILEA. 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 DD/FILEA 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 dataset created by the generation. If the dataset 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 datasets are input to the ADARAI RECOVER function:

- DD/RLOGR1, the recovery log.
- DD/PLOGR1 and DD/PLOGRn, the multiple protection logs, which are required when the ADARAI RECOVER parameter FEOFPL=YES (the default) is used.
- DD/JCLIN, 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.

  On BS2000 systems, DDJCLIN is a SAM dataset with variable record format. EDT can be used to create and edit this dataset. See the section **Skeleton Job Control** on page 151 for more information.

On OS/390 or z/OS systems, the DDJCLIN dataset 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 DD/JCLOUT 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** on page 155.

On BS2000 systems, DDJCLOUT and DDJCLCON are SAM datasets with variable record format. They conform to the BS2000 job control conventions.

On OS/390 or z/OS systems, the DDJCLOUT DD statement must point to a dataset 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** on page 144 and **Skeleton Job Control** on page 151.

Executing the RECOVER Function

The RECOVER function is executed 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 DD/JCLOUT 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 page 118 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 datasets.

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 DD/JCLOUT 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 DD/JCLOUT 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 DD/JCLOUT 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 DD/SAVE input dataset that can be used to create ADASAV RESTORE FILE=... job control.

Syntax

```
ADARAI RECOVER  [AUTOBACKOUT]
                [DRIVES={number|1}]
                [DSIMSIZE=size,DSIMDEV={device | | ADARUN-device}]
                [FEOFPL={NO}]
                [FEOFPL={YES} [,PLOGDEV={device | ADARUN-device}] ]
                [FILE=file-list [,AUTOBACKOUT] ]
                [JCLLOG={YES|NO}]
                [OPT={YES|NO}]
                [RELGEN={number|0}]
                [RESTFILE={YES|NO}]
                [RLOGDEV={device|ADARUN-device}]
```

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 datasets 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 DD/RESTn DD/DLBLs required and allocates an equal number of input datasets for each DD/RESTn DD/DLBL.

DSIMDEV : DSIM Dataset Device Type

DSIMDEV specifies the DSIM dataset device type if different from that specified by the ADARUN DEVICE parameter, which is the default.

DSIMSIZE : Size of the DSIM Dataset

The size is specified in cylinders.

When the Adabas Delta Save 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 datasets 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 or the JCS in JCLIN). 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 datasets 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 datasets: 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 dataset 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 DD/JCLIN 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 OS or POWER JCL and JOB statements for VSE.

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 chapter).

Step Trailer

%%JCL–STEP–TRAILER

Step trailer statements are placed after each step in the recovery job stream.

DD/KARTE Job Control

%%JCL–DDKARTE

The operating-system-dependent DD/KARTE statements are included in each job step before DD/KARTE parameters generated by ADARAI from the RLOG.

For OS/390, z/OS, and VSE, these statements should indicate that the DD/KARTE parameters are contained in the job stream.

DD/FILEA 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/DLBL statement to the original DD/FILEA statement in the job. If specified, it will be inserted instead of the original DD/FILEA 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 DD/JCLIN:

%%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, DD/ASSOR1, DD/DATAR1, DD/WORKR1, DD/SORTR1, DD/TEMPR1, and so on, as needed for the utility execution;

- DD/FILEA for ADALOD if used as a DD/TEMPR1 overflow file;
- A DD/PRINT and DD/DRUCK statement or assignment;
- A DD/CARD 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 procedure or partitioned dataset (PDS) member for the DD/CARD 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
- DD/CARD 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 EDT in procedure mode for BS2000 and IEBGENER for MVS. For examples of this job control, see the %%JCL–STARTNUC sections in the examples of skeleton job control later in this chapter.

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 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 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 DD/JCLIN skeleton job control to control the generation of the DD/JCLOUT 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 DD/KARTE parameters are generated. ADARAI creates the necessary DD/KARTE 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 DD/JCLOUT 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.

REMOVE : Remove the Recovery Aid

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.

ADARAI REMOVE

The ADARAI REMOVE function disables recovery logging by updating the Associator GCB 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 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 on page 135 for information about preparing the RLOG.

Example

ADARAI REMOVE

Stops all Recovery Aid logging.

JCL/JCS Requirements and Examples

This section describes the job control information required to run ADARAI with BS2000, OS/390 or z/OS, VM/ESA or z/VM, and VSE/ESA systems and shows examples of each of the job streams.

BS2000

This section describes additional considerations and requirements for using ADARAI on a BS2000 system.

Dataset	Link Name	Storage	More Information
Associator	DDASSORn	disk	
Data Storage	DDDATARn	disk	Required for RECOVER
Work	DDWORKR1	disk	Required for RECOVER
Data protection log	DDPLOGRn	tape/disk	Required for RECOVER
Recovery log (RLOG)	DDRLOGR1	disk	
Job stream input	DDJCLIN	disk	Required for RECOVER
Recovery job output	DDJCLOUT	disk	Required for RECOVER
Recovery job output (JCL for console subtask)	DDJCLCON	disk	Optional; used only for RECOVER
ADARUN parameters	SYSDTA/ DDCARD		*Operations Manual*
ADARAI parameters	SYSDTA/ DDKARTE		*Operations Manual*
ADARUN messages	SYSOUT DDPRINT		*Messages and Codes*
ADARAI messages	SYSLST DDDRUCK		*Messages and Codes*

BS2000 Datasets

Handling Sequential Disk Datasets

There are no restrictions for sequential datasets on “public” disks.

Sequential datasets on “private” disks must not be exported; this means that the catalog entry must not be erased and that the recovery job control will not contain /IMPORT–FILE commands for datasets on private disks.

Handling Sequential Tape Datasets

Software AG recommends not removing catalog entries for sequential datasets on tapes from the system's catalog. If RECOVER does not find a catalog entry for a sequential dataset on tape or cartridge, it includes a /IMPORT–FILE FILE–NAME=tempfile,... statement in the job stream, where "tempfile" has the following structure:

#ADARAI.RECOVER.TAPE.nnnnn

Using File Generation Groups (FGGs)

There are no restrictions when using FGGs for sequential datasets, whether on disk or on tape.

Input Datasets

ADARAI tries to access sequential BS2000 datasets using the original catalog entries. If a sequential dataset is in a file generation group (FGG), ADARAI assigns the absolute member; ADARAI does not use or change the base pointer.

If sequential datasets are read-protected by passwords, the %%JCL-JOB-HEADER section in the skeleton job control must contain these passwords.

ADARAI neither exports nor erases datasets. If catalog entries for tape datasets no longer exist, ADARAI creates temporary datasets with the names #ADARAI.RECOVER.TAPE.nnnnn, where "nnnnn" equals 00001, 00002, and so on.

ADARAI JCL Examples (BS2000)

Begin Recovery Logging (ADARAI PREPARE)

In SDF Format:

```
/.ADARAI LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK *A D A R A I   START RECOVERY LOGGING
/REMARK *
/ASS-SYSLST L.RAI.DATA
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDASSOR1,DB.yyyyy.ASSO
/SET-FILE-LINK DDRLOGR1,DB.yyyyy.RLOGR1,OPEN-MODE=OUTIN,BUFF-LEN=STD(2)
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADARAI,DBID=yyyyy,MODE=MULTI
ADARAI PREPARE RLOGSIZE=5,RLOGDEV=dddd,MINGENS=5
/LOGOFF SYS-OUTPUT=DEL
```

In ISP Format:

```
/.ADARAI LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK *A D A R A I   START RECOVERY LOGGING
/REMARK *
/REMARK *
/SYSFILE SYSLST=L.RAI.DATA
/FILE ADAvrs.MOD,LINK=DDLIB
/FILE DB.yyyyy.ASSO ,LINK=DDASSOR1
/FILE DB.yyyyy.RLOGR1,LINK=DDRLOGR1,OPEN=OUTIN,BLKSIZE=(STD,2)
/EXEC (ADARUN,ADAvrs.MOD)
ADARUN PROG=ADARAI,DBID=yyyyy,MODE=MULTI
ADARAI PREPARE RLOGSIZE=5,RLOGDEV=dddd,MINGENS=5
/LOGOFF NOSPOOL
```

List the RLOG (ADARAI LIST)

In SDF Format:

```
/.ADARAI LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK *A D A R A I   LIST RECOVERY LOGS
/REMARK *
/ASS-SYSLST L.RAI.DATA
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDASSOR1,DB.yyyyy.ASSO
/SET-FILE-LINK DDRLOGR1,DB.yyyyy.RLOGR1
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADARAI,DBID=yyyyy,MODE=MULTI
ADARAI LIST GENS=NO,RLOGDEV=dddd,RELGEN=1
/LOGOFF SYS-OUTPUT=DEL
```

In ISP Format:

```
/.ADARAI LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK *A D A R A I   LIST RECOVERY LOGS
/REMARK *
/REMARK *
/SYSFILE SYSLST=L.RAI.DATA
/FILE ADAvrs.MOD,LINK=DDLIB
/FILE DB.yyyyy.ASSO ,LINK=DDASSOR1
/FILE DB.yyyyy.RLOGR1,LINK=DDRLOGR1
/EXEC (ADARUN,ADAvrs.MOD)
ADARUN PROG=ADARAI,DBID=yyyyy,MODE=MULTI
ADARAI LIST GENS=NO,RLOGDEV=dddd,RELGEN=1
/LOGOFF NOSPOOL
```

Create Recovery JCL (ADARAI RECOVER)

In SDF Format:

```
/.ADARAI LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK *A D A R A I   BUILD RECOVERY JCL STREAM
/REMARK *
/DELETE-FILE DB.yyyyy.JCLOUT
/SET-JOB-STEP
/CREATE-FILE DB.yyyyy.JCLOUT,PUB(SPACE=(48,48))
/SET-JOB-STEP
/DELETE-FILE DB.yyyyy.JCLCON
/SET-JOB-STEP
/CREATE-FILE DB.yyyyy.JCLCON,PUB(SPACE=(48,48))
/SET-JOB-STEP
/ASS-SYSLST L.RAI.DATA
/ASS-SYSDTA *SYSCMD
```

```
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDASSOR1,DB.yyyyy.ASSO
/SET-FILE-LINK DDRLOGR1,DB.yyyyy.RLOGR1
/SET-FILE-LINK DDPLOGR1,DB.yyyyy.PLOGR1
/SET-FILE-LINK DDPLOGR2,DB.yyyyy.PLOGR2
/SET-FILE-LINK DDJCLIN,DB.yyyyy.JCLIN
/SET-FILE-LINK DDJCLOUT,DB.yyyyy.JCLOUT
/SET-FILE-LINK DDJCLCON,DB.yyyyy.JCLCON
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADARAI,DBID=yyyyy,MODE=MULTI
ADARUN UEX2=EXITR2
ADARAI RECOVER PLOGDEV=2201,FEOFPL=YES,RELGEN=1
/LOGOFF SYS-OUTPUT=DEL
```

In ISP Format:

```
/.ADARAI LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK *A D A R A I   BUILD RECOVERY JCL STREAM
/REMARK *
/REMARK *


/SYSFILE SYSLST=L.RAI.DATA
/FILE ADAvrs.MOD,LINK=DDLIB
/FILE DB.yyyyy.ASSO,LINK=DDASSOR1
/FILE DB.yyyyy.RLOGR1,LINK=DDRLOGR1
/FILE DB.yyyyy.PLOGR1,LINK=DDPLOGR1
/FILE DB.yyyyy.PLOGR2,LINK=DDPLOGR2
/FILE DB.yyyyy.JCLIN,LINK=DDJCLIN
/FILE DB.yyyyy.JCLOUT,LINK=DDJCLOUT
/FILE DB.yyyyy.JCLCON,LINK=DDJCLCON
/EXEC (ADARUN,ADAvrs.MOD)
ADARUN PROG=ADARAI,DBID=yyyyy,MODE=MULTI
ADARUN UEX2=EXITR2
ADARAI RECOVER PLOGDEV=2201,FEOFPL=YES,RELGEN=1
/LOGOFF NOSPOOL
```

Skeleton Job Control

For a BS2000 system, the skeleton job control should have the following characteristics:

- The first two positions must be blank;
- Statements can be up to 256 characters long, **including** the blanks in the first two positions;
- Statements can be continued; the continuation mark (–) can be in any position (it does not have to be in position 74);
- In version 10, ADARAI automatically adds the command:

  /MODIFY-SDF-OPTIONS CONTINUATION=NEW-MODE

- Job control statements for the BS2000 executive are automatically broken up into 72-byte segments, if necessary. Normal program control statements are **not** segmented.
- Except for string substitution (described below), ADARAI does not check or change the job control statement syntax.

In addition to the general JCL characteristics described above, BS2000 systems can include the following in the skeleton JCL:

- Substitution strings for repeating often-used job control input;
- DDFILEA dataset overwriting, tape volume assignment, and disk space credit;
- Subtask processing of RECOVER-generated console messages.

These options must be specified at the beginning of the skeleton JCL, immediately before the %%JCL-JOB-HEADER statement.

Specifying Substitution Strings

Frequently occurring strings such as dataset names can be defined in the substitution section. The strings are then inserted into the BS2000 JCL as well as in the user program control statements.

To use string substitution, include the following JCL statement before %%JCL-JOB-HEADER:

%%JCL-BS2-SUBSTITUTION

This statement is followed by the substitution definitions, which have the following format:

%%argname=substring

—where “argname” is the 1- to 8-character JCL or user program control argument to be replaced, and “substring” is the replacement string of up to 128 characters. One substitution statement per line is allowed; the substitution statements are ended by the next %%JCL statement.

DDFILEA JCL Options

The JCL statements described in this section must appear before %%JCL-JOB-HEADER in the skeleton JCL.

ADAORD is the only utility that opens DDFILEA output datasets during the BS2000 recovery job. ADARAI assists in allocating those datasets on disk or tape regardless of whether the datasets existed at RECOVER time.

Existing “tape” datasets are never overwritten. To overwrite existing DDFILEA “disk” datasets, specify the following statement in the skeleton JCL:

%%JCL-BS2-WORK-DATASET-OVERWRITE=YES

OVERWRITE defaults to NO. If no DDFILEA assignments are found and overwriting is prohibited (the default), ADARAI tries to write the DDFILEA datasets on TAPE-C1 cartridges.

To allocate DDFILEA on disk instead of tape, include the following JCL statement:

%%JCL-BS2-WORK-DISK-SPACE

Then specify one or more disk space assignment statements, as follows:

***:* catid *:*=pam-pages**

ADARAI checks the availability of the specified pubsets and for permission of the ADARAI task’s “/LOGON userid” to allocate the specified number of PAM pages on those pubsets. If the checks fail, the user’s logon ID must be added to the joinfile of the related pubset, and the ADARAI RECOVER job step must be repeated before starting the generated recovery job. Otherwise, the ADAORD job steps may ABEND.

To assign DDFILEA output tape devices and volumes, include the following statement:

%%JCL-BS2-WORK-TAPE-VOLUMES=device-type

If output tape datasets must be created, ADARAI uses temporary datasets named #ADARAI.RECOVER.TAPE.nnnnn, where “nnnnn” equals 00001, 00002, and so on. If the console subtask option described in the next section is enabled, each line can contain one or more volser numbers separated by blanks or commas. The first two positions on each line must be blanks; the maximum line length is 256 characters.

The following is an example:

```
%%JCL-BS2-WORK-TAPE-VOLUMES=TAPE-C1
  A00001,A00002,A00003,A00004
  A00005,A00006,A00007,A00008
```

Up to 512 volumes can be specified.

BS2000 Console Subtask

When a recovery job created by ADARAI RECOVER is submitted, console messages may occur that require operator intervention, such as the following:

- The catalog entry for a tape dataset is not available, and ADARAI inserts a /IMPORT-FILE statement for a temporary tape dataset, causing a DMS0DA5 console message.
- Tape output for the DDFILEA dataset is required, and ADARAI includes JCL that causes a message requesting that a scratch tape be mounted.

These messages can be answered automatically by a UCON program, which runs as a subtask in parallel with the recovery job. The UCON program (ADAR2C) receives all relevant console messages and sends answers whenever possible.

To run a console subtask, include the following section before %%JCL-JOB-HEADER:

%%JCL-BS2-CONSOLE-SUBTASK-SPECIFICATION
CONSOLE-NAME=name,*C'*password*'* (this statement is required)
DCAM-APPL=dcamappl (this statement is optional)

—where

name	is a /LOGON userid;
password	is the user ID's logon password; and
dcamappl	is the name of a DCAM application (the default is "RAIRUCON").

For more information, see the **Authorized User Tasks** chapter in the BS2000 *Systems Administration* manual.

The following optional statement can be included in the skeleton JCL, followed by volser definitions. If the console subtask is not called, the volser definitions have no effect.

%%JCL-BS2-WORK-TAPE-VOLUMES=device-type

Include the following statement in the JCL for the ADARAI RECOVER function (**not** in the skeleton JCL):

/SET-FILE-LINK FILE-NAME=console-job,LINK-NAME=DDJCLCON

The following example JCL for the RECOVER function includes the console subtask:

```
/  LOGON
/  SET-FILE-LINK  FILE-NAME=ADA.JCLIN,LINK-NAME=DDJCLIN
/  SET-FILE-LINK  FILE-NAME=ADA.JCLOUT,LINK-NAME=DDJCLOUT
/  SET-FILE-LINK  FILE-NAME=ADA.JCLCON,LINK-NAME=DDJCLCON
```

```
.
.
/  ASSIGN-SYSDTA  TO-FILE=*SYSCMD
/  SET-FILE-LINK  ADAvrs.MOD,LINK-NAME=DDLIB
/  START-PROG *M(E=ADARUN,L=ADAvrs.MOD)
ADARUN PROG=ADARAI,DB=47
ADARAI RECOVER
/  LOGOFF NOSPOOL
```

The console subtask evaluates the following BS2000 console messages, where “tsn” is the task serial number, and “mn” is the mnemo-technical device name:

NKVT010 VOLUME volser IS MOUNTED ON DEVICE mn

If the volser and device are specified as described in the section **DDFILEA JCL Options** on page 164, they are registered in the subtask’s online volser table. If the message NKVT013 is also outstanding at this time, the subtask returns the response “tsn.mn”.

NKVT011/97 VOLUME volser IS DISMOUNTED FROM DEVICE mn...

The volser is removed from the online volser table.

NKVT013 MOUNT TAPE ‘*SCRAT’ ON DEVICE ...

If this message is related to the RECOVER task and the volser is available in the online volser table, the subtask sends the response “tsn.mn”.

DMS0DFB ACKNOWLEDGE VSN volser ON DEVICE mn...

If this message is related to the RECOVER task, the subtask sends the response “tsn”.

DMS0DA5 INVALID FILE SPECIFICATION: VSN volser FOR FILE file...

If the related “tsn” is the RECOVER task’s “tsn” and “file” is a temporary dataset with the name #ADARAI.RECOVER.TAPE.nnnnn, the message is answered with the response “tsn.I”.

For the subtask to respond, the logon ID must be able to ignore tapes with incorrect file IDs. To enable this, issue the following statement under the system administrator’s logon ID (TSOS):

/MOD-USER userid,PROTECTION-ATTRIBUTE=(TAPE-ACCESS=READ)

Skeleton Job Control Example (BS2000)

```
%%JCL-BS2-WORK-DATASET-OVERWRITE = NO
%%JCL-BS2-WORK-DISK-SPACE
      :A: = 500000
      :B: = 1000000
%%JCL-BS2-CONSOLE-SUBTASK-SPECIFICATION
  CONSOLE-NAME=CON1,C'PASSWORD'
%%JCL-BS2-WORK-TAPE-VOLUMES=TAPE-C1
   A00001,A00002,A00003,A00004,A00005
   A00006,A00007,A00008,A00009,A00010
%%JCL-BS2-SUBSTITUTION
  %%USERID = ADAvr
  %%ACCOUNT = 1
  %%ASSOR1 = $ADAvr.DByyyyy.ASSOR1
  %%ASSOR2 = $ADAvr.DByyyyy.ASSOR2
  %%DATAR1 = $ADAvr.DByyyyy.DATAR1
  %%DATAR2 = $ADAvr.DByyyyy.DATAR2
  %%WORK   = $ADAvr.DByyyyy.WORK
  %%SORT   = $ADAvr.DByyyyy.SORT%
  %%TEMP   = $ADAvr.DByyyyy.TEMP
  %%RLOGR1 = $ADAvr.DByyyyy.RLOGR1
  %%PLOGR1 = $ADAvr.DByyyyy.PLOGR1
  %%PLOGR2 = $ADAvr.DByyyyy.PLOGR2
  %%DBID   = yyyyy
  %%IDTNAME = ADABASvB
  %%USEREX  = ADAvrs.USEREXITS.MOD
  %%DDLIB   = ADAvrs.MOD
  %%DEVICE  = dddd
%%JCL-JOB-HEADER
  /.RECOVER LOGON %%USERID,%%ACCOUNT
  /       SET-FILE-LINK FILE-NAME=%%USEREX,LINK-NAME=BLSLIB01
  /       SET-FILE-LINK FILE-NAME=%%DDLIB,LINK-NAME=DDLIB
  /       ASSIGN-SYSOUT TO-FILE=$ADAvv.RAI.OUT
  /       ASSIGN-SYSLST TO-FILE=$ADAvv.RAI.LST
%%JCL-JOB-TRAILER
  /REMARK ------------------------------------------------------------*
  /REMARK *          RECOVERY JOB SUCCESSFULLY TERMINATED            *
  /REMARK ------------------------------------------------------------*
  /LOGOFF NOSPOOL
  /.JOBERROR REMARK
  /REMARK ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
  /REMARK *          RECOVERY JOB TERMINATED WITH ERROR              *
  /REMARK ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
  /LOGOFF NOSPOOL
%%JCL-ADARES
```

```
  /REMARK
  /REMARK REGENERATE / BACKOUT
  /REMARK
  /       SET-FILE-LINK FILE-NAME=%%RLOGR1,LINK-NAME=DDRLOGR1,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%ASSOR1,LINK-NAME=DDASSOR1,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%ASSOR2,LINK-NAME=DDASSOR2,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%DATAR1,LINK-NAME=DDDATAR1,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%DATAR2,LINK-NAME=DDDATAR2,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%WORK,LINK-NAME=DDWORKR1,         -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=RAI.DRU.JCLO.ADARES,LINK-NAME=DDDRUCK
  /       SET-FILE-LINK FILE-NAME=RAI.PRI.JCLO.ADARES,LINK-NAME=DDPRINT
  %SEQUENTIAL
  /       ASSIGN-SYSDTA TO-FILE=*SYSCMD
  /       START-PROGRAM FROM-FILE=*MODULE(ELEMENT=ADARUN,LIBRARY=%%DDLIB)
  ADARUN MODE=MULTI,PROG=ADARES
  ADARUN DBID=%%DBID,DE=%%DEVICE,IDTNAME=%%IDTNAME
  %KARTE
%%JCL-STEP-TRAILER
  /SET-JOB-STEP%%JCL-ADASAV
  /REMARK
  /REMARK RESTORE FILE(S)/DATABASE
  /REMARK
  /.%STEP REMARK
  /       SET-FILE-LINK FILE-NAME=%%RLOGR1,LINK-NAME=DDRLOGR1,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%ASSOR1,LINK-NAME=DDASSOR1,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%ASSOR2,LINK-NAME=DDASSOR2,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%DATAR1,LINK-NAME=DDDATAR1,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%DATAR2,LINK-NAME=DDDATAR2,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%WORK,LINK-NAME=DDWORKR1,         -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=RAI.DRU.JCLO.RESTORE,LINK-NAME=DDDRUCK
  /       SET-FILE-LINK FILE-NAME=RAI.PRI.JCLO.RESTORE,LINK-NAME=DDPRINT
  %SEQUENTIAL
  /       ASSIGN-SYSDTA TO-FILE=*SYSCMD
  /       START-PROGRAM FROM-FILE=*MODULE(ELEMENT=ADARUN,LIBRARY=%%DDLIB)
```

```
  ADARUN MODE=MULTI,PROG=ADASAV
  ADARUN DBID=%%DBID,DE=%%DEVICE,IDTNAME=%%IDTNAME
  %KARTE
%%JCL-ENDNUC
  /REMARK
  /REMARK ADADBS END NUCLEUS
  /REMARK
  /       SET-FILE-LINK FILE-NAME=%%RLOGR1,LINK-NAME=DDRLOGR1,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%ASSOR1,LINK-NAME=DDASSOR1,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%ASSOR2,LINK-NAME=DDASSOR2,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%DATAR1,LINK-NAME=DDDATAR1,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%DATAR2,LINK-NAME=DDDATAR2,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%WORK,LINK-NAME=DDWORKR1,         -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%SORT,LINK-NAME=DDSORTR1
  /       SET-FILE-LINK FILE-NAME=%%TEMP,LINK-NAME=DDTEMPR1
  /       SET-FILE-LINK FILE-NAME=RAI.DRU.JCLO.ADADBS,LINK-NAME=DDDRUCK
  /       SET-FILE-LINK FILE-NAME=RAI.PRI.JCLO.ADADBS,LINK-NAME=DDPRINT
  /       ASSIGN-SYSDTA TO-FILE=*SYSCMD
  /       START-PROGRAM FROM-FILE=*MODULE(ELEMENT=ADARUN,LIBRARY=%%DDLIB)
  ADARUN MODE=MULTI,PROG=ADADBS
  ADARUN DBID=%%DBID,DE=%%DEVICE,IDTNAME=%%IDTNAME
  ADADBS OPERCOM ADAEND
  /REMARK
  /REMARK CHECK INACTIVE DATABASE
  /REMARK
  /       SET-FILE-LINK FILE-NAME=RAI.DRU.JCLO.ADARAI,LINK-NAME=DDDRUCK
  /       SET-FILE-LINK FILE-NAME=RAI.PRI.JCLO.ADARAI,LINK-NAME=DDPRINT
  /       ASSIGN-SYSDTA TO-FILE=*SYSCMD
  /       START-PROGRAM FROM-FILE=*MODULE(ELEMENT=ADARUN,LIBRARY=%%DDLIB)
  ADARUN MODE=MULTI,PROG=ADARAI,DBID=%%DBID,DE=%%DEVICE,IDTNAME=%%IDTNAME
  ADARAI CHKDB INACTIVE
%%JCL-ADADEF
  /REMARK
  /REMARK DEFINE NEWWORK
  /REMARK
  /       SET-FILE-LINK FILE-NAME=%%RLOGR1,LINK-NAME=DDRLOGR1,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%ASSOR1,LINK-NAME=DDASSOR1,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%ASSOR2,LINK-NAME=DDASSOR2,       -
```

```
  /                    SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%DATAR1,LINK-NAME=DDDATAR1,       -
  /                    SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%DATAR2,LINK-NAME=DDDATAR2,       -
  /                    SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%WORK,LINK-NAME=DDWORKR1,         -
  /                    SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%SORT,LINK-NAME=DDSORTR1
  /       SET-FILE-LINK FILE-NAME=%%TEMP,LINK-NAME=DDTEMPR1
  /       SET-FILE-LINK FILE-NAME=RAI.DRU.JCLO.ADADEF,LINK-NAME=DDDRUCK
  /       SET-FILE-LINK FILE-NAME=RAI.PRI.JCLO.ADADEF,LINK-NAME=DDPRINT
  /       ASSIGN-SYSDTA TO-FILE=*SYSCMD
  /       START-PROGRAM FROM-FILE=*MODULE(ELEMENT=ADARUN,LIBRARY=%%DDLIB)
  ADARUN MODE=MULTI,PROG=ADADEF
  ADARUN DBID=%%DBID,DE=%%DEVICE,IDTNAME=%%IDTNAME
  %KARTE
%%JCL-ADALOD
  /REMARK
  /REMARK LOAD A FILE / MASS UPDATE
  /REMARK
  /       SET-FILE-LINK FILE-NAME=%%RLOGR1,LINK-NAME=DDRLOGR1,       -
  /                    SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%ASSOR1,LINK-NAME=DDASSOR1,       -
  /                    SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%ASSOR2,LINK-NAME=DDASSOR2,       -
  /                    SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%DATAR1,LINK-NAME=DDDATAR1,       -
  /                    SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%DATAR2,LINK-NAME=DDDATAR2,       -
  /                    SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%WORK,LINK-NAME=DDWORKR1,         -
  /                    SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%SORT,LINK-NAME=DDSORTR1
  /       SET-FILE-LINK FILE-NAME=%%TEMP,LINK-NAME=DDTEMPR1
  /       SET-FILE-LINK FILE-NAME=RAI.DRU.JCLO.ADALOD,LINK-NAME=DDDRUCK
  /       SET-FILE-LINK FILE-NAME=RAI.PRI.JCLO.ADALOD,LINK-NAME=DDPRINT
  %SEQUENTIAL
  /       ASSIGN-SYSDTA TO-FILE=*SYSCMD
  /       START-PROGRAM FROM-FILE=*MODULE(ELEMENT=ADARUN,LIBRARY=%%DDLIB)
  ADARUN MODE=MULTI,PROG=ADALOD
  ADARUN DBID=%%DBID,DE=%%DEVICE,IDTNAME=%%IDTNAME
  %KARTE
%%JCL-ADAORD
  /REMARK
  /REMARK REORDER
  /REMARK
```

```
  /       SET-FILE-LINK FILE-NAME=%%RLOGR1,LINK-NAME=DDRLOGR1,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%ASSOR1,LINK-NAME=DDASSOR1,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%ASSOR2,LINK-NAME=DDASSOR2,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%DATAR1,LINK-NAME=DDDATAR1,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%DATAR2,LINK-NAME=DDDATAR2,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%WORK,LINK-NAME=DDWORKR1,         -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%SORT,LINK-NAME=DDSORTR1
  /       SET-FILE-LINK FILE-NAME=%%TEMP,LINK-NAME=DDTEMPR1
  /       SET-FILE-LINK FILE-NAME=RAI.DRU.JCLO.ADAORD,LINK-NAME=DDDRUCK
  /       SET-FILE-LINK FILE-NAME=RAI.PRI.JCLO.ADAORD,LINK-NAME=DDPRINT
  %SEQUENTIAL
  /       ASSIGN-SYSDTA TO-FILE=*SYSCMD
  /       START-PROGRAM FROM-FILE=*MODULE(ELEMENT=ADARUN,LIBRARY=%%DDLIB)
  ADARUN MODE=MULTI,PROG=ADAORD
  ADARUN DBID=%%DBID,DE=%%DEVICE,IDTNAME=%%IDTNAME
  %KARTE
%%JCL-ADAINV
  /REMARK
  /REMARK INVERT / COUPLE
  /REMARK
  /       SET-FILE-LINK FILE-NAME=%%RLOGR1,LINK-NAME=DDRLOGR1,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%ASSOR1,LINK-NAME=DDASSOR1,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%ASSOR2,LINK-NAME=DDASSOR2,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%DATAR1,LINK-NAME=DDDATAR1,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%DATAR2,LINK-NAME=DDDATAR2,       -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%WORK,LINK-NAME=DDWORKR1,         -
  /                     SUP=DISK(SHARE-UPD=YES)
  /       SET-FILE-LINK FILE-NAME=%%SORT,LINK-NAME=DDSORTR1
  /       SET-FILE-LINK FILE-NAME=%%TEMP,LINK-NAME=DDTEMPR1
  /       SET-FILE-LINK FILE-NAME=RAI.DRU.JCLO.ADADBS,LINK-NAME=DDDRUCK
  /       SET-FILE-LINK FILE-NAME=RAI.PRI.JCLO.ADADBS,LINK-NAME=DDPRINT
  /       ASSIGN-SYSDTA TO-FILE=*SYSCMD
  /       START-PROGRAM FROM-FILE=*MODULE(ELEMENT=ADARUN,LIBRARY=%%DDLIB)
  ADARUN MODE=MULTI,PROG=ADAINV
  ADARUN DBID=%%DBID,DE=%%DEVICE,IDTNAME=%%IDTNAME
```

```
  %KARTE
%%JCL-STARTNUC
  /REMARK
  /REMARK START NUCLEUS
  /REMARK
  /       MODIFY-JOB-SWITCHES ON=(4,5)
  /       ASSIGN-SYSDTA TO-FILE=*SYSCMD
  /       START-PROGRAM FROM-FILE=EDT
  :/.ADANUC LOGON %%USERID,%%ACCOUNT
  :/    SET-FILE-LINK FILE-NAME=%%RLOGR1,LINK-NAME=DDRLOGR1,-
  :/                  SUP=DISK(SHARE-UPD=YES)
  :/    SET-FILE-LINK FILE-NAME=%%PLOGR1,LINK-NAME=DDPLOGR1,-
  :/                  SUP=DISK(SHARE-UPD=YES)
  :/    SET-FILE-LINK FILE-NAME=%%PLOGR2,LINK-NAME=DDPLOGR2,-
  :/                  SUP=DISK(SHARE-UPD=YES)
  :/    SET-FILE-LINK FILE-NAME=%%ASSOR1,LINK-NAME=DDASSOR1,-
  :/                  SUP=DISK(SHARE-UPD=YES)
  :/    SET-FILE-LINK FILE-NAME=%%ASSOR2,LINK-NAME=DDASSOR2,-
  :/                  SUP=DISK(SHARE-UPD=YES)
  :/    SET-FILE-LINK FILE-NAME=%%DATAR1,LINK-NAME=DDDATAR1,-
  :/                  SUP=DISK(SHARE-UPD=YES)
  :/    SET-FILE-LINK FILE-NAME=%%DATAR2,LINK-NAME=DDDATAR2,-
  :/                  SUP=DISK(SHARE-UPD=YES)
  :/    SET-FILE-LINK FILE-NAME=%%WORK,LINK-NAME=DDWORKR1,-
  :/                  SUP=DISK(SHARE-UPD=YES)
  :/    SET-FILE-LINK FILE-NAME=RAI.DRU.JCLO.ADANUC,LINK-NAME=DDDRUCK
  :/    SET-FILE-LINK FILE-NAME=RAI.PRI.JCLO.ADANUC,LINK-NAME=DDPRINT
  :/    ASSIGN-SYSDTA TO-FILE=*SYSCMD
  :/    SET-FILE-LINK FILE-NAME=%USEREX,LINK-NAME=BLSLIB01
  :/    SET-FILE-LINK FILE-NAME=%%DDLIB,LINK-NAME=DDLIB
  :/    START-PROGRAM FROM-FILE=*MODULE(ELEMENT=ADARUN,LIBRARY=%%DDLIB)
  :ADARUN PROG=ADANUC
  :ADARUN DATABASE=%%DBID           DATA BASE ID
  :ADARUN IDTNAME=%%IDTNAME                NAME OF IDT
  :ADARUN DEVICE=%%DEVICE
  :ADARUN IDTNAME=%%IDTNAME
  :ADARUN LFIOP=1000000,LWP=800000,LBP=4000000,LU=64000
  :ADARUN LP=200,LS=20000,TT=900,TNAE=900,OPENRQ=NO,PLOGRQ=YES
  :ADARUN NAB=20,NH=400,NU=200,NISNHQ=50,NC=20,NPLOG=2,PLOGDEV=2000
  :ADARUN PLOGSIZE=1000,NCLOG=2,CLOGDEV=2000,CLOGSIZE=500
  :ADARUN UEX2=USEREX2
  :/LOGOFF SYSOUT=DEL
  @D &:1-1:
  @W '#NUCENT' O
  @H
  /STEP
```

```
/ENTER-JOB #NUCENT,PROC-ADMIS=PAR(USER-ID=%%USERID,                -
/                                 ACCOUNT=%%ACCOUNT)
/STEP
/       SET-FILE-LINK FILE-NAME= %%DDLIB,LINK-NAME=DDLIB
/       SET-FILE-LINK FILE-NAME=RAI.DRU.JCLO.ADARES,LINK-NAME=DDDRUCK
/       SET-FILE-LINK FILE-NAME=RAI.PRI.JCLO.ADARES,LINK-NAME=DDPRINT
/       ASSIGN-SYSDTA TO-FILE=*SYSCMD
/       START-PROGRAM FROM-FILE=*MODULE(ELEMENT=ADARUN,LIBRARY=%%DDLIB)
ADARUN MODE=MULTI,PROG=ADARAI
ADARUN DBID=%%DBID,DE=%%DEVICE,IDTNAME=%%IDTNAME
ADARAI CHKDB ACTIVE
```

ADAR2E Utility

If a pubset member is lost because of, for example, a head crash or other unrecoverable hardware error, initialize a new member, include it in the pubset, and restore all datasets in the pubset. In most cases, you must also reallocate the ASSO, DATA, WORK, and PLOG datasets on exactly the same disk locations they were on before the hardware failure.

The ADAR2E utility reads the TSOSCAT entries of assigned datasets and creates a procedure that reallocates the Adabas files at exactly the same disk locations where they were before. The ADAR2E utility operates totally independently of ADARAI and other Adabas utilities; ADAR2E should be run whenever disk allocation changes are made to the major Adabas components (ASSO, DATA, WORK, PLOG, CLOG, TEMP, or SORT).

The following example JCL is for running the ADAR2E utility:

```
/  LOGON
/  SET-FILE-LINK FILE-NAME=DB.yyyyy.ASSO-01,LINK-NAME=DDASSOR1
/  SET-FILE-LINK FILE-NAME=DB.yyyyy.DATA-01,LINK-NAME=DDDATAR1
/  SET-FILE-LINK FILE-NAME=DB.yyyyy.WORK-01,LINK-NAME=DDWORKR1
/  SET-FILE-LINK FILE-NAME=DB.yyyyy.TEMP-01,LINK-NAME=DDTEMPR1
/  SET-FILE-LINK FILE-NAME=DB.yyyyy.PLOGR1,LINK-NAME=DDPLOGR1
/  SET-FILE-LINK FILE-NAME=DB.yyyyy.PLOGR2,LINK-NAME=DDPLOGR2
/  SET-FILE-LINK FILE-NAME=ALLOCATE.JOB,LINK-NAME=DDJCLOUT
/  START-PROGRAM *MOD(LIB=ADAvrs,ELEM=ADAR2E)
/  LOGOFF SYS-OUTPUT=DEL
```

The following is a list of available link names:

```
DDASSOR1,DDASSOR2,DDASSOR3,DDASSOR4,DDASSOR5
DDDATAR1,DDDATAR2,DDDATAR3,DDDATAR4,DDDATAR5
DDWORKR1,DDWORKR2
DDSORTR1,DDSORTR2
DDTEMPR1,DDTEMPR2
DDPLOGR1,DDPLOGR2,DDPLOGR3,DDPLOGR4,DDPLOGR5,DDPLOGR6,DDPLOGR7,DDPLOGR8
```

The following is an example of the job control created by the ADAR2E utility:

```
/  BEGIN-PROCEDURE
/  MODIFY-SDF-OPTIONS CONTINUATION=NEW-MODE
/  CREATE-FILE FILE-NAME=:A:$ADABAS.DB.yyyyy.ASSO-01,-
/       SUPPORT=PUBLIC-DISK(VOLUME=PUBA00,-
/       DEVICE-TYPE=Ddddd,SPACE=ABSOLUTE-
/       (FIRST-PAGE=32833,SIZE=5001))
/  MODI-FILE-ATTR FILE-NAME=:A:$ADABAS.DB.yyyyy.ASSO-01,-
/       SUPPORT=PUBLIC-DISK(VOLUME=PUBA00,-
/       DEVICE-TYPE=Ddddd,SPACE=ABSOLUTE-
/       (FIRST-PAGE=29761,SIZE=5001))
        .
        .
/  CREATE-FILE FILE-NAME=:A:$ADABAS.DB.yyyyy.DATA-01,-
/       SUPPORT=PUBLIC-DISK(VOLUME=PUBA02,-
/       DEVICE-TYPE=Ddddd,SPACE=ABSOLUTE-
/       (FIRST-PAGE=119809,SIZE=8001))
/  MODI-FILE-ATTR FILE-NAME=:A:$ADABAS.DB.yyyyy.DATA-01,-
/       SUPPORT=PUBLIC-DISK(VOLUME=PUBA03,-
/       DEVICE-TYPE=Ddddd,SPACE=ABSOLUTE-
/       (FIRST-PAGE=75841,SIZE=8001))
```

Execute the ADAR2E-generated reallocation procedure under the system administrator's logon ID (TSOS) before starting the ADARAI RECOVER function.

Under BS2000 version 11 and above, it is possible to allow all users with access rights to a certain pubset to allocate disk space absolutely by issuing the following command under TSOS:

/MODIFY-MASTER-CATALOG CATID=*catid*,PHYSICAL-ALLO=USER-ALLOWED

OS/390 or z/OS

Dataset	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 Manual*
ADARAI parameters	DDKARTE	reader	

JCL Examples (OS/390 or z/OS)

Prepare for Recovery Logging (ADARAI PREPARE):

```
//RAIPREP   EXEC PGM=ADARUN
//STEPLIB   DD   DISP=SHR,DSN=ADABAS.Vvrs.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.Vvrs.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.Vvrs.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 (OS/390 or z/OS)

```
%%JCL-JOB-HEADER
  //ADARECOV JOB 5,'ADA-USER',MSGCLASS=X,CLASS=A,REGION=4096K
  //*
  //JOBLIB DD DSN=ADABAS.Vvrs.LOAD,DISP=SHR
  //*
%%JCL-JOB-TRAILER
  //*
  //* END OF RECOVERY
  //*
%%JCL-STEP-TRAILER
  //*
  //* END OF STEP
  //*
%%JCL-DDKARTE
  //DDKARTE DD *
%%JCL-ADASAV
  //*
  //* RESTORE FILE(S)/DATABASE
  //*
  //%STEP    EXEC PGM=ADARUN        ADASAV JOB STEP
  %SEQUENTIAL
  //DDRLOGR1 DD DSN=ADABAS.Vvrs.RLOGR1,DISP=SHR
  //DDASSOR1 DD DSN=ADABAS.Vvrs.ASSOR1,DISP=SHR
  //DDDATAR1 DD DSN=ADABAS.Vvrs.DATAR1,DISP=SHR
  //DDWORKR1 DD DSN=ADABAS.Vvrs.WORKR1,DISP=SHR
  //DDDRUCK  DD SYSOUT=X
  //DDPRINT  DD SYSOUT=X
  //DDCARD   DD *
   ADARUN MODE=MULTI,PROG=ADASAV,DBID=yyyyy,DEVICE=dddd,SVC=xxx
  //DDKARTE  DD *
  %KARTE
%%JCL-ADARES
  //*
  //* REGENERATE/BACKOUT
  //*
  //%STEP    EXEC PGM=ADARUN        ADARES JOB STEP
  %SEQUENTIAL
  //DDRLOGR1 DD DSN=ADABAS.Vvrs.RLOGR1,DISP=SHR
  //DDASSOR1 DD DSN=ADABAS.Vvrs.ASSOR1,DISP=SHR
  //MERGINTI DD DSN=ADABAS.Vvrs.MERGINTI,DISP=SHR
  //MERGINTO DD DSN=ADABAS.Vvrs.MERGINTO,DISP=SHR
  //DDDRUCK  DD SYSOUT=X
  //DDPRINT  DD SYSOUT=X
  //SYSUDUMP DD SYSOUT=X
```

```
  //DDCARD   DD *
  ADARUN MODE=MULTI,PROG=ADARES,DBID=yyyyy,DEVICE=dddd,SVC=xxx
  //DDKARTE  DD *
  %KARTE
%%JCL-ADADEF
  //* DEFINE NEWWORK
  //*
  //%STEP    EXEC PGM=ADARUN        ADADEF JOB STEP
  //DDRLOGR1 DD DSN=ADABAS.Vvrs.RLOGR1,DISP=SHR
  //DDASSOR1 DD DSN=ADABAS.Vvrs.ASSOR1,DISP=SHR
  //DDDATAR1 DD DSN=ADABAS.Vvrs.DATAR1,DISP=SHR
  //DDWORKR1 DD DSN=ADABAS.Vvrs.WORKR1,DISP=SHR
  //DDTEMPR1 DD DSN=ADABAS.Vvrs.TEMPR1,DISP=SHR
  //DDSORTR1 DD DSN=ADABAS.Vvrs.SORTR1.DISP=SHR
  //DDDRUCK  DD SYSOUT=X
  //DDPRINT  DD SYSOUT=X
  //DDCARD   DD *
  ADARUN MODE=MULTI,PROG=ADADEF,DBID=yyyyy,DEVICE=dddd,SVC=xxx
  //DDKARTE  DD *
  %KARTE
%%JCL-ADALOD
  //*
  //* LOAD A FILE/MASS UPDATE
  //*
  //%STEP    EXEC PGM=ADARUN        ADALOD JOB STEP
  %SEQUENTIAL
  //DDRLOGR1 DD DSN=ADABAS.Vvrs.RLOGR1,DISP=SHR
  //DDASSOR1 DD DSN=ADABAS.Vvrs.ASSOR1,DISP=SHR
  //DDDATAR1 DD DSN=ADABAS.Vvrs.DATAR1,DISP=SHR
  //DDWORKR1 DD DSN=ADABAS.Vvrs.WORKR1,DISP=SHR
  //DDTEMPR1 DD DSN=ADABAS.Vvrs.TEMPR1,DISP=SHR
  //DDSORTR1 DD DSN=ADABAS.Vvrs.SORTR1.DISP=SHR
  //DDFILEA  DD DSN=ADABAS.Vvrs.SORT.OVERFLOW
  //DDDRUCK  DD SYSOUT=X
  //DDPRINT  DD SYSOUT=X
  //DDCARD   DD *
  ADARUN MODE=MULTI,PROG=ADALOD,DBID=yyyyy,DEVICE=dddd,SVC=xxx
  //DDKARTE  DD *
  %KARTE
%%JCL-ADAORD
  //*
  //* REORDER
  //*
  //%STEP    EXEC PGM=ADARUN        ADAORD JOB STEP
  %SEQUENTIAL
  //DDRLOGR1 DD DSN=ADABAS.Vvrs.RLOGR1,DISP=SHR
```

```
  //DDASSOR1 DD DSN=ADABAS.Vvrs.ASSOR1,DISP=SHR
  //DDDATAR1 DD DSN=ADABAS.Vvrs.DATAR1,DISP=SHR
  //DDWORKR1 DD DSN=ADABAS.Vvrs.WORKR1,DISP=SHR
  //DDDRUCK  DD SYSOUT=X
  //DDPRINT  DD SYSOUT=X
  //DDCARD   DD *
  ADARUN MODE=MULTI,PROG=ADAORD,DBID=yyyyy,DEVICE=dddd,SVC=xxx
  //DDKARTE  DD *
  %KARTE
%%JCL-ADAINV
  //* INVERT / COUPLE
  //*
  //%STEP    EXEC PGM=ADARUN        ADAINV JOB STEP
  //DDRLOGR1 DD DSN=ADABAS.Vvrs.RLOGR1,DISP=SHR
  //DDASSOR1 DD DSN=ADABAS.Vvrs.ASSOR1,DISP=SHR
  //DDDATAR1 DD DSN=ADABAS.Vvrs.DATAR1,DISP=SHR
  //DDWORKR1 DD DSN=ADABAS.Vvrs.WORKR1,DISP=SHR
  //DDDRUCK  DD SYSOUT=X
  //DDPRINT  DD SYSOUT=X
  //SYSUDUMP DD SYSOUT=X
  //DDCARD   DD *
  ADARUN MODE=MULTI,PROG=ADAINV,DBID=yyyyy,DEVICE=dddd,SVC=xxx
  //DDKARTE  DD *
  %KARTE
%%JCL-STARTNUC
  //* START NUCLEUS
  //*
  //STEP1    EXEC   PGM=IEBGENER                (submit to internal reader)
  //SYSPRINT DD     SYSOUT=X
  //SYSIN    DD     DUMMY
  //SYSUT2   DD     SYSOUT=(*,INTRDR)
  //SYSUT1   DD  DATA,DLM='$$'
  //ADANUC   JOB 5,'ADA-NUC',CLASS=A,MSGCLASS=X,REGION=6M
  //%STEP    EXEC PGM=ADARUN
  //STEPLIB  DD DSN=ADABAS.Vvrs.LOAD,DSIP=SHR
  //*
  //DDASSOR1 DD DSN=ADABAS.Vvrs.ASSOR1,DISP=SHR
  //DDDATAR1 DD DSN=ADABAS.Vvrs.DATAR1,DISP=SHR
  //DDWORKR1 DD DSN=ADABAS.Vvrs.WORKR1,DISP=SHR
  //*
  //DDRLOGR1 DD DSN=ADABAS.Vvrs.RLOGR1,DISP=SHR
  //*
  //DDPLOGR1 DD DSN=ADABAS.Vvrs.PLOGR1,DISP=SHR
  //*
  //SYSUDUMP DD SYSOUT=X
  //DDPRINT  DD SYSOUT=X
```

```
  //DDCARD   DD *
  ADARUN PROG=ADANUC
  ADARUN UEX2=UEX2
  ADARUN NPLOG=2,PLOGDEV=dddd
  ADARUN PLOGSIZE=size
  ADARUN MODE=MULTI
  ADARUN DATABASE=yyyyy
  ADARUN SVC=xxx
  ADARUN UTIONLY=YES
  $$                                          (internal reader delimiter)
  //CHKDB    EXEC PGM=ADARUN                  (check for an active nucleus)
  //STEPLIB DD  DSN=ADABAS.Vvrs.LOAD,DISP=SHR
  //SYSOUT  DD SYSOUT=*
  //DDPRINT DD SYSOUT=*
  //DDDRUCK DD SYSOUT=*
  //DDCARD  DD *
  ADARUN MODE=MULTI,PROGRAM=ADARAI,
  ADARUN DBID=yyyyy,SVC=xxx,DE=dddd
  //DDKARTE DD *
  ADARAI CHKDB ACTIVE
  //*
%%JCL-ENDNUC
  //*
  //* ADADBS END NUCLEUS
  //*
  //%STEP    EXEC PGM=ADARUN
  //DDASSOR1 DD DSN=ADABAS.Vvrs.ASSOR1,DISP=SHR
  //DDDATAR1 DD DSN=ADABAS.Vvrs.DATAR1,DISP=SHR
  //DDWORKR1 DD DSN=ADABAS.Vvrs.WORKR1,DISP=SHR
  //DDTEMPR1 DD DSN=ADABAS.Vvrs.TEMPR1,DISP=SHR
  //DDSORTR1 DD DSN=ADABAS.Vvrs.SORTR1,DISP=SHR
  //DDRLOGR1 DD DSN=ADABAS.Vvrs.RLOGR1,DISP=SHR
  //DDDRUCK  DD SYSOUT=X
  //DDPRINT  DD SYSOUT=X
  //DDCARD   DD *
  ADARUN MODE=MULTI,PROG=ADADBS,DBID=yyyyy,DE=dddd,SVC=xxx
  //DDKARTE  DD *
  ADABAS OPERCOM ADAEND
  //CHKDB    EXEC PGM=ADARUN                  (check for an inactive nucleus)
  //STEPLIB DD  DSN=ADABAS.Vvrs.LOAD,DISP=SHR
  //SYSOUT  DD SYSOUT=*
  //DDPRINT DD SYSOUT=*
  //DDDRUCK DD SYSOUT=*
  //DDCARD  DD *
  ADARUN MODE=MULTI,PROGRAM=ADARAI,
  ADARUN DBID=yyyyy,SVC=xxx,DE=dddd
```

```
//DDKARTE DD *
ADARAI CHKDB INACTIVE
//*
```

VM/ESA or z/VM

ADARAI is **not** currently supported by either the ADAMAINT EXEC (database maintenance) or the INSTADA EXEC (database installation). The information needed to run ADARAI must be added manually to the EXECs, CONTROL files, and so on.

Add the required information as follows:

1. Update ADARAI PARM to contain the correct database ID (the default ID is “00001”).
2. Add MULTI write LINK commands to the ADARAI minidisks in the PROFILE EXEC of the DBA virtual machine. The ADARAI minidisks such as DDRLOGR1 must be defined in the CP directory of the Adabas nucleus virtual machine.
3. Complete the ADFdbid EXEC with a DATADEF statement for DDRLOGR1. This statement should always be defined for every utility and for the nucleus. The dataset must be CMS-formatted and reserved with the defined file name and file types.

Dataset	**DD Name**	**Storage**	**More Information**
Associator	DDASSORn	disk	
Data Storage	DDDATARn	disk	Required for RECOVER
Work	DDWORKR1	disk	Required for RECOVER
Datra protection log	DDPLOGRn	tape/disk	Required for RECOVER
Recovery log (RLOG)	DDRLOGR1	disk	
Job stream input	DDJCLIN	disk	Required for RECOVER
Recovery job output	DDJCLOUT	disk	Required for RECOVER
ADARAI messages	DDDRUCK	disk/terminal/ printer	*Messages and Codes*
ADARUN messages	DDPRINT	disk/terminal/ printer	*Messages and Codes*
ADARUN parameters	DDCARD	disk/terminal/ reader	*Operations Manual*
ADARAI parameters	DDKARTE	disk/terminal/ reader	

JCL Examples (VM/ESA or z/VM)

Start Recovery Logging (ADARAI PREPARE)

```
DATADEF DDASSOR1,DSN=ADABASVv.ASSO,VOL=ASSOV1
DATADEF DDRLOGR1,DSN=ADABASVv.RLOG1,VOL=RLOGV1
DATADEF DDDRUCK,DSN=ADARAI.DDDRUCK,MODE=A
DATADEF DDPRINT,DSN=ADAFRM.DDPRINT,MODE=A
DATADEF DDCARD,DSN=RUNRAI.CONTROL,MODE=A
DATADEF DDKARTE,DSN=ADARAI.CONTROL,MODE=A
ADARUN
```

Contents of RUNRAI CONTROL A1:

```
ADARUN PROG=ADARAI,DEVICE=dddd,DB=99,MODE=MULTI
```

Contents of ADARAI CONTROL A1:

```
ADARUN PREPARE RLOGSIZE=5,RLOGDEV=eeee,MINGENS=5
```

List The RLOG (ADARAI LIST)

```
DATADEF ASSOR1,DSN=ADABASVv.ASSO,VOL=ASSOV1
DATADEF RLOGR1,DSN=ADABASVv.RLOG1,VOL=RLOGV1
DATADEF DDDRUCK,DSN=ADARAI.DDDRUCK,MODE=A
DATADEF DDPRINT,DSN=ADARAI.DDPRINT,MODE=A
DATADEF DDCARD,DSN=RUNRAI.CONTROL,MODE=A
DATADEF DDKARTE,DSN=ADARAI.CONTROL,MODE=A
```

Contents of RUNRAI CONTROL A1:

```
ADARUN PROG=ADARAI,DEVICE=dddd,DB=yyyyy,MODE=MULTI
```

Contents of ADARAI CONTROL A1:

```
ADARUN LIST
```

VSE/ESA

The following functions are available for use with ADARAI on a VSE/ESA system:

Function	Action
CHKDB	check the database status
DISABLE	disable the recovery log (RLOG)
LIST	list the RLOG contents
PREPARE	start the RLOGs
REMOVE	remove the RLOG

The ADARAI RECOVER function used to rebuild the job stream is not currently supported on a VSE/ESA system.

File	Symbolic Name	Storage	Logical Unit	More Information
Associator	ASSORn	disk	See note 1	
Data Storage	DATARn	disk	See note 1	Required for RECOVER
Work	WORKR1	disk		Required for RECOVER
Recovery log (RLOG)	RLOGR1	disk		
Data protection log	PLOGRn	tape disk		Required for RECOVER
Job stream input	JCLIN	SYSIPT		Required for RECOVER
Recovery job output	JCLOUT	SYSPCH		Required for RECOVER
ADARAI messages		printer	SYS009	*Messages and Codes*
ADARUN messages		printer	SYSLST	*Messages and Codes*
ADARUN parameters	— CARD CARD	reader tape disk	SYSRDR SYS000 See note 1	
ADARAI parameters		reader	SYSIPT	

Note:
Any programmer logical unit may be used.

Example JCS (VSE/ESA)

Start Recovery Logging (ADARAI PREPARE)

```
// ASSGN SYS009,00F
// EXEC PROC=ADAVvFIL
// EXEC PROC=ADAVvLIB
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADARAI,SVC=xxx,DEVICE=dddd,DB=yyyyy,MODE=MULTI
/*
ADARAI PREPARE RLOGSIZE=5,MINGENS=5
/*
```

See appendix B for descriptions of the VSE/ESA procedures (PROCs).

List the RLOG (ADARAI LIST)

```
// ASSGN SYS009,00F
// EXEC PROC=ADAVvFIL
// EXEC PROC=ADAVvLIB
// ASSGN SYS000,SYSIPT
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADARAI,SVC=xxx,DEVICE=dddd,DB=yyyyy,MODE=MULTI
/*
ADARAI LIST
/*
```

17

ADAREP : REPORT

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.
- detailed information (optionally by file) that includes all summary information plus MINISN/MAXISN settings, detailed space information, creation and last use date/time, field definition fable (FDT) contents, and general or extended checkpoint file information.

Syntax

```
ADAREP [REPORT] [ {CPEXLIST [,OFFSET={column-number | 63} ]
                  | CPLIST }
                      [FROMDATE=yyyymmdd] [TODATE=yyyymmdd]
                      [FROMSESSION=session] [TOSESSION=session] ]
                  [ {FILE={file-list|all-files}[,LAYOUT=1] | NOFILE } ]
                  [ {LIMCOUNT | NOCOUNT } ]
                  [NOFDT]
                  [ {NOLGLIST | NOPHLIST | NOSTD } ]
                  [NOUSERABEND]
                  [SAVETAPE [PLOGNUM=protection-log-number
                      [ {SYN1 | SYN4}=PLOG-block-number] ] ]
```

Optional Parameters

ADAREP can be specified alone to retrieve a database status report. You can optionally customize the report by added parameter values.

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).

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 report table is divided into two 80-column subtables suitable for terminal display.

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.

NOUSERABEND : Termination without ABEND

When an error is encountered while the function is running, the utility 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.

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.

PLOGNUM : Protection Log Number

PLOGNUM specifies the number of the nucleus protection log used during the ADASAV save operation. The parameter is required if the supplied save tape was created by an Adabas version 5.1 online save operation. For tapes created using Adabas version 5.2 or above, the PLOGNUM is supplied on the tape. Specifying PLOGNUM for such tapes **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** on page 190.

CPLIST/CPEXLIST cannot be specified with SAVETAPE. NOCOUNT must be specified with SAVETAPE because ADAREP does not count the number of records in a file on a save tape. If CPLIST/CPEXLIST is specified or NOCOUNT is not, ADAREP prints a warning message, changes these options internally, and continues processing.

SYN1/SYN4 : Beginning Block Number

SYN1/SYN4 specifies the block number containing the SYN1/SYN4 checkpoint at which the corresponding save operation began. The parameter is required if the supplied save tape was created by an Adabas version 5.1 online save operation. For tapes created using ADASAV version 5.2 or above, the SYN1/SYN4 checkpoint information is supplied on the tape. Specifying SYN1/SYN4 for such tapes **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.

Processing Save Tape Input

If the SAVETAPE keyword is specified, the report is printed from a save tape.

The save tape

- must have been created using ADASAV version 5.1 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 DD/SAVE 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 block (GCB), 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.

Notes:

1. *Adabas version 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 versions of Adabas prior to version 7.2 back to version 5.1 is still reported.*
2. *A save tape created using ADASAV version 5.1 does not contain the mirror table, the mirror BUB, or the FST. At the corresponding location in the report, a message appears indicating that these blocks are not available.*

Supplying Protection Log Input

If an online save tape is used, the corresponding protection log may optionally be specified as a DD/PLOG sequential input file:

- If DD/PLOG is supplied, ADAREP scans the protection log for FCB and FST blocks to ensure that it has the most recent versions.
- If DD/PLOG 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.

If the save tape you are using was created using ADASAV version 5.1, you must specify the parameters PLOGNUM and SYN1/SYN4 to indicate the protection log number and block number of the SYN1 or SYN4 checkpoint. For tapes created using ADASAV version 5.2 or above, this information is supplied on the tape. Specifying PLOGNUM or SYN1/SYN4 for such tapes **overrides** the information on the tape.

When DD/PLOG 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 DD/SAVE and DD/PLOG input datasets, 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.

If the save tape was created using ADASAV version 5.3.2 or above, the VOLSER number is printed on the report. For save tapes created using earlier versions of ADASAV, 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 GCB.
- 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    Vvr                         *
*  DBID                 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 GCB 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.

Report Description

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 SYSLST (BS2000), DDDRUCK (z/OS or z/VM), or SYS009 (VSE) dataset, the following figures show examples of the report output displayed at a terminal. The following pages display sections in the order they appear in the report; a description of each part is provided on the same page or the facing page.

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        = 99
DATE LOADED             = yyyy-mm-dd
TIME LOADED             = hh:mm:ss
SYSTEM FILES            = 10
TRIGGER FILES           = 6
MAXIMUM NUMBER OF FILES = 15
NUMBER OF FILES LOADED  = 3
CURRENT LOG TAPE NUMBER = 48
RABNSIZE                = 3
RECOVERY AID            = NO
```

Additionally, if universal encoding support (UES) is enabled (UES=YES), the following encoding information is displayed:

```
UNIVERSAL ENCODING SUP. = YES
ALPHA FILE ENCODING     =   37
WIDE FILE ENCODING      = 4095
ALPHA ASCII ENCODING    =  437
WIDE USER ENCODING      =  950
```

If UES=NO, this information is suppressed.

Field	Explanation
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.
TIME LOADED	Time of day when the database was initially defined.
SYSTEM FILES	File numbers of Adabas system files.
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.
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.
CURRENT LOG TAPE NUMBER	Number of the most recent data protection log for 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.
UNIVERSAL ENCODING SUPPORT	Whether universal encoding support (UES) is active for the database.

Field	Explanation
ALPHA FILE ENCODING	Current file encoding set for alphanumeric (A) format fields in the database. Must be EBCDIC-compatible.
WIDE FILE ENCODING	Current file encoding set for wide-character (W) format fields in the database.
ALPHA ASCII ENCODING	Current user encoding set for alphanumeric (A) format fields in the database. Must be ASCII-compatible.
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.
EXTENTS IN BLK	The extents, listed by block range.

Column	Explanation
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.

Contents of the Database : General File Status

The next section contains information on the status of each file in the database:

```
***********************************************
*                                             *
* CONTENTS OF DATABASE  99 (EXAMPLE-DB      ) *          yyyy-mm-dd  hh:mm:ss
*                                             *
***********************************************


FILE    NAME            LOADED    TOP-ISN   MAX-ISN  EXTENTS PADDING
                                                        NUA D   A%  D%

   1  EMPLOYEES      1995-10-27      1107      2003   111 1   10  10
   2  VEHICLES       1995-10-27       773      2003   111 1   10  10
  10  CHECKPOINT     1995-10-27         3       667   111 1   10  10
```

The columns in this table provide the following information:

Column	Explanation
FILE	Adabas file number.
NAME	File name (see the ADALOD utility, NAME parameter).
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).

Column	Explanation
EXTENTS	Number of logical extents currently assigned to the normal index (N), upper index (U), address converter (A), and Data Storage (D). A maximum of 5 logical extents may be allocated to an element. If an element has been assigned 5 extents, 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.
PADDING	The block padding factor defined for the Associator (A) and Data Storage (D) (see the ADALOD utility, ASSOPFAC and DATAPFAC parameters).

File Options

The next section lists the file options that are active for each file in the database:

```
****************
* FILE OPTIONS *
****************
                           ADAM FILE
                           .  COUPLED FILE
                           .  .  ISNREUSE
                           .  .  .  DSREUSE
                           .  .  .  .  CIPHERED FILE
                           .  .  .  .  .  EXPANDED FILE
                           .  .  .  .  .  .  USERISN
                           .  .  .  .  .  .  .  NOACEXTENSION
                           .  .  .  .  .  .  .  .  MIXDSDEV
                           .  .  .  .  .  .  .  .  .  PGMREFRESH
                           .  .  .  .  .  .  .  .  .  .  MULTICLIENT FILE
                           .  .  .  .  .  .  .  .  .  .  .  INDEX COMPRESSED
FILE  NAME                 .  .  .  .  .  .  .  .  .  .  .  .
-----------------------------------------------------------------------------
   1  EMPLOYEES            .  C  I  D  .  .  .  .  .  .  .  .
   2  VEHICLES             .  C  .  D  .  .  .  .  .  .  .  .
  10  CHECKPOINT           .  .  .  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.
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.
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:

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

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.

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

      106    -   119         14 3380   DSST    00 ADA001
      120    -   120         01 3380    AC     10 ADA001
      121    -   121         01 3380    UI     10 ADA001
      122    -   131         10 3380    NI     10 ADA001
      132    -   134         03 3380    AC     01 ADA001
      135    -   164         30 3380    UI     01 ADA001
      165    -   264        100 3380    NI     01 ADA001
      265    -   267         03 3380    AC     02 ADA001
      268    -   287         20 3380    UI     02 ADA001
      288    -   327         10 3380    NI     02 ADA001
      328  -   28481      28154 3380  UNUSED   00 ADA001

       01     -   20         20 3380    DS     10 ADA002
       21    -   100         80 3380    DS     01 ADA002
      101    -   130         30 3380    DS     02 ADA002
      131  -   26991      26861 3380  UNUSED   00 ADA002
```

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.	
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 datasets 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:

```
************
* FILE  01 *    EMPLOYEES                                yyyy-mm-dd  hh:mm:ss
************

TOP-ISN            =      1,107        HIGHEST INDEX LEVEL =  3
MAX-ISN EXPECTED   =      2,003        PADDING FACTOR ASSO = 10%
RECORDS LOADED     =      1,107        PADDING FACTOR DATA = 10%
MIN-ISN            =          1        LENGTH OF CLIENT NR =  0
NUMBER OF UPDATES  =          0        ISNSIZE             =  3

MAX COMP REC LEN   =      4,734        DATE LOADED         =  1995-01-29
BLK/ADD DS  EXT    =          0        TIME LOADED         = 13:30:14
BLK/ADD UI  EXT    =          0        DATE OF LAST UPDATE =  1997-01-15
BLK/ADD NI  EXT    =          0        TIME OF LAST UPDATE = 23:45:10

FILE ALPHA CODE    =        500
FILE WIDE CODE     =      4,095
USER WIDE CODE     = DB DEFAULT
```

The following information is provided:

Field	Explanation
TOP-ISN	Highest ISN currently used in the file.
MAX-ISN EXPECTED	Highest ISN planned for the file. See the ADALOD utility, MAXISN parameter.
RECORDS LOADED	Number of records currently contained in the file.
MINIMUM 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.
MAX COMP REC LENGTH	Maximum compressed record length permitted for the file. See the ADALOD utility, MAXRECL parameter.

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 UI EXT	Maximum number of blocks which may be allocated for each secondary upper index extent. See the ADALOD utility, MAXUI 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.
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.
USER WIDE CODE	Current user 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.
PADDING FACTOR ASSO	Associator padding factor. See the ADALOD utility, ASSOPFAC parameter or the ARDORD utility, REORASSO and REORFASSO functions.
PADDING FACTOR DATA	Data Storage padding factor. See the ADALOD utility, DATAPFAC parameter or the ADAORD utility, REORDATA and REORFDATA functions.
LENGTH OF CLIENT NR	Length of the owner ID for a multiclient file.
ISNSIZE	Whether the file contains 3-byte or 4-byte ISNs.
DATE LOADED	Date the file was loaded.
TIME LOADED	Time the file was loaded.
DATE OF LAST UPDATE	Date the file was last changed.
TIME OF LAST UPDATE	Time the file was last changed.

Options

File-option settings for the file are displayed next:

```
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
MULTICLIENT FILE    NO
PRIVILEGED USAGE    NO
ONLINE INVERT       NONE
INDEX COMPRESSED    NO

ADABAS VERSION NEEDED FOR THIS FILE: V71 OR LATER
```

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.

Field	Indicates . . .
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.

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 changes 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:

```
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
DSST I 3380  2004 I          1      0I       106      106I                  I
 AC  I 3380  2004 I          3      0I       132      134I                  I
 UI  I 3380  2004 I         30      0I       135      164I        17       0I
 NI  I 3380  2004 I        100      0I       165      264I        24       0I
     I            I                  I                  I                  I
 DS  I 3380  4820 I         80      0I        21      100I        31       0I
     I            I                  I                  I                  I
-------------------------------------------------------------------------------
```

The space allocations table provides the following information:

Column	Explanation	
LIST TYPE	The database component:	
	AC	address converter
	NI	normal index
	UI	upper index
	DS	Data Storage
	DSF	File-specific delta save logging area
	DSST	Data Storage space table
	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.

```
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
```

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

Field	Explanation	
OPTIONS	DE	Descriptor
	FI	Fixed storage
	LA	Long alphanumeric
	MU	Multiple-value field
	NC	Null/not counted
	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, subdescriptors, 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   GA0TB1
          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   GA0TB1
          RESTRUCT
```

The columns in this table provide the following information:

Column	Explanation
CP–NAME	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: 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 A checkpoint from Adabas Online System (SYSAOS) or ADADBS with four exceptions from the nucleus. The function identified by this checkpoint is implemented without user intervention during regeneration.

Column	Explanation	
		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	Indicates that a volume ID changed during sequential write to a dataset is being closed.
	SYNX	A checkpoint from a utility requiring exclusive control (EXU) of one or more files.
	SYN1	A checkpoint made at the beginning of online ADASAV execution (SAVE database function).
	SYN2	A checkpoint made at the end of online ADASAV execution (SAVE database function).
	SYN4	A checkpoint made at the beginning of online ADASAV execution (SAVE files operation).
	SYN5	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	ET user
	EXF	exclusive-file-control user or utility (privileged user)
	EXU	exclusive-file-update user or utility
	UTI	utility-update-control utility (privileged user)
	UTS	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.	

Column	Explanation
VOLSER–NUMBER	The volume serial number of the sequential protection (DD/SIBA) 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	SYNS	ADANUC	Written by the nucleus at the start of a second or subsequent Adabas Cluster Services or Adabas Parallel Services nucleus session.
01	SYNF	User/Util.	User/utility session OPEN with files used in EXF (exclusive use) mode.
01	SYNX	EXU user	EXU user open.
02	SYNV	ADANUC	VOLSER entry. Written at volume switch on DD/SIBA and at the end of the session if sequential logging is used.
03	SYNF	User/Util.	Close checkpoint for an EXF user.
03	SYNX	EXU	Close checkpoint for an EXU user.
05	SYNP	ADASAV	SAVE file(s)—start of operation
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 DD/SAVE and at SAVE-operation end.
0A	SYNV	ADASAV	SAVE database, VOLSER entry. Written at volume switch on DD/SAVE and at SAVE-operation end.
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)

Type	Name	Originator	Description
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
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
27	SYNV	ADARES	COPY sequential protection log
28	SYNP	ADARES	PLCOPY function successfully completed
28	SYNV	ADARES	PLCOPY dual or multiple protection log
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

Type	Name	Originator	Description
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
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
5B	SYNS	ADARES	Write refreshed statistics (command, file, and thread usage; DRES and DSTAT)
5C	SYNS	SYSAOS	CHANGE default field format
5D	SYNS	SYSAOS	Change file encoding
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
68	SYNS	SYSAOS	Set USERISN on/off

Type	Name	Originator	Description
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
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.

JCL/JCS Requirements and Examples

This section describes the job control information required to run ADAREP with BS2000, OS/390 or z/OS, VM/ESA or z/VM, and VSE/ESA systems and shows examples of each of the job streams.

BS2000

Dataset	Link Name	Storage	More Information
Associator	DDASSORn	disk	
Data Storage	DDDATARn	disk	
Save tape	DDSAVE	tape/disk	Only with SAVETAPE
Protection log	DDPLOG	tape/disk	Option with online save tape
ADARUN parameters	SYSDTA/ DDCARD		*Operations Manual*
ADAREP parameters	SYSDTA/ DDKARTE		
ADARUN messages	SYSOUT DDPRINT		*Messages and Codes*
ADAREP messages	SYSLST DDDRUCK		*Messages and Codes*

ADAREP JCL Example (BS2000)

In SDF Format:

```
/.ADAREP LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK * A D A R E P ALL FUNCTIONS
/REMARK *
/ASS-SYSLST L.REP
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDASSOR1,ADAyyyyy.ASSO,SHARE-UPD=YES
/SET-FILE-LINK DDDATAR1,ADAyyyyy.DATA,SHARE-UPD=YES
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADAREP,DB=yyyyy,IDTNAME=ADABAS5B
ADAREP CPLIST
/LOGOFF SYS-OUTPUT=DEL
```

In ISP Format:

```
/.ADAREP LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK * A D A R E P ALL FUNCTIONS
/REMARK *
/SYSFILE SYSLST=L.REP
/FILE ADA.MOD,LINK=DDLIB
/FILE ADAyyyyy.ASSO ,LINK=DDASSOR1,SHARUPD=YES
/FILE ADAyyyyy.DATA ,LINK=DDDATAR1,SHARUPD=YES
/EXEC (ADARUN,ADA.MOD)
ADARUN PROG=ADAREP,DB=yyyyy,IDTNAME=ADABAS5B
ADAREP CPLIST
/LOGOFF NOSPOOL
```

OS/390 or z/OS

Dataset	DD Name	Storage	More Information
Associator	DDASSORn	disk	
Data Storage	DDDATARn	disk	
Save tape	DDSAVE	tape/disk	Only with SAVETAPE
Protection log	DDPLOG	tape/disk	Option with online save tape
ADARUN parameters	DDCARD	reader	*Operations Manual*
ADAREP parameters	DDKARTE	reader	
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADAREP messages	DDDRUCK	printer	*Messages and Codes*

ADAREP JCL Example (OS/390 or z/OS)

All Functions

```
//ADAREP    JOB
//*
//*   ADAREP : ALL FUNCTIONS
//*
//REP      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
//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 MVSJOBS dataset 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.Vvrs.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 MVSJOBS dataset for this example.

VM/ESA or z/VM

Dataset	DD Name	Medium	Additional Information
Associator	DDASSORn	disk	
Data Storage	DDDATARn	disk	
Save tape	DDSAVE	tape/disk	Only with SAVETAPE
Protection log	DDPLOG	tape/disk	Option with online save tape
ADARUN parameters	DDCARD	disk/terminal/reader	*Operations Manual*
ADAREP parameters	DDKARTE	disk/terminal/reader	
ADARUN messages	DDPRINT	disk/terminal/printer	*Messages and Codes*
ADAREP messages	DDDRUCK	disk/terminal/printer	*Messages and Codes*

ADAREP Example (VM/ESA or z/VM)

```
DATADEF DDASSOR1,DSN=ADABASVv.ASSO,VOL=ASSOV1
DATADEF DDDATAR1,DSN=ADABASVv.DATA,VOL=DATAV1
DATADEF DDPRINT,DSN=ADAREP.DDPRINT,MODE=A
DATADEF DUMP,DUMMY

DATADEF DDDRUCK,DSN=ADAREP.DDDRUCK,MODE=A
DATADEF DDCARD,DSN=RUNREP.CONTROL,MODE=A
DATADEF DDKARTE,DSN=ADAREP.CONTROL,MODE=A
ADARUN
```

Contents of RUNREP CONTROL A1

```
ADARUN PROG=ADAREP,DEVICE=dddd,DB=yyyyy
```

Contents of ADAREP CONTROL A1

```
ADAREP CPLIST
```

VSE/ESA

Dataset	Symbolic Name	Storage	Logical Unit	More Information
Associator	ASSORn	disk		
Data Storage	DATARn	disk		
Save tape	SAVE	tape disk	SYS010 see note	Only with SAVETAPE
Protection log	PLOG	tape disk	SYS011 see note	Option with online save tape
ADARUN parameters	SYSRDR CARD CARD	reader tape disk	SYSRDR SYS000 see note	*Operations Manual*
ADAREP parameters		reader	SYSIPT	
ADARUN messages		printer	SYSLST	*Messages and Codes*
ADAREP report		printer	SYS009	

Note:
Any programmer logical unit may be used.

ADAREP JCS Example (VSE/ESA)

See appendix B for descriptions of the VSE/ESA procedures (PROCs).

All Functions

```
* $$ JOB JNM=ADAREP,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADAREP
*       ALL FUNCTIONS
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADAREP,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADAREP    CPEXLIST
/*
/&
* $$ EOJ
```

Refer to member ADAREP.X for this example.

Report from a Save Tape

```
* $$ JOB JNM=ADAREPS,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADAREPS
*       REPORT FROM A SAVE TAPE
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// ASSGN SYSTEN,TAPE
// PAUSE MOUNT LOAD SAVE FILE ON TAPE cuu
// TLBL SAVE,'EXAMPLE.DByyyyy.SAVE'
// MTC REW,SYS010
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADAREP,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADAREP    REPORT SAVETAPE,NOCOUNT
/*
/&
* $$ EOJ
```

Refer to member ADAREPS.X for this example.

18

ADARES : RESTART

Functional Overview

The ADARES utility performs functions related to database recovery.

Function	Action	Page
BACKOUT	remove all the updates applied between two checkpoints. The checkpoints used are normally the result of a non-synchronized checkpoint command (C1) but may also be a synchronized checkpoint. The complete database may be included in the backout process, or backout may be limited to selected files.	230
	BACKOUT using the sequential protection log (SIBA)	231
	BACKOUT DPLOG using dual protection logging or MPLOG using multiple protection logging	238
CLCOPY	copy a command log dataset from disk to a sequential dataset. This function is necessary only if multiple command logging is in effect for an Adabas session.	246
COPY	copy a sequential Adabas protection log dataset. This function should be executed if the Adabas session in which the sequential protection log dataset was created was terminated abnormally.	248
MERGE CLOG	merge command log datasets from multiple nuclei in a cluster environment into a single log file.	252
PLCOPY	copy a protection log dataset from disk to a sequential dataset. This function is necessary only if multiple protection logging is in effect for an Adabas session.	253
REGENERATE	reapply all the updates performed between two user-specified checkpoints. The checkpoints specified may either be the result of a nonsynchronized checkpoint command (C1) or a synchronized checkpoint. All files may be included in the regenerate process or it may be limited to one or more files. This function is most often used after the database (or one or more files) has been restored to a previous status with the RESTORE or RESTONL function of the ADASAV utility.	256

Function	Action	Page
REPAIR	repair one or more blocks in Data Storage that, for any reason, have become unusable. The most recent save tape of the database and any protection log tapes created thereafter are used as input to this function. **Use only with guidance from your Software AG technical support representative.**	266

Information about Using ADARES

1. The functions BACKOUT (except BACKOUT DPLOG or MPLOG), REGENERATE, and REPAIR require a sequential dataset 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/SIIN input file contains the sessions in ascending order by session number; gaps in the session number sequence exist only for those sessions representing save dataset 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 session number with PLOGNUM, or with FROMPLOG without TOPLOG; to specify a **range** of sessions, specify FROMPLOG and TOPLOG.

4. The ADARES COPY function accepts ADASAV output save (DD/SAVEn) 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 DD/CARD 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, Software AG recommends 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

For recovery processing, all protection log datasets (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 datasets.

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 datasets and a sample user exit 2 (USEREX2P) is provided. See page 281. It is not necessary to remove the PLOG DD statements, however. If they remain, they are ignored.

By default, dual/multiple command log datasets (CLOGs) can be copied to a sequential dataset for each nucleus using the ADARES CLCOPY function, but the resulting datasets are not then automatically merged across the cluster into a single CLOG dataset 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 datasets 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 datasets.

Existing CLCOPY jobs must be modified to include the intermediate datasets. A sample CLCOPY job ADARESMC is provided that illustrates the necessary addition of the intermediate datasets. See page 284. 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 dataset, the job ‘partially’ merges the dataset.

Intermediate Datasets

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 datasets. Records beyond this point are written to an ‘intermediate’ dataset, which must be supplied as input to the subsequent merge. A cross-check ensures that the correct intermediate dataset 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 datasets. When intermediate datasets are used for both CLCOPY and PLCOPY jobs, the dataset names for each must be unique so that they are not overwritten.

- PLCOPY example:

```
//MERGINI DD DISP=SHR,DSN=EXAMPLE.PINTERI
//MERGINO DD DISP=SHR,DSN=EXAMPLE.PINTERO
```

- CLCOPY example:

```
//MERGINI DD DISP=SHR,DSN=EXAMPLE.CINTERI
//MERGINO 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 datasets have been supplied on the session start-up 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 dataset to expect as input. It checks the header information in both datasets to determine which to use for input and which for output.

The following checks are made to ensure that the intermediate dataset has been supplied correctly:

1. The DBID is stored in the intermediate dataset header and must match the DBID in the log.
2. The log number is stored in the intermediate dataset header and must either match or be one less than the current number from the log dataset.
3. The STCK in the intermediate dataset 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 dataset 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
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.

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 dataset or to a dual/multiple protection log dataset (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 dataset. This dataset can be used as input to ADARES BACKOUT or REGENERATE.

Software AG does 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 OS/390 or z/OS when processing compressed data. Also, the BACKOUT function is not supported for compressed data on VSE/ESA and VM/ESA systems.

BACKOUT : Back Out Updates Using the Sequential Protection Log (SIBA)

The BACKOUT function removes all the updates applied between two specified checkpoints. Both checkpoints must be contained on the sequential protection log input dataset.

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 dataset is reached. You can limit the extent of BACKOUT processing using the TOCP parameter.

By default, all files in the specified input dataset are included in the BACKOUT processing. You have to 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 dataset 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.

```
ADARES BACKOUT {PLOGNUM=protection-log-number
                | FROMPLOG=start-session [,TOPLOG=stop-session]}
                [EXCLUDE=file-list]
                [FILE=file-list [CONTINUE]]
                [FROMCP=checkpoint-name [,FROMBLK=checkpoint-block] ]
                [IGNORECOUPLE]
                [IGNOREEXP]
                [MTR={YES| NO} [NPCALLS=maximum-number-of-parallel-calls] ]
                [NOAUTOBACKOUT]
                [NOUSERABEND]
                [PARALLELREAD]
                [PLOGDBID={alternate-log-dbid | ADARUN-dbid}]
                [TEST]
                [TOCP=checkpoint-name [,TOBLK=checkpoint-block] ]
```

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 (DD/SIIN) 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 datasets 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.

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.

Note:
NOAUTOBACKOUT cannot be specified in single-user mode.

NOUSERABEND : Terminate without ABEND

When an error is encountered while the function is running, the utility 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.

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.

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 that indicate inconsistencies.

Note:
During ADARES operation with PARALLELREAD, temporary differences between the Associator and Data Storage may cause nucleus responses 113 or 199 to occur.

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.

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.

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 (DD/SIIN) 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.

BACKOUT DPLOG or MPLOG : Back Out Updates Using the Dual or Multiple Protection Log

The BACKOUT {DPLOG | MPLOG} function removes all the updates applied between two checkpoints contained on the same Adabas dual or multiple protection log dataset, respectively.

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 dataset and switch to a new one. The new protection log dataset 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 dataset. 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 dataset. The closed dataset contains all information required for BACKOUT DPLOG or MPLOG.

4. Run ADARES PLCOPY.

   Copy the content of the protection log dataset to a sequential dataset. 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 dataset 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. Software AG recommends that you back out using the entire contents of one protection log dataset.

During backout, the nucleus writes new protection log information to the protection log dataset currently available. This is the only dataset 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 dataset has been copied with ADARES PLCOPY, and then resumes the backout run.

By default, all files in the specified input dataset 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 datasets 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 dataset 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.

```
ADARES BACKOUT {DPLOG  [DUALPLD={device-type|ADARUN-device}]|
                MPLOG  [PLOGDEV={device-type|ADARUN-device}]}
               [EXCLUDE=file-list]
               [FILE=file-list [CONTINUE]  ]
               [FROMCP=checkpoint-name  [,FROMBLK=checkpoint-block] ]
               [IGNORECOUPLE]
               [IGNOREEXP]
               [MTR={YES | NO} [NPCALLS=maximum-number-of-parallel-calls] ]
               [NOAUTOBACKOUT]
               [NOUSERABEND]
               [PARALLELREAD]
               [PLOGDBID={alternate-log-dbid|ADARUN-dbid}]
               [TEST]
               [TOCP=checkpoint-name  [,TOBLK=checkpoint-block]  ]
```

Essential Parameter

DPLOG | MPLOG : Dual or Multiple PLOG Source

DPLOG indicates that a dual protection log dataset is to be used as input; MPLOG indicates that a multiple protection log dataset 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 datasets 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 datasets; PLOGDEV specified the device type used for the multiple protection log datasets. 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.

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.

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.

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.

Note:
NOAUTOBACKOUT cannot be specified in single-user mode.

NOUSERABEND : Terminate without ABEND

When an error is encountered while the function is running, the utility 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.

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.

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 that indicate inconsistencies.

Note:
During ADARES operation with PARALLELREAD, temporary differences between the Associator and Data Storage may cause nucleus responses 113 or 199 to occur.

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.

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.

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 dataset.

1. Switch to a new PLOG.
2. Run the user session creating PLOG data on the new PLOG dataset.
3. Close the PLOG dataset. User exit 2 submits a job which copies the contents of the PLOG dataset just closed.
4. Perform a BACKOUT from that PLOG for the files 20 and 21 up to the beginning of the PLOG.

CLCOPY : Copy Dual Command Log

The CLCOPY function is used only if dual logging of command information was specified for the Adabas session. This function copies the dataset that has the earlier time stamp to a sequential dataset. Once the CLCOPY function is completed successfully, the copied dataset is marked as empty. This function may, therefore, be used only once for any given dataset.

The CLCOPY function is not allowed in single-user mode.

```
ADARES CLCOPY   [DUALCLD={device-type|ADARUN-device}]
                [NOUSERABEND]
                [OPENOUT]
                [TEST]
                [TWOCOPIES]
```

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 datasets. This parameter is required if the device type used for the command log dataset is different from that specified with the ADARUN DEVICE parameter.

NOUSERABEND : Terminate without ABEND

When an error is encountered while the function is running, the utility 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.

OPENOUT : Open DDSIAUS1/2 or SIAUS1/2 Output Datasets

The OPENOUT parameter indicates that the DD/SIAUS1/2 output datasets are to be opened by ADARES, even if no data is actually to be copied. Without OPENOUT, the sequential output datasets 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 datasets 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.

COPY : Copy a Sequential Protection Log or Save Tape

The COPY function copies an Adabas sequential protection log dataset. If the Adabas session that created the sequential protection log dataset was terminated abnormally, the COPY function must be executed before the dataset can be used as input to any other ADARES function.

ADARES COPY

- must be used to copy a data protection log dataset from disk to a tape dataset 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 datasets.
- also accepts ADASAV SAVE output (DD/SAVEn) 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 DD/SIIN job control file.
- may be executed any number of times for a given input dataset.

```
ADARES COPY    [ {PLOGNUM=protection-log-number
                | FROMPLOG=start-session [,TOPLOG=stop-session] } ]
               [NONUC]
               [NOUSERABEND]
               [OPENOUT]
               [RLOGDEV={device-type|ADARUN-device}] ]
               [TEST]
               [TWOCOPIES]
               [UTICPLIST]
```

The COPY function has special uses if you are using the Adabas Delta Save Facility. Refer to the *Adabas Delta Save Facility Manual* 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 Backout

FROMPLOG specifies the session number at which the specified ADARES function is to start. ADARES searches the PLOG input (DD/SIIN) file for the correct starting session.

NONUC : Ignore Nucleus Response

The NONUC parameter may be used to ignore the nucleus response code in cases where the Adabas nucleus is unable to perform an autorestart. ADARES then attempts to physically copy the input tape as far as possible.

This parameter should only be specified after all other corrective actions have failed; however, NONUC **must** be specified if the PLOG sequential input (DD/SIIN) file contains several nucleus sessions to be copied.

NOUSERABEND : Terminate without ABEND

When an error is encountered while the function is running, the utility 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.

OPENOUT : Open DDSIAUS1/2 or SIAUS1/2 Output Datasets

The OPENOUT parameter indicates that the DD/SIAUS1/2 output datasets are to be opened by ADARES, even if no data is actually to be copied. Without OPENOUT, the sequential output datasets 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 datasets are opened before the first input record is read.

PLOGNUM : Protection Log Number

The Adabas protection log number of the dataset 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 Dataset

The RLOGDEV parameter is used if the Adabas Recovery Aid (ADARAI) is active to specify a device-type for the recovery log (RLOG) dataset.

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 GCB of the database has 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 (DD/SIIN) 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 All Utility Checkpoints

The UTICPLIST parameter causes ADARES to select and print all utility 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.

MERGE CLOG : Merge Nucleus Cluster Command Logs

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 datasets are expected as input to the MERGE CLOG function; therefore, the ADARES CLCOPY function must be executed prior to the ADARES MERGE function.

The timestamp contained in the CLOGLAYOUT=5 is required for the proper merging of command logs records.

Essential Parameter

NUMLOG : Number of Command Log Datasets

The NUMLOG parameter is required: it specifies the number of command log datasets to be included in the merge process. The maximum number is 32.

PLCOPY : Copy Protection Log to Sequential Dataset

The PLCOPY function is used only if dual/multiple logging of protection information was specified for the Adabas session. This function copies the dataset that has the earlier time stamp to a sequential dataset. Once the PLCOPY function is successfully completed, the copied dataset is marked as empty. This function may, therefore, be used only once in an Adabas session for any given dataset.

The use of hardware compression (IDRC) is **not** recommended for protection log files. The ADARES BACKOUT function is not supported for hardware-compressed data on VSE/ESA and z/VM systems. 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.

```
ADARES PLCOPY  [PLOGDEV={device-type|ADARUN-device}]
               [NOPPT]
               [NOUSERABEND]
               [OPENOUT]
               [RLOGDEV={device-type|ADARUN-device}]
               [TEST]
               [TWOCOPIES]
               [UTICPLIST]
```

The PLCOPY function has special uses if you are using the Adabas Delta Save Facility. Refer to the *Adabas Delta Save Facility Manual* 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 datasets. This parameter is required if the device type used for the dual/multiple protection log dataset is different from that specified with the ADARUN DEVICE parameter.

NOPPT (Clustered Nucleus Environments Only)

The parallel participant table (PPT) tells ADARES PLCOPY which datasets to copy. If the PPT is destroyed, the ADARES NOPPT function allows the DBA to specify the PLOG datsets that are to be copied and merged.

If ADARAI is used, the PLOG datasets 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 datasets 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 DD/PLOG datasets are to be supplied with JCL.

Warning:
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 dataset; and
- the correct input protection logs from all nuclei in the form of DD/PLOG01–nn.

Warning:
Without the PPT, ADARES cannot perform any extensive validations on the input datasets.

NOUSERABEND : Terminate without ABEND

When an error is encountered while the function is running, the utility 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.

OPENOUT : Open DDSIAUS1/2 or SIAUS1/2 Output Datasets

The OPENOUT parameter indicates that the DD/SIAUS1/2 output datasets are to be opened by ADARES, even if no data is actually to be copied. Without OPENOUT, the sequential output datasets 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 datasets are opened before the first input record is read.

RLOGDEV : Device Type for RLOG Dataset

The RLOGDEV parameter is used if the Adabas Recovery Aid (ADARAI) is active to specify a device-type for the recovery log (RLOG) dataset.

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 GCB of the database has 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.

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 All Utility Checkpoints

The UTICPLIST parameter causes ADARES to select and print all utility 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

REGENERATE : Regenerate Updates

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 (dataset size)
DECREASE (dataset size)	RECOVER

Syntax

```
ADARES REGENERATE  {PLOGNUM=protection-log-number
                   | FROMPLOG=start-session [,TOPLOG=stop-session] }
                   [ALLOCATION={FORCE | NOFORCE}]
                   [EXCLUDE=file-list]
                   [FILE=file-list [,CONTINUE] []
                   [FROMCP={checkpoint-name
                   | 'SYNS,INCLUDE'} [FROMBLK=checkpoint-block] ]
                   [IGNORECOUPLE]
                   [IGNOREEXP]
                   [MTR={YES| NO} [NPCALLS=max-number-of-parallel-calls]  ]
                   [NOAUTOBACKOUT]
                   [NOUSERABEND]
                   [PARALLELREAD]
                   [PLOGDBID={alternate-log-dbid|ADARUN-dbid} ]
                   [RAID]
                   [TEST]
                   [TOCP={checkpoint-name | 'SYNS,INCLUDE'}
                         [AUTOBACKOUT] [TOBLK=checkpoint-block] ]
```

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.

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 datasets 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 on page 260.

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 dataset 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.

Note:
NOAUTOBACKOUT cannot be specified in single-user mode.

NOUSERABEND : Terminate without ABEND

When an error is encountered while the function is running, the utility 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.

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.

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 that indicate inconsistencies.

Note:
During ADARES operation with PARALLELREAD, temporary differences between the Associator and Data Storage may cause nucleus responses 113 or 199 to occur.

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).

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 on page 260.

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.

REPAIR : Repair Data Storage Blocks

Warning:
The REPAIR function can cause data loss if not used correctly. It should only be used with guidance from your Software AG 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/SIIN input must be concatenated in the following sequence:
   - ADASAV SAVE (DD/SAVEn) output;
   - protection log.

Syntax

```
ADARES REPAIR   DSRABN={rabn | rabn – rabn }
                [FILE={locked-file-list|entire-database}]
                [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 an error is encountered while the function is running, the utility 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.

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.

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       764224
COMMANDS PROCESSED                302273
TRANSACTIONS PROCESSED             55045
NUMBER OF ADABAS CALLS             56450
MAXIMUM CALLS IN PARALLEL             71
AVERAGE CALLS IN PARALLEL             43
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

JCL/JCS Requirements and Examples

This section describes the job control information required to run ADARES with BS2000, OS/390 or z/OS, VSE/ESA, and VM/ESA or z/VM systems and shows examples of each of the job streams.

Notes:

1. *The DD/SIAUS1/2 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 chapter.*
2. *When running with the optional Recovery Aid (ADARAI), all temporary datasets must also be cataloged in the job control.*

BS2000

Dataset	Link 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.
Multiple command log	DDCLOGRn	disk	Input logs for CLCOPY function.
Sequential protection log	DDBACK	tape	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).
Data Storage	DDDATARn	disk	Required only for REGENERATE if FROMCP=SYN1 or SYN4.
Associator	DDASSORn	disk	
Recovery log (RLOG)	DDRLOGR1	disk	Required when using ADARAI.
ADARUN parameters	SYSDTA/ DDCARD		*Operations Manual*
ADARES parameters	SYSDTA/ DDKARTE		
ADARUN messages	SYSOUT/ DDPRINT		*Messages and Codes*
ADARES messages	SYSLST DDDRUCK		*Messages and Codes*

ADARES JCL Examples (BS2000)

Copy Dual/Multiple Command Log

In SDF Format:

```
/.ADARES LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK * A D A R E S COPY DUAL/MULTIPLE COMMAND LOG
/REMARK *
/DELETE-FILE ADAyyyyy.AUS1
/SET-JOB-STEP
/CREATE-FILE ADAyyyyy.AUS1,PUB(SPACE=(960,480))
/SET-JOB-STEP
/ASS-SYSLST L.RES.CLCO
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDASSOR1,ADAyyyyy.ASSO,SHARE-UPD=YES
/SET-FILE-LINK DDCLOGR1,ADAyyyyy.CLOGR1,SHARE-UPD=YES
/SET-FILE-LINK DDCLOGR2,ADAyyyyy.CLOGR2,SHARE-UPD=YES
/SET-FILE-LINK DDSIAUS1,ADAyyyyy.AUS1
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADARES,DB=yyyyy,IDTNAME=ADABAS5B
ADARES CLCOPY
/LOGOFF SYS-OUTPUT=DEL
```

In ISP Format:

```
/.ADARES LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK * A D A R E S COPY DUAL/MULTIPLE COMMAND LOG
/REMARK *
/SYSFILE SYSLST=L.RES.CLCO
/FILE ADA.MOD,LINK=DDLIB
/FILE ADAyyyyy.ASSO ,LINK=DDASSOR1,SHARUPD=YES
/FILE ADAyyyyy.CLOGR1 ,LINK=DDCLOGR1,SHARUPD=YES
/FILE ADAyyyyy.CLOGR2 ,LINK=DDCLOGR2,SHARUPD=YES
/FILE ADAyyyyy.AUS1 ,LINK=DDSIAUS1,SPACE=(960,480)
/EXEC (ADARUN,ADA.MOD)
ADARUN PROG=ADARES,DB=yyyyy,IDTNAME=ADABAS5B
ADARES CLCOPY
/LOGOFF NOSPOOL
```

Copy Sequential Protection Log

In SDF Format:

```
/.ADARES LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK * A D A R E S COPY SEQUENTIAL PLOG
/REMARK *
/DELETE-FILE ADAyyyyy.SIBA.COP1
/SET-JOB-STEP
/CREATE-FILE ADAyyyyy.SIBA.COP1,PUB(SPACE=(960,480))
/SET-JOB-STEP
/ASS-SYSLST L.RES.COPY
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDASSOR1,ADAyyyyy.ASSO,SHARE-UPD=YES
/SET-FILE-LINK DDSIIN,ADAyyyyy.SIBA
/SET-FILE-LINK DDSIAUS1,ADAyyyyy.SIBA.COP1
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADARES,DB=yyyyy,IDTNAME=ADABAS5B
ADARES COPY PLOGNUM=ppp
/LOGOFF SYS-OUTPUT=DEL
```

In ISP Format:

```
/.ADARES LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK * A D A R E S COPY SEQUENTIAL PLOG
/REMARK *
/SYSFILE SYSLST=L.RES.COPY
/FILE ADA.MOD,LINK=DDLIB
/FILE ADAyyyyy.ASSO ,LINK=DDASSOR1,SHARUPD=YES
/FILE ADAyyyyy.SIBA ,LINK=DDSIIN
/FILE ADAyyyyy.SIBA.COP1,LINK=DDSIAUS1,SPACE=(960,480)
/EXEC (ADARUN,ADA.MOD)
ADARUN PROG=ADARES,DB=yyyyy,IDTNAME=ADABAS5B
ADARES COPY PLOGNUM=ppp
/LOGOFF NOSPOOL
```

Copy Dual/Multiple Protection Log

In SDF Format:

```
/.ADARES LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK * A D A R E S COPY DUAL/MULTIPLE PROTECTION LOG
/REMARK *
/DELETE-FILE ADAyyyyy.AUS1
/SET-JOB-STEP
/CREATE-FILE ADAyyyyy.AUS1,PUB(SPACE=(960,480))
/SET-JOB-STEP
/DELETE-FILE ADAyyyyy.AUS2
/SET-JOB-STEP
/CREATE-FILE ADAyyyyy.AUS2,PUB(SPACE=(960,480))
/SET-JOB-STEP
/ASS-SYSLST L.RES.PLCO
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDASSOR1,ADAyyyyy.ASSO,SHARE-UPD=YES
/SET-FILE-LINK DDPLOGR1,ADAyyyyy.PLOGR1,SHARE-UPD=YES
/SET-FILE-LINK DDPLOGR2,ADAyyyyy.PLOGR2,SHARE-UPD=YES
/SET-FILE-LINK DDSIAUS1,ADAyyyyy.AUS1
/SET-FILE-LINK DDSIAUS2,ADAyyyyy.AUS2
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADARES,DB=yyyyy,IDTNAME=ADABAS5B
ADARES PLCOPY TWOCOPIES
/LOGOFF SYS-OUTPUT=DEL
```

In ISP Format:

```
/.ADARES LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK * A D A R E S COPY DUAL/MULTIPLE PROTECTION LOG
/REMARK *
/SYSFILE SYSLST=L.RES.PLCO
/FILE ADA.MOD,LINK=DDLIB
/FILE ADAyyyyy.ASSO ,LINK=DDASSOR1,SHARUPD=YES
/FILE ADAyyyyy.PLOGR1 ,LINK=DDPLOGR1,SHARUPD=YES
/FILE ADAyyyyy.PLOGR2 ,LINK=DDPLOGR2,SHARUPD=YES
/FILE ADAyyyyy.AUS1 ,LINK=DDSIAUS1,SPACE=(960,480)
/FILE ADAyyyyy.AUS2 ,LINK=DDSIAUS2,SPACE=(960,480)
/EXEC (ADARUN,ADA.MOD)
ADARUN PROG=ADARES,DB=yyyyy,IDTNAME=ADABAS5B
ADARES PLCOPY TWOCOPIES
/LOGOFF NOSPOOL
```

Backout Using a Sequential Protection Log

In SDF Format:

```
/.ADARES LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK * A D A R E S BACKOUT FROM SEQUENTIAL PLOG
/REMARK *
/ASS-SYSLST L.RES.BACK
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDASSOR1,ADAyyyyy.ASSO,SHARE-UPD=YES
/SET-FILE-LINK DDBACK,ADAyyyyy.BACK
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADARES,DB=yyyyy,IDTNAME=ADABAS5B
ADARES BACKOUT
/LOGOFF SYS-OUTPUT=DEL
```

In ISP Format:

```
/.ADARES LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK * A D A R E S BACKOUT FROM SEQUENTIAL PLOG
/REMARK *
/SYSFILE SYSLST=L.RES.BACK
/FILE ADA.MOD,LINK=DDLIB
/FILE ADAyyyyy.ASSO ,LINK=DDASSOR1,SHARUPD=YES
/FILE ADAyyyyy.BACK ,LINK=DDBACK
/EXEC (ADARUN,ADA.MOD)
ADARUN PROG=ADARES,DB=yyyyy,IDTNAME=ADABAS5B
ADARES BACKOUT
/LOGOFF NOSPOOL
```

Backout Using a Dual/Multiple Protection Log

In SDF Format:

```
/.ADARES LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK * A D A R E S BACKOUT FROM DUAL/MULTIPLE PLOG
/REMARK *
/ASS-SYSLST L.RES.BADP
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDASSOR1,ADAyyyyy.ASSO,SHARE-UPD=YES
/SET-FILE-LINK DDPLOGR1,ADAyyyyy.PLOGR1,SHARE-UPD=YES
/SET-FILE-LINK DDPLOGR2,ADAyyyyy.PLOGR2,SHARE-UPD=YES
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADARES,DB=yyyyy,IDTNAME=ADABAS5B
ADARES BACKOUT DPLOG
/LOGOFF SYS-OUTPUT=DEL
```

In ISP Format:

```
/.ADARES LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK * A D A R E S BACKOUT FROM DUAL/MULTIPLE PLOG
/REMARK *
/SYSFILE SYSLST=L.RES.BADP
/FILE ADA.MOD,LINK=DDLIB
/FILE ADAyyyyy.ASSO ,LINK=DDASSOR1,SHARUPD=YES
/FILE ADAyyyyy.PLOGR1 ,LINK=DDPLOGR1,SHARUPD=YES
/FILE ADAyyyyy.PLOGR2 ,LINK=DDPLOGR2,SHARUPD=YES
/EXEC (ADARUN,ADA.MOD)
ADARUN PROG=ADARES,DB=yyyyy,IDTNAME=ADABAS5B
ADARES BACKOUT DPLOG
/LOGOFF NOSPOOL
```

Regenerate Function

In SDF Format:

```
/.ADARES LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK * A D A R E S REGENERATE
/REMARK *
/ASS-SYSLST L.RES.REGE
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDASSOR1,ADAyyyyy.ASSO,SHARE-UPD=YES
/SET-FILE-LINK DDSIIN,ADAyyyyy.SIBA
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADARES,DB=yyyyy,IDTNAME=ADABAS5B
ADARES REGENERATE FILE=1,CONTINUE,PLOGNUM=ppp
/LOGOFF SYS-OUTPUT=DEL
```

In ISP Format:

```
/.ADARES LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK * A D A R E S REGENERATE
/REMARK *
/SYSFILE SYSLST=L.RES.REGE
/FILE ADA.MOD,LINK=DDLIB
/FILE ADAyyyyy.ASSO ,LINK=DDASSOR1,SHARUPD=YES
/FILE ADAyyyyy.SIBA ,LINK=DDSIIN
/EXEC (ADARUN,ADA.MOD)
ADARUN PROG=ADARES,DB=yyyyy,IDTNAME=ADABAS5B
ADARES REGENERATE FILE=1,CONTINUE,PLOGNUM=ppp
/LOGOFF NOSPOOL
```

Repair Data Storage

In SDF Format:

```
/.ADARES LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK * A D A R E S REPAIR DATASTORAGE
/REMARK *
/ASS-SYSLST L.RES.REPA
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDASSOR1,ADAyyyyy.ASSO,SHARE-UPD=YES
/SET-FILE-LINK DDSIIN,ADAyyyyy.SAVE
/SET-FILE-LINK DDSIIN01,ADAyyyyy.PLOG5
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADARES,DB=yyyyy,IDTNAME=ADABAS5B
ADARES REPAIR DSRABN=3456 3490
/LOGOFF SYS-OUTPUT=DEL
```

In ISP Format:

```
/.ADARES LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK * A D A R E S REPAIR DATASTORAGE
/REMARK *
/SYSFILE SYSLST=L.RES.REPA
/FILE ADA.MOD,LINK=DDLIB
/FILE ADAyyyyy.ASSO ,LINK=DDASSOR1,SHARUPD=YES
/FILE ADAyyyyy.SAVE ,LINK=DDSIIN
/FILE ADAyyyyy.PLOG5 ,LINK=DDSIIN01
/EXEC (ADARUN,ADA.MOD)
ADARUN PROG=ADARES,DB=yyyyy,IDTNAME=ADABAS5B
ADARES REPAIR DSRABN=3456 3490
/LOGOFF NOSPOOL
```

OS/390 or z/OS

Dataset	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.
Multiple protection log	DDPLOGnn	disk	Input logs for PLCOPY NOPPT function.
Multiple command log	DDCLOGnn	disk	Input logs for MERGE CLOG function.
Sequential protection log	DDBACK	tape	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 Manual*
ADARES parameters	DDKARTE	reader	
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADARES messages	DDDRUCK	printer	*Messages and Codes*

ADARES JCL Examples (OS/390 or z/OS)

Copy Sequential Protection Log

```
//ADARESCP  JOB
//*
//*    ADARES: COPY SEQUENTIAL PROTECTION LOG
//*
//RES      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
//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 MVSJOBS dataset 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.VVRS.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 MVSJOBS dataset for this example.

Automatically Copy/Merge Nucleus Cluster Protection Logs

Note:
Note that when intermediate datasets are used for both CLCOPY and PLCOPY, the dataset names must be unique so that they are not overwritten.

Following is sample JCL for allocating the required intermediate datasets MERGINI and MERGINO:

```
//ALLOC JOB
//*
//* Example to allocate the MERGINI and the MERGINO datasets
//*
//ALLOC   EXEC PGM=IEFBR14
//MERGINI DD  DISP=(NEW,CATLG,DELETE),DSN=EXAMPLE.PINTERI,
//             SPACE=(CYL,(1,10,0)),UNIT=3390,VOL=SER=volser,
//             RECFM=VB,BLKSIZE=27998,LRECL=27994
//MERGINO 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 MVSJOBS dataset 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.VVRS.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
//MERGINO  DD   DISP=SHR,DSN=EXAMPLE.INTERO         <=== INTER
//MERGINI  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=3380,DBID=YYYYY
/*
//DDKARTE   DD   *
ADARES PLCOPY NOPPT
/*
//
```

Refer to ADARESIP in the MVSJOBS dataset.

Copy Dual/Multiple Command Log

```
//ADARESCC  JOB
//*
//*     ADARES : COPY DUAL/MULTIPLE COMMAND LOG
//*
//RES      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
//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 MVSJOBS dataset for this example.

Automatically Copy/Merge Nucleus Cluster Command Logs

Note:
Note that when intermediate datasets are used for both CLCOPY and PLCOPY, the dataset names must be unique so that they are not overwritten.

Following is sample JCL for allocating the required intermediate datasets MERGINI and MERGINO:

```
//ALLOC JOB
//*
//* Example to allocate the MERGINI and the MERGINO datasets
//*
//ALLOC   EXEC PGM=IEFBR14
//MERGINI DD  DISP=(NEW,CATLG,DELETE),DSN=EXAMPLE.CINTERI,
//             SPACE=(CYL,(1,10,0)),UNIT=3390,VOL=SER=volser,
//             RECFM=VB,BLKSIZE=27998,LRECL=27994
//MERGINO 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 MVSJOBS dataset 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.VVRS.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=3380,DBID=YYYYY
/*
//DDKARTE   DD   *
ADARES MERGE CLOG,NUMLOG=3
/*
//
```

Refer to ADARESCM in the MVSJOBS dataset 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.Vvrs.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 MVSJOBS dataset 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.Vvrs.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 MVSJOBS dataset for this example.

Regenerate Function

```
//ADARESR   JOB
//*
//*    ADARES : REGENERATE
//*
//RES      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
//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 MVSJOBS dataset for this example.

Repair Data Storage

```
//ADARESRP  JOB
//*
//*    ADARES :  REPAIR DATASTORAGE
//*
//RES      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
//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 MVSJOBS dataset for this example.

VM/ESA or z/VM

Dataset	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 and BACKOUT DPLOG/MPLOG.
Multiple command log	DDCLOGRn	disk	Input logs for CLCOPY.
Sequential protection log	DDBACK	tape	Input log for BACKOUT (not BACKOUT DPLOG).
Copied log	DDSIAUS1	tape/disk	Output of CLCOPY and PLCOPY.
Extra copied log	DDSIAUS2	tape/disk	Required for TWOCOPIES.
Data Storage	DDDATARn	disk	Required for REGENERATE if FROMCP=SYN1 or SYN4.
Associator	DDASSORn	disk	
Recovery log (RLOG)	DDRLOGR1	disk	Required when using ADARAI.
ADARUN parameters	DDCARD	disk/ terminal/ reader	*Operations Manual*
ADARES parameters	DDKARTE	disk/ terminal/ reader	
ADARUN messages	DDPRINT	disk/ terminal/ printer	*Messages and Codes*
ADARES messages	DDDRUCK	disk/ terminal/ printer	*Messages and Codes*

ADARES JCL Examples (VM/ESA or z/VM)

Copy Sequential Protection Log

```
DATADEF DDASSOR1,DSN=ADABASVv.ASSO,VOL=ASSOV1
DATADEF DDSIIN,DSN=ADABASVv.SIBA,MODE=A
DATADEF DDSIAUS1,DSN=ADABASVv.SIAUS1,UNIT=181,VOL=SIBA01
DATADEF DDPRINT,DSN=ADARES.DDPRINT,MODE=A
DATADEF DUMP,DUMMY
DATADEF DDDRUCK,DSN=ADARES.DDDRUCK,MODE=A
DATADEF DDCARD,DSN=RUNRES.CONTROL,MODE=A
DATADEF DDKARTE,DSN=ADARES.CONTROL,MODE=A
ADARUN
```

Contents of RUNRES CONTROL A1

```
ADARUN PROG=ADARES,DEVICE=dddd,DB=yyyyy
```

Contents of ADARES CONTROL A1

```
ADARES COPY PLOGNUM=ppp
```

Copy Dual/Multiple Protection Log

```
DATADEF DDASSOR1,DSN=ADABASVv.ASSO,VOL=ASSOV1
DATADEF DDPLOGR1,DSN=ADABASVv.PLOG1,VOL=PLOGV1
DATADEF DDPLOGR2,DSN=ADABASVv.PLOG2,VOL=PLOGV2
DATADEF DDSIAUS1,DSN=ADABASVv.SIAUS1,UNIT=181,VOL=SIBA01
DATADEF DDSIAUS2,DSN=ADABASVv.SIAUS2,UNIT=182,VOL=SIBA02
DATADEF DDPRINT,DSN=ADARES.DDPRINT,MODE=A
DATADEF DUMP,DUMMY
DATADEF DDDRUCK,DSN=ADARES.DDDRUCK,MODE=A
DATADEF DDCARD,DSN=RUNRES.CONTROL,MODE=A
DATADEF DDKARTE,DSN=ADARES.CONTROL,MODE=A
ADARUN
```

Contents of RUNRES CONTROL A1

```
ADARUN PROG=ADARES,DEVICE=dddd,DB=yyyyy
```

Contents of ADARES CONTROL A1

```
ADARES PLCOPY TWOCOPIES
```

Copy Dual/Multiple Command Log

```
DATADEF DDASSOR1,DSN=ADABASVv.ASSO,VOL=ASSOV1
DATADEF DDCLOGR1,DSN=ADABASVv.CLOG1,VOL=CLOGV1
DATADEF DDCLOGR1,DSN=ADABASVv.CLOG1,VOL=CLOGV2
DATADEF DDSIAUS1,DSN=ADABASVv.CLOG1,UNIT=181,VOL=CLOG01
DATADEF DDPRINT,DSN=ADARES.DDPRINT,MODE=A
DATADEF DUMP,DUMMY
DATADEF DDDRUCK,DSN=ADARES.DDDRUCK,MODE=A
DATADEF DDCARD,DSN=RUNRES.CONTROL,MODE=A
DATADEF DDKARTE,DSN=ADARES.CONTROL,MODE=A
LOAD ADARUN ( START
```

Contents of RUNRES CONTROL A1

```
ADARUN PROG=ADARES,DEVICE=dddd,DB=yyyyy
```

Contents of ADARES CONTROL A1

```
ADARES CLCOPY
```

Backout Using a Sequential Protection Log

```
DATADEF DDASSOR1,DSN=ADABASVv.ASSO,VOL=ASSOV1
DATADEF DDBACK,DSN=ADABASVv.SIBA,MODE=A
DATADEF DDPRINT,DSN=ADARES.DDPRINT,MODE=A
DATADEF DUMP,DUMMY
DATADEF DDDRUCK,DSN=ADARES.DDDRUCK,MODE=A
DATADEF DDCARD,DSN=RUNRES.CONTROL,MODE=A
DATADEF DDKARTE,DSN=BACKOUT.CONTROL,MODE=A
LOAD ADARUN ( START
```

Contents of RUNRES CONTROL A1

```
ADARUN PROG=ADARES,DEVICE=dddd,DB=yyyyy
```

Contents of BACKOUT CONTROL A1

```
ADARES BACKOUT
```

Backout Using a Dual/Multiple Protection Log

```
DATADEF DDASSOR1,DSN=ADABASVv.ASSO,VOL=ASSOV1
DATADEF DDPLOGR1,DSN=ADABASVv.PLOG1,VOL=PLOGV1
DATADEF DDPLOGR2,DSN=ADABASVv.PLOG2,VOL=PLOGV2
DATADEF DDPRINT,DSN=ADARES.DDPRINT,MODE=A
DATADEF DUMP,DUMMY
DATADEF DDDRUCK,DSN=ADARES.DDDRUCK,MODE=A
DATADEF DDCARD,DSN=RUNRES.CONTROL,MODE=A
DATADEF DDKARTE,DSN=BACKOUT.CONTROL,MODE=A
LOAD ADARUN ( START
```

Contents of RUNRES CONTROL A1

```
ADARUN PROG=ADARES,DEVICE=dddd,DB=yyyyy
```

Contents of BACKOUT CONTROL A1

```
ADARES BACKOUT DPLOG
```

Regenerate Function

```
DATADEF DDASSOR1,DSN=ADABASVv.ASSO,VOL=ASSOV1
DATADEF DDSIIN,DSN=ADABASVv.SIBA,MODE=A
DATADEF DDPRINT,DSN=ADARES.DDPRINT,MODE=A
DATADEF DUMP,DUMMY
DATADEF DDDRUCK,DSN=ADARES.DDDRUCK,MODE=A
DATADEF DDCARD,DSN=RUNRES.CONTROL,MODE=A
DATADEF DDKARTE,DSN=REGENER.CONTROL,MODE=A
LOAD ADARUN ( START
```

Contents of RUNRES CONTROL A1

```
ADARUN PROG=ADARES,DEVICE=dddd,DB=yyyyy
```

Contents of REGENER CONTROL A1

```
ADARES REGENERATE FILE=1
```

Repair Data Storage

```
DATADEF DDASSOR1,DSN=ADABASVv.ASSO,VOL=ASSOV1
DATADEF DDSIIN,DSN=ADABASVv.SAVE,MODE=A
DATADEF DDSIIN,DSN=ADABASVv.SIBA,MODE=A,CONCAT=1
DATADEF DDPLOGR1,DSN=ADABASVv.PLOG1,VOL=PLOGV1
DATADEF DDPLOGR2,DSN=ADABASVv.PLOG2,VOL=PLOGV2
DATADEF DDPRINT,DSN=ADARES.DDPRINT,MODE=A
DATADEF DUMP,DUMMY
DATADEF DDDRUCK,DSN=ADARES.DDDRUCK,MODE=A
DATADEF DDCARD,DSN=RUNRES.CONTROL,MODE=A
DATADEF DDKARTE,DSN=BACKOUT.CONTROL,MODE=A
LOAD ADARUN ( START
```

Contents of RUNRES CONTROL A1

```
ADARUN PROG=ADARES,DEVICE=dddd,DB=yyyyy
```

Contents of BACKOUT CONTROL A1

```
ADARES REPAIR DSRABN=3456-3490
```

VSE/ESA

Dataset	Symbolic	Storage	Logical Unit	More Information
Sequential protection log or ADASAV SAVEn output	SIIN	tape disk	SYS020 See note	Input log for COPY, REGENERATE, and REPAIR.
Multiple protection log	PLOGRn	disk	See note	Input logs for PLCOPY and BACKOUT DPLOG/MPLOG.
Multiple command log	CLOGRn	disk	See note	Input logs for CLCOPY.
Multiple protection log	PLOGnn	disk	See note	Input logs for PLCOPY NOPPT function.
Multiple command log	CLOGnn	disk	See note	Input logs for MERGE CLOG function.
Sequential protection log	BACK	tape	SYS020	Input log for BACKOUT (not BACKOUT DPLOG).
Copied log	SIAUS1	tape disk	SYS021 See note	Output of COPY, CLCOPY, and PLCOPY.
Extra copied log	SIAUS2	tape disk	SYS022 See note	Required for TWOCOPIES.
Data Storage	DATARn	disk		Required for REGENERATE if FROMCP=SYN1 or SYN4.
Associator	ASSORn		See note	
Recovery log (RLOG)	RLOGR1	disk		Required when using ADARAI.
ADARUN parameters	— CARD CARD	reader tape disk	SYSRDR SYS000 See note	
ADARES parameters		reader	SYSIPT	
ADARUN messages		printer	SYSLST	
ADARES messages		printer	SYS009	

Note:
Any programmer logical unit can be used.

ADARES JCS Examples (VSE/ESA)

See appendix B for descriptions of the VSE/ESA procedures (PROCs).

Refer to the following members for these examples:

Example	Member
Copy sequential protection log	ADARESCP.X
Copy dual/multiple protection log	ADARESCD.X
Copy dual/multiple command log	ADARESCC.X
Regenerate	ADARESR.X
Backout from a sequential protection log	ADARESSP.X
Backout from a dual protection log	ADARESB.X
Repair Data Storage	ADARESRP.X

Copy Sequential Protection Log

```
* $$ JOB JNM=ADARESCP,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADARESCP
*       COPY SEQUENTIAL PLOG(TAPE)
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// ASSGN SYS020,TAPE,D0
// PAUSE MOUNT INPUT TAPE ON TAPE cuu
// MTC REW,SYS020
// TLBL SIIN,'ADABAS.Vvr.SIBA'
// ASSGN SYS022,TAPE,D0
// PAUSE MOUNT SCRATCH TAPE ON TAPE cuu
// MTC REW,SYS022
// MTC WTM,SYS022,5
// MTC REW,SYS022
// TLBL SIAUS1,'ADABAS.Vvr.SIAUS1'
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADARES,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADARES COPY
/*
/&
* $$ EOJ
```

Copy Dual/Multiple Protection Log

```
* $$ JOB JNM=ADARESCD,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADARESCD
*       CLCOPY WITH OPTION TWOCOPIES(TAPE)
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// ASSGN SYS021,TAPE,D0
// ASSGN SYS022,TAPE,D0
// PAUSE MOUNT SCRATCH TAPE ON TAPES cu1 AND cu2
// MTC REW,SYS022
// MTC WTM,SYS022,5
// MTC REW,SYS022
// MTC REW,SYS021
// MTC WTM,SYS021,5
// MTC REW,SYS021
// TLBL SIAUS1,'ADABAS.Vvr.PLOGC1'
// TLBL SIAUS2,'ADABAS.Vvr.PLOGC2'
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADARES,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADARES PLCOPY TWOCOPIES
/*
/&
* $$ EOJ
```

Copy Dual/Multiple Command Log

```
* $$ JOB JNM=ADARESCC,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADARESCC
*       COPY DUAL/MULTIPLE COMMAND LOG
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// ASSGN SYS022,DISK,VOL=vvvvvv,SHR
// DLBL SIAUS1,'ADABAS.Vvr.CLOG',0,SD
// EXTENT SYS022,vvvvvv,1,0,sssss,nnnnn
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADARES,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADARES CLCOPY
/*
/&
* $$ EOJ
```

Regenerate

```
* $$ JOB JNM=ADARESR,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADARESR
*       REGENERATE
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// ASSGN SYS020,DISK,VOL=vvvvvv,SHR
// DLBL SIIN,'EXAMPLE.DByyyyy.PLOG'
// EXTENT SYS020,vvvvvv
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADARES,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADARES REGENERATE FILE=1
/*
/&
* $$ EOJ
```

Backout from a Sequential Protection Log

```
* $$ JOB JNM=ADARESSP,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADARESSP
*       BACKOUT FROM A SEQUENTIAL PLOG
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// ASSGN SYS020,TAPE
// PAUSE  ***  PLEASE MOUNT TAPE ***
// MTC REW,SYS020
// TLBL BACK,'DByyyyy.PLCOPY.TAPE'
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADARES,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADARES BACKOUT PLOGNUM=ppp
/*
/&
* $$ EOJ
```

Backout from a Dual Protection Log

```
* $$ JOB JNM=ADARESB,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADARESB
*       BACKOUT FROM DUAL PLOG
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADARES,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADARES BACKOUT DPLOG
/*
/&
* $$ EOJ
```

Repair Data Storage

```
* $$ JOB JNM=ADARESRP,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADARESRP
*       REPAIR DATASTORAGE
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// ASSGN SYS020,TAPE
// PAUSE MOUNT INPUT TAPE ON cuu
// MTC REW,SYS020
// TLBL SIIN,'EXAMPLE.ADAyyyyy.SAVE1'
// TLBL SIIN01,'ADABAS.ADAyyyyy.PLOG5'       (*)
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADARES,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADARES REPAIR DSRABN=3456-3490
/*
/&
* $$ EOJ
```

(*) See appendix A, **Sequential Input Files**, VSE concatenation starting on page NO TAG.

19

ADASAV : SAVE/RESTORE DATABASE OR FILES

Functional Overview

The ADASAV utility saves and restores the contents of the database, specific files, or a file to or from a sequential dataset.

ADASAV should be run as often as required for the number and size of the files contained in the database, and 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. For more information, see the *Adabas Delta Save Facility Manual*.

RESTONL and RESTORE Functions

For either RESTORE or RESTONL function operations, the Associator and Data Storage datasets 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 dataset created while the Adabas nucleus was **active** (that is, online); RESTORE functions restore from a SAVE dataset 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 block, Associator RABNs 2–30 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.

ADASAV Function Directory

Function	Action	Page
RESTONL	Restore the entire database from online source.	301
RESTONL GCB	Restore from online source the general control block, Associator RABNs 2–30, and specified files.	305
RESTONL FILES	Restore from online source one or more files to an existing database to their original RABN assignments.	311
RESTONL FMOVE	Restore from online source one or more files to an existing database to any free space, allowing extent size changes.	318
RESTORE	Restore the entire database from offline source.	329
RESTORE GCB	Restore from offline source the general control block, Associator RABNs 2–30, and specified files.	333
RESTORE FILES	Restore from offline source one or more files to an existing database to their original RABN assignments.	337
RESTORE FMOVE	Restore from offline source one or more files to an existing database to any free space, allowing extent size changes.	343
RESTPLOG	Apply updates not included in unsuccessful RESTONL execution.	352
SAVE	Save the entire database.	355
SAVE FILES	Save one or more files.	358

RESTONL : Restore Database from Online Source

The RESTONL function restores a database from a database SAVE dataset created while the Adabas nucleus was **active**.

Conditions

To use the RESTONL (database) function, the following conditions must be met:

1. The correct SAVE dataset 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 datasets must be present and must have been previously formatted. The SAVE dataset 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) dataset containing information written by the nucleus session at the time of the SAVE operation (see output of SAVE run) must be supplied. PLOG datasets from other sessions may also be included.
5. If the SAVE operation was performed with the DRIVES parameter, the SAVE datasets created can also be restored with the DRIVES parameter. In that case, the restore operation is performed from the different SAVE datasets in parallel. Alternatively, the SAVE datasets can be concatenated to a single SAVE dataset for a restore operation without the DRIVES parameter.

Result

The result of this function is a database with the same physical status it had at the end of the ADASAV SAVE operation.

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 [RESTORE] (database) [GCB] job control contains the DD names, symbolic names, or link names for one or more Work datasets, these datasets are reset (made logically empty). More than one Work dataset can be specified for cluster databases (Adabas Cluster Services or Adabas Parallel Services), where the Work datasets of all cluster nuclei should be given. Specifying the Work datasets in the job control is optional but recommended. An Adabas nucleus will not start if its Work dataset still contains data after the database has been restored.

Syntax

```
ADASAV RESTONL  [BUFNO={number-of-buffers|1}]
                [CLOGDEV={CLOG1-device-type|ADARUN-device}]
                [DRIVES=count|1]
                [EXCLUDE= file-list]
                [NEWDBID=new-database-id]
                [NEWDBNAME=new-database-name]
                [NOUSERABEND]
                [OVERWRITE]
                [PLOGDEV={PLOG-device-type|ADARUN-device}]
                [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. 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 dataset.

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).

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 1–65,535; 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 dataset.

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 an error is encountered while the function is running, the utility 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.

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 dataset.

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 dataset.

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 dataset.

RESTONL GCB : Restore Database Incremental from Online Source

From a database SAVE dataset created while the Adabas nucleus was **active,** the RESTONL GCB function restores

- the general control block (GCB);
- Associator RABNs 2–30 of the database;
- the checkpoint file;
- the security file (if present); and
- all files specified with the FILES parameter.

Conditions

To use the RESTONL GCB function, the following conditions must be met:

1. The correct SAVE dataset 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 datasets must be present and must have been previously formatted. The SAVE dataset 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) dataset containing information written by the nucleus session at the time of the SAVE operation (see output of SAVE run) must be supplied. PLOG datasets from other sessions may also be included.
5. If the SAVE operation was performed with the DRIVES parameter, the SAVE datasets created can also be restored with the DRIVES parameter. In that case, the restore operation is performed from the different SAVE datasets in parallel. Alternatively, the SAVE datasets can be concatenated to a single SAVE dataset for a restore operation without the DRIVES parameter.
6. For restoring just a few files from a multivolume database SAVE dataset, 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 datasets is completely overwritten and any files in that database are lost.

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 [RESTORE] (database) [GCB] job control contains the DD names, symbolic names, or link names for one or more Work datasets, these datasets are reset (made logically empty). More than one Work dataset can be specified for cluster databases (Adabas Cluster Services or Adabas Parallel Services), where the Work datasets of all cluster nuclei should be given. Specifying the Work datasets in the job control is optional but recommended. An Adabas nucleus will not start if its Work dataset still contains data after the database has been restored.

Syntax

```
ADASAV RESTONL GCB [BUFNO={number-of-buffers|1}]
                   [CLOGDEV={CLOG-device-type|ADARUN-device}]
                   [DRIVES={count|1}]
                   [EXCLUDE= file-list]
                   [FILES=file-list]
                   [NEWDBID=new-database-id]
                   [NEWDBNAME=new-database-name]
                   [NOUSERABEND]
                   [OVERWRITE]
                   [PLOGDEV={PLOG-device-type|ADARUN-device}]
                   [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 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.

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 dataset.

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).

FILES : Files to Be Restored

FILES specifies the file or files to be included in the database restore operation. The checkpoint and security files (if any) are always restored. 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.

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 1–65,535; 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 dataset.

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 an error is encountered while the function is running, the utility 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.

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 dataset.

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 dataset.

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 GCB 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.

RESTONL FILES : Restore Files to Original RABNs from Online Source

The RESTONL FILES function restores files from a file or database SAVE dataset 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.

Conditions

To use the RESTONL FILES function, the following conditions must be met:

1. The correct SAVE dataset must be supplied. It can be a database or file SAVE dataset and must contain the files to be restored. SAVE datasets from 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. 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. All RABNs originally used by the file(s) to be restored must have the same device type in the target database of the restore as in the original database of the save.
4. 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.
5. The protection log (PLOG) dataset containing information written by the nucleus session at the time of the SAVE operation (see output of SAVE run) must be supplied. PLOG datasets 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 dataset(s) can be omitted.
6. 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.
7. If the SAVE operation was performed with the DRIVES parameter, the SAVE datasets created can also be restored with the DRIVES parameter. In that case, the restore operation is performed from the different SAVE datasets in parallel. Alternatively, the SAVE datasets can be concatenated to a single SAVE dataset for a restore operation without the DRIVES parameter.

8. For restoring just a few files from a multivolume database SAVE dataset, 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.

9. 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.

10. 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.

11. 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.

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.

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]
                                 [TEST]
```

The FILES file list specifies the file or files to be restored. 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 dataset.

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 dataset 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.

The parameter is optional: if not specified, no files are excluded. A file number may be listed only once.

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.

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 starting on page 317.

NOUSERABEND : Termination without ABEND

When an error is encountered while the function is running, the utility 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.

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, Software AG recommends 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 on page 317 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 dataset.

Note:
This is not possible when restoring from a version 5.1 SAVE dataset.

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 dataset.

Note:
This is not possible when restoring from a version 5.1 SAVE dataset.

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.

RESTONL FMOVE : Restore Files to Any RABNs from Online Source

The RESTONL FMOVE function restores files from a file or database SAVE dataset 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.

Conditions

To use the RESTONL FMOVE function, the following conditions must be met:

1. The correct SAVE dataset must be supplied. It can be a database or file SAVE dataset and must contain the files to be restored. SAVE datasets 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. 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.

4. 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.

5. The protection log (PLOG) dataset containing information written by the nucleus session at the time of the SAVE operation (see output of SAVE run) must be supplied. PLOG datasets 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 dataset(s) can be omitted.

6. 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.

7. If the SAVE operation was performed with the DRIVES parameter, the SAVE datasets created can also be restored with the DRIVES parameter. In that case, the restore operation is performed from the different SAVE datasets in parallel. Alternatively, the SAVE datasets can be concatenated to a single SAVE dataset for a restore operation without the DRIVES parameter.

8. For restoring just a few files from a multivolume database SAVE dataset, 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.

9. 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.

10. 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.

11. 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.

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.

Syntax

```
ADASAV RESTONL FMOVE=file-list [ACRABN=AC-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]
                               [NEWFILES= file-list]
                               [NIRABN=NI-start-list]
                               [NISIZE=NI-size-list]
                               [NOUSERABEND]
                               [OVERWRITE]
                               [PASSWORD='password-list']
                               [PLOGNUM=protection-log-number]
                                    [{SYN1 | SYN4}=PLOG-block-number]
                               [TEST]
                               [UIRABN=UI-start-rabn-list]
                               [UISIZE=UI-size-list]
```

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.

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.

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 on page 327.

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.

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 on page 327.

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.

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 on page 327.

DRIVES : Tape Drives for Parallel Restore

ADASAV is able to restore files from multiple save dataset 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.

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 on page 327.

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.

The size can be specified in cylinders, or in blocks (by appending “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 on page 327.

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.

The parameter is optional: if not specified, no files are excluded. A file number may be listed only once.

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).

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 on page 327.

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 addition 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.

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.

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 on page 327.

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.

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 on page 327.

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.

The size can be specified in cylinders, or in blocks (by appending “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 on page 327.

NOUSERABEND : Termination without ABEND

When an error is encountered while the function is running, the utility 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.

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, Software AG recommends 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 on page 327 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 dataset.

Note:
This is not possible when restoring from a version 5.1 SAVE dataset.

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 dataset.

Note:
This is not possible when restoring from a version 5.1 SAVE dataset.

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.

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 on page 327.

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.

The size can be specified in cylinders, or in blocks (by appending “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 starting on page 327.

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.

RESTORE : Restore Database from Offline Source

The RESTORE function restores a database from a database SAVE dataset created while the Adabas nucleus was **inactive**.

Conditions

To use the RESTORE (database) function, the following conditions must be met:

1. The correct SAVE dataset 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 datasets must be present and must have been previously formatted. The SAVE dataset 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 datasets created can also be restored with the DRIVES parameter. In that case, the restore operation is performed from the different SAVE datasets in parallel. Alternatively, the SAVE datasets can be concatenated to a single SAVE dataset for a restore operation without the DRIVES parameter.

Result

The result of this function is a database with the same physical status it had at the time of the ADASAV SAVE operation.

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 RESTONL [RESTORE] (database) [GCB] job control contains the DD names, symbolic names, or link names for one or more Work datasets, these datasets are reset (made logically empty). More than one Work dataset can be specified for cluster databases (Adabas Cluster Services or Adabas Parallel Services), where the Work datasets of all cluster nuclei should be given. Specifying the Work datasets in the job control is optional but recommended. An Adabas nucleus will not start if its Work dataset still contains data after the database has been restored.

Syntax

```
ADASAV RESTORE [BUFNO={number-of-buffers|1}]
               [CLOGDEV={CLOG-device-type|ADARUN-device}]
               [DRIVES={count|1}]
               [EXCLUDE= file-list]
               [NEWDBID=new-database-id]
               [NEWDBNAME=new-database-name]
               [NOUSERABEND]
               [OVERWRITE]
               [PLOGDEV={PLOG-device-type|ADARUN-device}]
               [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 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. 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 dataset.

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).

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 1–65,535; 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 dataset.

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 an error is encountered while the function is running, the utility 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.

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 dataset.

RESTORE GCB : Restore Database Incremental from Offline Source

From a database SAVE dataset created while the Adabas nucleus was **inactive,** the RESTORE GCB function restores

- the general control block (GCB);
- Associator RABNs 2–30 of the database;
- the checkpoint file;
- the security file (if present); and
- all files specified with the FILES parameter.

Conditions

To use the RESTORE GCB function, the following conditions must be met:

1. The correct SAVE dataset 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 datasets must be present and must have been previously formatted. The SAVE dataset 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 datasets created can also be restored with the DRIVES parameter. In that case, the restore operation is performed from the different SAVE datasets in parallel. Alternatively, the SAVE datasets can be concatenated to a single SAVE dataset for a restore operation without the DRIVES parameter.
5. For restoring just a few files from a multivolume database SAVE dataset, 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 datasets is completely overwritten and any files in that database are lost.

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 RESTONL [RESTORE] (database) [GCB] job control contains the DD names, symbolic names, or link names for one or more Work datasets, these datasets are reset (made logically empty). More than one Work dataset can be specified for cluster databases (Adabas Cluster Services or Adabas Parallel Services), where the Work datasets of all cluster nuclei should be given. Specifying the Work datasets in the job control is optional but recommended. An Adabas nucleus will not start if its Work dataset still contains data after the database has been restored.

Syntax

```
ADASAV RESTORE GCB [BUFNO={number-of-buffers|1}]
                   [CLOGDEV={CLOG-device-type|ADARUN-device}]
                   [DRIVES={count|1}]
                   [EXCLUDE= file-list]
                   [FILES=file-list]
                   [NEWDBID=new-database-id]
                   [NEWDBNAME=new-database-name]
                   [NOUSERABEND]
                   [OVERWRITE]
                   [PLOGDEV={PLOG-device-type|ADARUN-device}]
                   [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.

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 dataset.

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).

FILES : Files to Be Restored

FILES specifies the file or files to be included in the database restore operation. 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 1–65,535; 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 dataset.

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 an error is encountered while the function is running, the utility 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.

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 GCB,FILES=2,4,6,8
```

The database Associator GCB, RABNs 2–30, 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 GCB and files 3, 4, and 5 are to be restored; the existing database will be overwritten.

RESTORE FILES : Restore Files to Original RABNs from Offline Source

The RESTORE FILES function restores files from a file or database SAVE dataset created while the Adabas nucleus was **inactive**, or from a file SAVE dataset created with UTYPE=EXU. One or more files can be restored. The files are restored into an existing database to their original RABNs.

Conditions

To use the RESTORE FILES function, the following conditions must be met:

1. The correct SAVE dataset must be supplied. It can be a database or file SAVE dataset and must contain the files to be restored.

2. An existing database must be present. The files to be restored may have originated from this or from a different database. SAVE datasets from Adabas version 5.1 or above can be used.

3. 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. All RABNs originally used by the file(s) to be restored must have the same device type in the target database of the restore as in the original database of the save.

4. 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.

5. If the SAVE operation was performed with the DRIVES parameter, the SAVE datasets created can also be restored with the DRIVES parameter. In that case, the restore operation is performed from the different SAVE datasets in parallel. Alternatively, the SAVE datasets can be concatenated to a single SAVE dataset for a restore operation without the DRIVES parameter.

6. For restoring just a few files from a multivolume database SAVE dataset, 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.

7. 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.

8. 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.

9. 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.

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.

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']
                                 [TEST]
```

The FILES file list specifies the file or files to be restored.

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 dataset 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 dataset.

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.

The parameter is optional: if not specified, no files are excluded. A file number may be listed only once.

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.

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 starting on page 342.

NOUSERABEND : Termination without ABEND

When an error is encountered while the function is running, the utility 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.

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, Software AG recommends 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 on page 342 for more information about the PASSWORD parameter. When overwriting password-protected files, the Adabas nucleus must be active.

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.

RESTORE FMOVE : Restore Files to Any RABNs from Offline Source

The RESTORE FMOVE function restores files from a file or database SAVE dataset created while the Adabas nucleus was **inactive**, or from a file SAVE dataset 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.

Conditions

To use the RESTORE FMOVE function, the following conditions must be met:

1. The correct SAVE dataset must be supplied. It can be a database or file SAVE dataset and must contain the files to be restored. SAVE datasets 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. 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.

4. 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.

5. If the SAVE operation was performed with the DRIVES parameter, the SAVE datasets created can also be restored with the DRIVES parameter. In that case, the restore operation is performed from the different SAVE datasets in parallel. Alternatively, the SAVE datasets can be concatenated to a single SAVE dataset for a restore operation without the DRIVES parameter.

6. For restoring just a few files from a multivolume database SAVE dataset, 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.

7. 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.

8. 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.

9. 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.

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.

Syntax

The FMOVE file list specifies a file or files to be restored using new RABNs (and sizes).

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]
                               [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]
                               [NEWFILES=file-list]
                               [NIRABN=NI-start-list]
                               [NISIZE=NI-size-list]
                               [NOUSERABEND]
                               [OVERWRITE]
                               [PASSWORD='password-list']
                               [UIRABN=UI-start-rabn-list]
                               [UISIZE=UI-size-list]
                               [TEST]
```

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.

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 on page 351.

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.

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 on page 351.

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.

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 on page 351.

DRIVES : Tape Drives for Parallel Restore

ADASAV is able to restore files from multiple save dataset 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.

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 on page 351.

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.

The size can be specified in cylinders, or in blocks (by appending “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 on page 351.

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. A file number may be listed only once.

If not specified, no files are excluded.

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).

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 on page 351.

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.

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 on page 351.

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.

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 on page 351.

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.

The size can be specified in cylinders, or in blocks (by appending “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 on page 351.

NOUSERABEND : Termination without ABEND

When an error is encountered while the function is running, the utility 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.

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, Software AG recommends 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 on page 351 for more information about the PASSWORD parameter. When overwriting password-protected files, the Adabas nucleus must be active.

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.

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 on page 351.

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.

The size can be specified in cylinders, or in blocks (by appending “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 on page 351.

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.

RESTPLOG : Restore Protection Log Only

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 dataset 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.

If NEWFILES is not specified, the files to be restored retain their original numbers.

NOUSERABEND : Termination without ABEND

When an error is encountered while the function is running, the utility 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.

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.

SAVE : Save Database

The ADASAV SAVE (database) function saves the contents of the database to a sequential dataset. 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 dataset.

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 dataset 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 datasets in parallel when the DRIVES parameter is specified. This mode of operation may significantly reduce the duration of the save. The resulting SAVE datasets, when concatenated in the order of ascending drive number, are equivalent to a single SAVE dataset 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

```
ADASAV SAVE [BUFNO={number-of-buffers|1}]
            [DRIVES={count|1}]
            [INCREMENTAL]
            NOUSERABEND]
            [PERDRIVE=disk-drive-per-tape-drive,...]
            [TTSYN={ seconds | ADARUN-tt }]
            [TWOCOPIES]
            [TEST]
```

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 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 an error is encountered while the function is running, the utility 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.

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 DD/SAVEn and DD/DUALn job control specifications, as described at the end of this chapter.

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 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.

SAVE FILES : Save Specified Files

The ADASAV SAVE FILES function saves the contents of one or more files to a sequential dataset. 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 dataset.

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 dataset 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 datasets in parallel when the DRIVES parameter is specified. This mode of operation may significantly reduce the duration of the save. The resulting SAVE datasets, when concatenated in the order of ascending drive number, are equivalent to a single SAVE dataset 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 | ADARUN-tt}]
                             [TWOCOPIES]
                             [UTYPE=EXU]
```

The FILES file list specifies the file(s) to be saved. 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 an error is encountered while the function is running, the utility 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.

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 chapter 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 or SAVEn/DUALn job control specifications, as described at the end of this chapter.

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.

JCL/JCS Requirements and Examples

This section describes the job control information required to run ADASAV with BS2000, OS/390 or z/OS, VM/ESA or z/VM, and VSE/ESA systems and shows examples of each of the job streams.

BS2000

Dataset	Link Name	Storage	More Information
Associator	DDASSORn	disk	
Data Storage	DDDATARn	disk	
Work	DDWORKRn/ DDWORKmm	disk	Required if Adabas nucleus is not active
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 to the RESTORE function)	DDREST1-8	tape/disk	Required for RESTORE and RESTONL
Sequential protection log	DDPLOG	tape/disk	Required for RESTONL and RESTPLOG
ADARUN parameters	SYSDTA/ DDCARD		*Operations Manual*
ADASAV parameters	SYSDTA/ DDKARTE		
ADARUN messages	SYSOUT DDPRINT		*Messages and Codes*
ADASAV messages	SYSLST DDDRUCK		*Messages and Codes*

Notes:

1. *For RESTONL, the input SAVE tapes and the sequential protection log can be concatenated, using the name DDREST1.*
2. *Multiple Work datasets are specified as DD/WORKR1 to DD/WORKR9 and, beyond that, DD/WORK10 to DD/WORK99.*

ADASAV JCL Examples (BS2000)

Save Files, Save Database

In SDF Format:

```
/.ADASAV LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK * A D A S A V SAVE FILES, SAVE DATABASE
/REMARK *
/DELETE-FILE ADAyyyyy.SAVE1
/SET-JOB-STEP
/CREATE-FILE ADAyyyyy.SAVE1,SUP=TAPE(DEVICE=TAPE-C1,VOL=SAV101),-
/  PROT=(USER-ACCESS=ALL-USERS)
/SET-JOB-STEP
/DELETE-FILE ADAyyyyy.DUAL1
/SET-JOB-STEP
/CREATE-FILE ADAyyyyy.DUAL1,SUP=TAPE(DEVICE=TAPE-C1,VOL=SAV101),-
/  PROT=(USER-ACCESS=ALL-USERS)
/SET-JOB-STEP
/ASS-SYSLST L.SAV.SAVE
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDASSOR1,ADAyyyyy.ASSO,SHARE-UPD=YES
/SET-FILE-LINK DDDATAR1,ADAyyyyy.DATA,SHARE-UPD=YES
/SET-FILE-LINK DDWORKR1,ADAyyyyy.WORK,SHARE-UPD=YES
/SET-FILE-LINK DDSSAVE1,ADAyyyyy.SAVE1,TAPE(FILE-SEQ=1)
/SET-FILE-LINK DDSDUAL1,ADAyyyyy.DUAL1,TAPE(FILE-SEQ=1)
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADASAV,DB=yyyyy,IDTNAME=ADABAS5B
ADASAV SAVE TWOCOPIES
/LOGOFF SYS-OUTPUT=DEL
```

Save Files, Save Database (Continued)

In ISP Format:

```
/.ADASAV LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK * A D A S A V SAVE FILES, SAVE DATABASE
/REMARK *
```

```
/SYSFILE SYSLST=L.SAV.SAVE
/FILE ADA.MOD,LINK=DDLIB
/FILE ADAyyyyy.ASSO ,LINK=DDASSOR1,SHARUPD=YES
/FILE ADAyyyyy.DATA ,LINK=DDDATAR1,SHARUPD=YES
/FILE ADAyyyyy.WORK ,LINK=DDWORKR1,SHARUPD=YES
/FILE ADAyyyyy.SAVE1 ,LINK=DDSAVE1 ,DEVICE=TAPE-C1,VOLUME=SAV101
/FILE ADAyyyyy.DUAL1 ,LINK=DDDUAL1 ,DEVICE=TAPE-C1,VOLUME=SAV201
/EXEC (ADARUN,ADA.MOD)
ADARUN PROG=ADASAV,DB=yyyyy,IDTNAME=ADABAS5B
ADASAV SAVE TWOCOPIES
/LOGOFF NOSPOOL
```

Restore Files from SAVE Datasets Created Online

In SDF Format:

```
/.ADASAV LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK * A D A S A V RESTORE FILES, RESTORE DATABASE
/REMARK * FROM ONLINE CREATED SAVE DATASETS
/REMARK *
/ASS-SYSLST L.SAV.REON
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDASSOR1,ADAyyyyy.ASSO,SHARE-UPD=YES
/SET-FILE-LINK DDDATAR1,ADAyyyyy.DATA,SHARE-UPD=YES
/SET-FILE-LINK DDWORKR1,ADAyyyyy.WORK,SHARE-UPD=YES
/SET-FILE-LINK DDPLOGR1,ADAyyyyy.PLOGR1,SHARE-UPD=YES
/SET-FILE-LINK DDPLOGR2,ADAyyyyy.PLOGR2,SHARE-UPD=YES
/SET-FILE-LINK DDREST1,ADAyyyyy.SAVE1
/SET-FILE-LINK DDPLOG,ADAyyyyy.PLOG
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADASAV,DB=yyyyy,IDTNAME=ADABAS5B
ADASAV RESTONL FILES=2,PLOGNUM=ppp,SYN1=43
/LOGOFF SYS-OUTPUT=DEL
```

Restore Files from SAVE Datasets Created Online (Continued

In ISP Format:

```
/.ADASAV LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK * A D A S A V RESTORE FILES, RESTORE DATABASE
/REMARK * FROM ONLINE CREATED SAVE DATASETS
/REMARK *
```

```
/SYSFILE SYSLST=L.SAV.REON
/FILE ADA.MOD,LINK=DDLIB
/FILE ADAyyyyy.ASSO ,LINK=DDASSOR1,SHARUPD=YES
/FILE ADAyyyyy.DATA ,LINK=DDDATAR1,SHARUPD=YES
/FILE ADAyyyyy.WORK ,LINK=DDWORKR1,SHARUPD=YES
/FILE ADAyyyyy.SAVE1 ,LINK=DDREST1
/FILE ADAyyyyy.PLOG ,LINK=DDPLOG
/EXEC (ADARUN,ADA.MOD)
ADARUN PROG=ADASAV,DB=yyyyy,IDTNAME=ADABAS5B
ADASAV RESTONL FILES=2,PLOGNUM=ppp,SYN1=43
/LOGOFF NOSPOOL
```

Restore Database

In SDF Format:

```
/.ADASAV LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK * ADASAV:
/REMARK * EXAMPLE HOW TO USE ADASAV TO RESTORE THE
/REMARK * ENTIRE DATABASE /REMARK *
/REMARK *
/DELETE-FILE ADAyyyyy.SAVE1
/SET-JOB-STEP
/IMPORT-FILE SUP=TAPE(F-NAME=ADAyyyyy.SAVE1,DEV-TYPE=TAPE-C1,VOL=SAV101)
/SET-JOB-STEP
/ASS-SYSLST L.SAV.REST
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDASSOR1,ADAyyyyy.ASSO,SHARE-UPD=YES
/SET-FILE-LINK DDDATAR1,ADAyyyyy.DATA,SHARE-UPD=YES
/SET-FILE-LINK DDWORKR1,ADAyyyyy.WORK,SHARE-UPD=YES
/SET-FILE-LINK DDPLOGR1,ADAyyyyy.PLOGR1,SHARE-UPD=YES
/SET-FILE-LINK DDPLOGR2,ADAyyyyy.PLOGR2,SHARE-UPD=YES
/SET-FILE-LINK DDREST1,ADAyyyyy.SAVE1,TAPE(FILE-SEQ=1),ACC-METH=SAM,-
/  BUFF-LEN=32768,REC-FORM=V
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADASAV,DB=yyyyy,IDTNAME=ADABAS5B
ADASAV RESTORE OVERWRITE
/LOGOFF SYS-OUTPUT=DEL
```

Restore Database (Continued)

In ISP Format:

```
/.ADASAV LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK * ADASAV:
/REMARK * EXAMPLE HOW TO USE ADASAV TO RESTORE THE
/REMARK * ENTIRE DATABASE
/REMARK *
/SYSFILE SYSLST=L.SAV.REST
/FILE ADA.MOD,LINK=DDLIB
/FILE ADAyyyyy.ASSO ,LINK=DDASSOR1,SHARUPD=YES
/FILE ADAyyyyy.DATA ,LINK=DDDATAR1,SHARUPD=YES
/FILE ADAyyyyy.WORK ,LINK=DDWORKR1,SHARUPD=YES
/FILE ADAyyyyy.SAVE1 ,LINK=DDREST1,DEVICE=TAPE-C1,VOLUME=ADA001,-
/     STATE=FOREIGN.-
/     FCBTYPE=SAM,RECFORM=V,RECSIZE=,BLKSIZE=32768,LABEL=STD
/FILE ADAyyyyy.PLOGR1,LINK=DDPLOGR1,SHARUPD=YES
/FILE ADAyyyyy.PLOGR2,LINK=DDPLOGR2,SHARUPD=YES
/EXEC (ADARUN,ADA.MOD)
ADARUN PROG=ADASAV,DB=yyyyy,IDTNAME=ADABAS5B
ADASAV RESTORE OVERWRITE
/LOGOFF NOSPOOL
```

Restore Protection Log after an Interrupted RESTONL Function

In SDF Format:

```
/.ADASAV LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK * A D A S A V RESTORE PROTECTION LOG
/REMARK *
/ASS-SYSLST L.SAV.REPL
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDASSOR1,ADAyyyyy.ASSO,SHARE-UPD=YES
/SET-FILE-LINK DDDATAR1,ADAyyyyy.DATA,SHARE-UPD=YES
/SET-FILE-LINK DDWORKR1,ADAyyyyy.WORK,SHARE-UPD=YES
/SET-FILE-LINK DDPLOG,ADAyyyyy.PLOG
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADASAV,DB=yyyyy,IDTNAME=ADABAS5B
ADASAV RESTPLOG FILES=2,PLOGNUM=ppp,SYN1=43
/LOGOFF SYS-OUTPUT=DEL
```

In ISP Format:

```
/.ADASAV LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK * A D A S A V RESTORE PROTECTION LOG
/REMARK *
/SYSFILE SYSLST=L.SAV.REPL
/FILE ADA.MOD,LINK=DDLIB
/FILE ADAyyyyy.ASSO ,LINK=DDASSOR1,SHARUPD=YES
/FILE ADAyyyyy.DATA ,LINK=DDDATAR1,SHARUPD=YES
/FILE ADAyyyyy.WORK ,LINK=DDWORKR1,SHARUPD=YES
/FILE ADAyyyyy.PLOG ,LINK=DDPLOG
/EXEC (ADARUN,ADA.MOD)
ADARUN PROG=ADASAV,DB=yyyyy,IDTNAME=ADABAS5B
ADASAV RESTPLOG FILES=2,PLOGNUM=ppp,SYN1=43
/LOGOFF NOSPOOL
```

Restore Files from SAVE Datasets Created Offline

In SDF Format:

```
/.ADASAV LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK * A D A S A V RESTORE FILES, RESTORE DATABASE
/REMARK * FROM OFFLINE CREATED SAVE DATASETS
/REMARK *
/ASS-SYSLST L.SAV.REFM
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDASSOR1,ADAyyyyy.ASSO,SHARE-UPD=YES
/SET-FILE-LINK DDDATAR1,ADAyyyyy.DATA,SHARE-UPD=YES
/SET-FILE-LINK DDWORKR1,ADAyyyyy.WORK,SHARE-UPD=YES
/SET-FILE-LINK DDREST1,ADAyyyyy.SAVE1
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADASAV,DB=yyyyy,IDTNAME=ADABAS5B
ADASAV RESTORE FMOVE=2
/LOGOFF SYS-OUTPUT=DEL
```

In ISP Format:

```
/.ADASAV LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK * A D A S A V RESTORE FILES, RESTORE DATABASE
/REMARK * FROM OFFLINE CREATED SAVE DATASETS
/REMARK *
/SYSFILE SYSLST=L.SAV.REFM
/FILE ADA.MOD ,LINK=DDLIB
/FILE ADAyyyyy.ASSO ,LINK=DDASSOR1,SHARUPD=YES
/FILE ADAyyyyy.DATA ,LINK=DDDATAR1,SHARUPD=YES
/FILE ADAyyyyy.WORK ,LINK=DDWORKR1,SHARUPD=YES
/FILE ADAyyyyy.SAVE1 ,LINK=DDREST1
/EXEC (ADARUN,ADA.MOD)
ADARUN PROG=ADASAV,DB=yyyyy,IDTNAME=ADABAS5B
ADASAV RESTORE FMOVE=2
/LOGOFF NOSPOOL
```

OS/390 or z/OS

Dataset	DD Name	Storage	More Information
Associator	DDASSORn	disk	
Data Storage	DDDATARn	disk	
Work	DDWORKRn/ DDWORKmm	disk	Required if Adabas nucleus is not active
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 Manual*
ADASAV parameters	DDKARTE	reader	
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADASAV messages	DDDRUCK	printer	*Messages and Codes*

Notes:

1. *For RESTONL, the input SAVE tapes and the sequential protection log can be concatenated, using the name DDREST1.*
2. *Multiple Work datasets are specified as DD/WORKR1 to DD/WORKR9 and, beyond that, DD/WORK10 to DD/WORK99.*

ADASAV JCL Examples (OS/390 or z/OS)

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.Vvrs.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
//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 MVSJOBS dataset 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.Vvrs.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
//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 MVSJOBS dataset 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.Vvrs.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
//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 MVSJOBS dataset for this example.

Restore Files From SAVE Datasets 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.Vvrs.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
//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 MVSJOBS dataset for this example.

Restore Files From SAVE Datasets 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.Vvrs.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
//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 MVSJOBS dataset 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.Vvrs.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
//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 MVSJOBS dataset for this example.

VM/ESA or z/VM

Dataset	DD Name	Storage	More Information
Associator	DDASSORn	disk	
Data Storage	DDDATARn	disk	
Work	DDWORKRn/ DDWORKmm	disk	Required if Adabas nucleus not active
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 to 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	disk/terminal/ reader	*Operations Manual*
ADASAV parameters	DDKARTE	disk/terminal/ reader	
ADARUN messages	DDPRINT	disk/terminal/ printer	*Messages and Codes*
ADASAV messages	DDDRUCK	disk/terminal/ printer	*Messages and Codes*

Notes:

1. *For RESTONL, the input SAVE tapes and the sequential protection log can be concatenated, using the name DDREST1.*
2. *Multiple Work datasets are specified as DD/WORKR1 to DD/WORKR9 and, beyond that, DD/WORK10 to DD/WORK99.*

ADASAV Job Control Examples (VM/ESA or z/VM)

Save Database

```
DATADEF DDASSOR1,DSN=ADABASVv.ASSO,VOL=ASSOV1
DATADEF DDDATAR1,DSN=ADABASVv.DATA,VOL=DATAV1
DATADEF DDWORKR1,DSN=ADABASVv.WORK,VOL=WORKV1
DATADEF DDSAVE1,DSN=ADABASVv.COPY1,MODE=A
DATADEF DDDUAL1,DSN=ADABASVv.COPY2,MODE=A
DATADEF DDPRINT,DSN=ADASAV.DDPRINT,MODE=A
DATADEF DUMP,DUMMY
DATADEF DDDRUCK,DSN=ADASAV.DDDRUCK,MODE=A
DATADEF DDCARD,DSN=RUNSAV.CONTROL,MODE=A
DATADEF DDKARTE,DSN=ADASAV.CONTROL,MODE=A
ADARUN
```

Contents of RUNSAV CONTROL A1

```
ADARUN PROG=ADASAV,DEVICE=dddd,DB=yyyyy
```

Contents of ADASAV CONTROL A1

```
ADASAV SAVE TWOCOPIES
```

Restore Files from SAVE Datasets Created Online

```
DATADEF DDASSOR1,DSN=ADABASVv.ASSO,VOL=ASSOV1
DATADEF DDDATAR1,DSN=ADABASVv.DATA,VOL=DATAV1
DATADEF DDWORKR1,DSN=ADABASVv.WORK,VOL=WORKV1
DATADEF DDREST1,DSN=ADABASVv.SAVE,MODE=A
DATADEF DDPLOG,DSN=ADABASVv.SIBA,MODE=A
DATADEF DDPRINT,DSN=ADASAV.DDPRINT,MODE=A
DATADEF DUMP,DUMMY
DATADEF DDDRUCK,DSN=ADASAV.DDDRUCK,MODE=A
DATADEF DDCARD,DSN=RUNSAV.CONTROL,MODE=A
DATADEF DDKARTE,DSN=RESTONL.CONTROL,MODE=A
ADARUN
```

Contents of RUNSAV CONTROL A1

```
ADARUN PROG=ADASAV,DEVICE=dddd,DB=yyyyy
```

Contents of RESTONL CONTROL A1

```
ADASAV RESTONL FILES=2,PLOGNUM=ppp,SYN1=1
```

Restore Protection Log After Interrupted RESTONL Function

```
DATADEF DDASSOR1,DSN=ADABASVv.ASSO,VOL=ASSOV1
DATADEF DDDATAR1,DSN=ADABASVv.DATA,VOL=DATAV1
DATADEF DDWORKR1,DSN=ADABASVv.WORK,VOL=WORKV1
DATADEF DDPLOG,DSN=ADABASVv.PLOG,MODE=A
DATADEF DDPRINT,DSN=ADASAV.DDPRINT,MODE=A
DATADEF DUMP,DUMMY
DATADEF DDDRUCK,DSN=ADASAV.DDDRUCK,MODE=A
DATADEF DDCARD,DSN=RUNSAV.CONTROL,MODE=A
DATADEF DDKARTE,DSN=RESTFILE.CONTROL,MODE=A
ADARUN
```

Contents of RUNSAV CONTROL A1

```
ADARUN PROG=ADASAV,DEVICE=dddd,DB=yyyyy
```

Contents of RESTFILE CONTROL A1

```
ADASAV RESTPLOG FILES=2,PLOGNUM=ppp,SYN1=1
```

Restore Files from SAVE Datasets Created Offline

```
DATADEF DDASSOR1,DSN=ADABASVv.ASSO,VOL=ASSOV1
DATADEF DDDATAR1,DSN=ADABASVv.DATA,VOL=DATAV1
DATADEF DDWORKR1,DSN=ADABASVv.WORK,VOL=WORKV1
DATADEF DDREST1,DSN=ADABASVv.SAVE,MODE=A
DATADEF DDPRINT,DSN=ADASAV.DDPRINT,MODE=A
DATADEF DUMP,DUMMY
DATADEF DDDRUCK,DSN=ADASAV.DDDRUCK,MODE=A
DATADEF DDCARD,DSN=RUNSAV.CONTROL,MODE=A
DATADEF DDKARTE,DSN=RESTORE.CONTROL,MODE=A
ADARUN
```

Contents of RUNSAV CONTROL A1

```
ADARUN PROG=ADASAV,DEVICE=dddd,DB=yyyyy
```

Contents of RESTORE CONTROL A1

```
ADASAV RESTORE FMOVE=2
```

VSE/ESA

File	Symbolic Name	Storage	Logical Unit	Information
Associator	ASSORn	disk	see note 1	
Data Storage	DATARn	disk	see note 1	
Work	WORKRn/ WORKmm	disk	see note 1	Required if Adabas nucleus is not active
Recovery log (RLOG)	RLOGR1	disk		Required for ADARAI
Backup copy	SAVE1-8	tape disk	SYS011–SYS018 see note 1	Required for SAVE
Dual copy	DUAL1-8	tape disk	SYS021–SYS028 see note 1	Required for SAVE with two backup copies
Backup copy (input for RESTORE)	REST1–8	tape disk	SYS011–SYS018 see note 1	Required for RESTORE and RESTONL
Sequential protection log	PLOG	tape disk	SYS010 see note 1	Required for RESTONL and RESTPLOG
ADARUN parameters	SYSRDR CARD CARD	reader tape disk	SYSRDR SYS000 see note 1	
ADASAV parameters		reader	SYSIPT	
ADARUN messages		printer	SYSLST	
ADASAV messages		printer	SYS009	

Notes:

1. *Any programmer logical unit may be used.*
2. *For RESTONL, the input SAVE tapes and the sequential protection log can be concatenated, using the name REST1.*
3. *Multiple Work datasets are specified as DD/WORKR1 to DD/WORKR9 and, beyond that, DD/WORK10 to DD/WORK99.*

ADASAV JCS Examples (VSE/ESA)

See appendix B for descriptions of the VSE/ESA procedures (PROCs).

Refer to the following members for these examples:

Example	Member
Save database	ADASAV.X
Save database with two copies of output	ADASAVT.X
Restore database	ADASAVR.X
Restore files from save datasets created online	ADASAVRO.X
Restore protection log after an interrupted RESTONL function	ADASAVRP.X
Restore files from save datasets created offline	ADASAVRF.X

Save Database

```
* $$ JOB JNM=ADASAV,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADASAV
*       SAVE THE ENTIRE DATABASE
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// ASSGN SYS011,TAPE
// PAUSE MOUNT LOAD SAVE FILE ON TAPE cuu
// TLBL SAVE1,'EXAMPLE.DByyyyy.SAVE'
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADASAV,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADASAV SAVE
/*
/&
* $$ EOJ
```

Save Database with Two Copies of Output

```
* $$ JOB JNM=ADASAVT,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADASAVT
*       SAVE THE ENTIRE DATABASE CREATING TWO COPIES OF THE OUTPUT
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// PAUSE MOUNT LOAD SAVE TAPES ON cu1 AND cu2
// ASSGN SYS011,TAPE
// TLBL SAVE1,'EXAMPLE.DByyyyy.SAVE'
// ASSGN SYS021,TAPE
// TLBL DUAL1,'EXAMPLE.DByyyyy.SAVE.COPY'
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADASAV,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADASAV SAVE TWOCOPIES
/*
/&
* $$ EOJ
```

Restore Database

```
* $$ JOB JNM=ADASAVR,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADASAVR
*       RESTORE THE ENTIRE DATABASE
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// ASSGN SYS011,TAPE
// PAUSE MOUNT LOAD SAVE FILE ON TAPE cuu
// TLBL REST1,'EXAMPLE.DByyyyy.SAVE'
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADASAV,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADASAV RESTORE OVERWRITE
/*
/&
* $$ EOJ
```

Restore Files from Save Datasets Created Online

```
* $$ JOB JNM=ADASAVRO,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADASAVRO
*       RESTORE FILES FROM SAVE DATA SETS CREATED ONLINE
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// PAUSE MOUNT LOAD SAVE FILE ON TAPE cu1 AND PLOG ON TAPE cu2
// ASSGN SYS011,TAPE
// TLBL REST1,'EXAMPLE.DByyyyy.SAVE'
// ASSGN SYS010,TAPE
// TLBL PLOG,'EXAMPLE.DByyyyy.PLOG5'
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADASAV,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADASAV RESTONL FILES=2,PLOGNUM=ppp,SYN1=1
/*
/&
* $$ EOJ
```

Restore Protection Log after an Interrupted RESTONL Function

```
* $$ JOB JNM=ADASAVRP,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADASAVRP
*       RESTORE THE PROTECTION LOG AFTER AN INTERRUPTED RESTONL
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// PAUSE MOUNT LOAD PLOG FILE ON TAPE cuu
// ASSGN SYS010,TAPE
// TLBL PLOG,'EXAMPLE.DByyyyy.PLOG5'
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADASAV,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADASAV RESTPLOG PLOGNUM=ppp,SYN1=1
/*
/&
* $$ EOJ
```

Restore Files From Save Datasets Created Offline

```
* $$ JOB JNM=ADASAVRF,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADASAVRF
*       RESTORE A FILE TO ANY RABNS FROM AN OFFLINE SAVE
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// ASSGN SYS011,TAPE
// PAUSE MOUNT LOAD SAVE FILE ON TAPE cuu
// TLBL REST1,'EXAMPLE.DByyyyy.SAVE'
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADASAV,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADASAV RESTORE FMOVE=2
/*
/&
* $$ EOJ
```

ADASEL : SELECT PROTECTION DATA

Functional Overview

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 dataset (DDDRUCK / DRUCK) or to a user-specified output dataset.

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.

You can select before-images, after-images, or both for new, updated, and deleted records. You can also select data written to the protection log with an Adabas C5 command.

Note:
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.*

Syntax

Unlike other Adabas utilities, ADASEL does not require the utility name at the beginning of each parameter line. A selection request includes the following parts:

- the keyword “SELECT”, followed by the selection option and the file number;
- optional clauses and statements that specify additional selection criteria;
- one or more output instructions;
- the keyword “END”.

An overview of the ADASEL syntax is shown below.

```
SELECT { ALL                [RECORDS] [ { FROM } ] [FILE] file-number
         BEFORE IMAGE                  { IN   }
         BI
         AFTER IMAGE
         AI
         NEW
         DELETED
         NEWDEL
         UPDATED }

[STARTING FROM date-time]

[ENDING AT date-time]

[ WITH value-criterion [ { AND } value-criterion ] . . . ]
                       { OR  }

{ Output-instruction }
{ IF-statement       } . . .

END
```

You can code multiple selection requests. Each request begins with “SELECT” and ends with “END”.

Example:

```
SELECT ALL RECORDS FROM FILE 1
  DISPLAY AA BB CC
END

SELECT BEFORE IMAGE FILE 2
  OUTPUT TO EXPA1
END
```

SELECT Statement

```
SELECT { ALL              } [RECORDS] [{FROM} ] [FILE] file-number
       { BEFORE IMAGE     }           [{IN}   ]
       { BI               }
       { AFTER IMAGE      }
       { AI               }
       { NEW              }
       { DELETED          }
       { NEWDEL           }
       { UPDATED          }
```

After the keyword **SELECT**, specify one of the following selection options:

Option	Records Selected
ALL	Before-images derived from A1 (update) and E1 (delete) commands; after-images derived from A1 and N1 (add) commands.
BEFORE IMAGE | BI	Before-images derived from A1 and E1 commands.
AFTER IMAGE | AI	After-images derived from A1 and N1 commands.
NEW	After-images derived from N1 commands.
DELETED	Before-images derived from E1 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.

file-number

Specify the Adabas file for which protection log data is to be selected. Valid file numbers are 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.

Additional Selection Criteria

You can use the STARTING FROM, ENDING AT, and WITH clauses and the IF statement to specify additional selection criteria. These optional clauses and statement take precedence over the selection option.

```
[STARTING FROM date-time]
[ENDING AT date-time]
[WITH value-criterion [{AND} value-criterion]... ]
                       {OR}
{output-instruction}
{IF-statement       }...
```

The “STARTING FROM” and “ENDING AT” clauses restrict selections to records added, updated, or deleted within a time range. The “date/time” variable is discussed on page 389.

The “WITH” clause is used to select records that satisfy the value-criteria specified. Multiple conditions can be specified using the logical operators “AND” and “OR”. The “WITH” clause is discussed on page 390.

The “IF” statement is used to select records and execute output instructions on a conditional basis. The “IF” statement is discussed on page 391.

The syntax of the “value-criterion” variable used in both the “WITH” clause and the “IF” statement is described starting on page 392.

Output instructions are described starting on page 398.

date-time

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 “WITH” clause is 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, **each** condition 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 starting on page 392.

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 is used to select records and execute output instructions on a conditional basis.

By default, ADASEL permits up to 20 nested “IF” statements. For information about changing the default, see the section **Overriding ADASEL Defaults with Global Parameters** on page 407.

```
IF { value-criterion      } [ AND { value-criterion      } ]
   { field-name CHANGES   }   OR { field-name CHANGES   }

   [THEN] { output-instruction } [ ELSE { output-instruction } ]
          { IF-statement        }       { IF-statement        }
          { DO-group            }       { DO-group            }
          { IGNORE              }       { IGNORE              }
```

The syntax of the “value-criterion” variable is described starting on page 392. Output instructions are described starting on page 398.

A “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** starting on page 392, particularly in the subsection **Indexes for MUs and PE Member Fields** on page 397.

The syntax for DO group is as follows:

```
DO  output-instruction . . .  DOEND
```

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.

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:

```
{ field-name }
{ ISN        }
{ USERDATA   } = value  [THRU value] [BUT NOT value [THRU value]]
{ USERID     }
{ USERTID    }

{ field-name }  { >  }
{ ISN        }  { >= }
{ USERDATA   }  { <= }  value
{ USERID     }  { <  }
{ USERTID    }  { ¬= }
```

The "BUT NOT" clause excludes a value or subrange of values from the range specified in the equality (=).

Object of the Comparison

ADASEL can compare a value or range of values to 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), 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 **Indexes for MUs and PE Member Fields** on page 397.
- ISN; that is, the Adabas internal sequence number of the record.
- 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		
Field Format in FDT	**Valid Formats**	**Max. Bytes**	**Max. Digits**
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
Wide-character	Hexadecimal	253	506

***** Excluding minus sign

The default formats and maximum lengths (in bytes) for other items are as follows:

		Criterion Value	
Item	**Default Format**	**Valid Formats**	**Max. Length**
ISN	Binary	Decimal, hexadecimal	4
USERDATA	Alphanumeric	Alphanumeric, hexadecimal	30
USERID	Binary	Decimal, hexadecimal	8
USERTID	Binary	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 –2,147,483,648 through 2,147,483,647.

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 “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” are 1–191.

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

Output instructions include DISPLAY, OUTPUT, SKIP, and NEWPAGE. At least one output instruction is required. Multiple output instructions can be specified, and an output instruction can be included as part of an “IF” statement. The syntax is shown below:

```
[ DISPLAY  item . . .  ]
[ OUTPUT [{ [WITH]   LOGINFO | EXTENDED }] [TO]   EXPAn ]
[ SKIP n {LINE | LINES}  ]
[ NEWPAGE ]
```

The DISPLAY instruction is discussed on page 399; the OUTPUT instruction is discussed on page 403. SKIP and NEWPAGE are discussed on page 406.

DISPLAY Instruction

"DISPLAY" writes the output report to DDDRUCK / DRUCK. The syntax specifies one or more output types. When specifying multiple output types, they are separated by at least one space:

```
                 field-name   [HEX]
                 ISN   [HEX]
                 USERDATA   [HEX]
DISPLAY        { USERTID   [HEX]   }  ...
                 USERID   [HEX]
                 NOHEADER
                 'text'
```

—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 on page 401.
HEX	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.
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.

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
```

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
```

Valid index values are 1–191. 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.

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 . . .
MUi	occurrence “i” of the MU
MUi–j	occurrences “i” through “j” of the MU
MUi–N	all occurrences of the MU, starting with occurrence “i”
ABk	field AB in occurrence “k” of the PE to which the field belongs
ABk–l	field AB in occurrences “k” through “l” of the PE
ABk–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
MBk–l(i)	occurrence “i” of MB in occurrences “k through “l” of the PE
MBk–l(i–j)	occurrences “i” through “j” of MB in occurrences “k” through “l” of the PE
MBk–l(i–N)	all occurrences of MB (starting with occurrence “i”) in occurrences “k” through “l” of the PE
MBk–N(i–j)	occurrences “i” through “j” of MB in all occurrences of the PE (starting with occurrence “k” of the PE)
MBk–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 dataset.

```
OUTPUT [ { [WITH] LOGINFO } ] [TO] EXPAn
       [ { EXTENDED       }
```

Up to 20 output datasets are permitted. The output dataset is specified in the “EXPAn” parameter and the DDEXPAn / EXPAn job control statement.

Example:

Write the before-images of all updated or deleted records to dataset DDEXPA1 / EXPA1:

```
SELECT BEFORE IMAGE FILE 2
  OUTPUT TO EXPA1
END
```

Output Record Format

The format of the output record depends on whether the “LOGINFO” or “EXTENDED” parameter is specified. LOGINFO (see page 404) and EXTENDED (see page 405) are used to display additional information.

Fields common to all output records are shown below. Values in parentheses are field locations when LOGINFO (bytes 32–38) or EXTENDED (bytes 64–70) are specified.

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
6–7	Adabas file number (binary)

Bytes	Description
8–9(32–33, 64–65)	decompressed record length (including this length field and the ISN)
10–13(34–37, 66–69)	ISN (binary) or user data from a C5 command
14 (38, 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.

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.

LOGINFO

When "LOGINFO" is specified, the following additional information is included in each record:

Bytes	Description
8–15	ID of the user who added, deleted, or updated the record
16–19	low-order four bytes of the 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)

EXTENDED

When "EXTENDED" is specified, the following additional information is included in each record:

Bytes	Description
8–15	ID of the user (ETID) who added, deleted, or updated the record
16–23	low-order eight bytes of the terminal ID 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	reserved
38–41	transaction number
42–63	reserved

Output Dataset Designation

The "EXPAn" parameter identifies the output dataset. The value of "n" must match the value in the DDEXPAn / EXPAn JCL statement. Valid output dataset numbers are 1–20 with no leading zeros:

Valid statement:	**OUTPUT TO EXPA3**
Invalid statement:	**OUTPUT TO EXPA03**

The same rule applies to the DD/EXPAn JCL statement.

Example:

Select all records for file 1. Write decompressed records in which the BA field contains “SMITH” or “SMYTH” to DDEXPA1 / EXPA1. Write all others to DDEXPA2 / EXPA2:

```
SELECT ALL RECORDS FROM FILE 1
IF BA = 'SMITH' OR BA = 'SMYTH'
  THEN OUTPUT TO EXPA1
ELSE
  OUTPUT TO EXPA2
END
```

NEWPAGE and SKIP Instructions

The “NEWPAGE” and “SKIP” instructions control page formatting:

- NEWPAGE forces a page eject before displaying the next line; and
- SKIP prints the specified number of blank lines before displaying the next line of data.

Example:

```
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
ELSE DO
  SKIP 2 LINES
  DISPLAY BA BB BC
  DOEND
END
```

Overriding ADASEL Defaults with Global Parameters

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, SET GLOBALS must be the first ADASEL input statement. Comment statements can precede it. The statement has 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.

Example:

```
SET GLOBALS LST=15000 NF=15
SET GLOBALS LS=132
```

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.

Parameter	Description
LST={len | <u>12000</u> }	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.

Parameter	Description
NCFLD={n | 10 } NCUPD={n | 10 }	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 }	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 (DDEXPAn / EXPAn); although more than 20 files can be processed during an ADASEL run, a maximum of 20 output files can be written.
NIF={n | 20 }	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.
NV={n | 100 }	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**
LPV={n | 0 }	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.

Parameter	Description
PS={n | 60 }	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.
LS={n | 80 }	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.

JCL/JCS Requirements and Examples

This section describes the job control information required to run ADASEL with BS2000, OS/390 or z/OS, VM/ESA or z/VM, and VSE/ESA systems and shows examples of each of the job streams.

BS2000

Dataset	Link Name	Storage	More Information
Protection log	DDSIIN	tape/disk	Sequential log
Selected data	DDEXPAn	tape/disk	Output by ADASEL
Associator	DDASSORn	disk	
ADARUN parameters	SYSDTA/DDCARD		*Operations Manual*
ADASEL parameters	SYSDTA/DDKARTE		
ADARUN messages	SYSOUT/DDPRINT		*Messages and Codes*
ADASEL messages	SYSLST/DDDRUCK		*Messages and Codes*

ADASEL JCL Example (BS2000)

In SDF Format:

```
/.ADASEL LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK * A D A S E L ALL FUNCTIONS
/REMARK *
/DELETE-FILE SEL.AUS
/SET-JOB-STEP
/CREATE-FILE SEL.AUS,PUB(SPACE=(48,48))
/SET-JOB-STEP
/ASS-SYSLST L.SEL
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDASSOR1,ADAyyyyy.ASSO
/SET-FILE-LINK DDSIIN,ADAyyyyy.SIBA
/SET-FILE-LINK DDEXPA1,SEL.AUS
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADASEL,DB=yyyyy,IDTNAME=ADABAS5B
SELECT ALL FROM FILE 11
DISPLAY AA BB BA BC CA CC
END
/LOGOFF SYS-OUTPUT=DEL
```

In ISP Format:

```
/.ADASEL LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK * A D A S E L ALL FUNCTIONS
/REMARK *
/SYSFILE SYSLST=L.SEL
/FILE ADA.ASSO ,LINK=DDASSOR1
/FILE ADA.MOD ,LINK=DDLIB
/FILE ADAyyyyy.SIBA ,LINK=DDSIIN
/FILE SEL.AUS ,LINK=DDEXPA1 ,SPACE=(48,48)
/EXEC (ADARUN,ADA.MOD)
ADARUN PROG=ADASEL,DB=yyyyy,IDTNAME=ADABAS5B
SELECT ALL FROM FILE 11
DISPLAY AA BB BA BC CA CC
END
/LOGOFF NOSPOOL
```

OS/390 or z/OS

Dataset	DD Name	Storage	More Information
Protection log	DDSIIN	tape/disk	Sequential log
Selected data	DDEXPAn	tape/disk	Output by ADASEL
Associator	DDASSORn	disk	
ADARUN parameters	DDCARD	reader	*Operations Manual*
ADASEL parameters	DDKARTE	reader	
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADASEL messages	DDDRUCK	printer	*Messages and Codes*

ADASEL JCL Example (OS/390 or z/OS)

```
//ADASEL    JOB
//*
//*   ADASEL:
//*      SELECT PROTECTION DATA
//*
//SEL       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
//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 MVSJOBS dataset for this example.

VM/ESA or z/VM

Dataset	DD Name	Storage	More Information
Protection log	DDSIIN	tape/disk	Sequential log
Selected data	DDEXPAn	tape/disk	
Associator	DDASSORn	disk	
ADARUN parameters	DDCARD	disk/terminal/reader	*Operations Manual*
ADASEL parameters	DDKARTE	disk/terminal/reader	
ADARUN messages	DDPRINT	disk/terminal/printer	*Messages and Codes*
ADASEL messages	DDDRUCK	disk/terminal/printer	*Messages and Codes*

Example (VM/ESA or z/VM)

```
DATADEF DDASSOR1,DSN=ADABASVv.ASSO,VOL=ASSOV1
DATADEF DDSIIN,DSN=ADABASVv.SIBA,UNIT=181,VOL=SIBAXX
DATADEF DDEXPA1,DSN=ADABASVv.EXPA1,MODE=A
DATADEF DDPRINT,DSN=ADASEL.DDPRINT,MODE=A
DATADEF DUMP,DUMMY
DATADEF DDDRUCK,DSN=ADASEL.DDDRUCK,MODE=A
DATADEF DDCARD,DSN=RUNSEL.CONTROL,MODE=A
DATADEF DDKARTE,DSN=ADASEL.CONTROL,MODE=A
LOAD ADARUN ( START
```

Contents of RUNSEL CONTROL A1

```
ADARUN PROG=ADASEL,DEVICE=dddd,DB=yyyyy
```

Contents of ADASEL CONTROL A1

```
SELECT ALL FROM FILE 11
   DISPLAY ISN
END
```

VSE/ESA

File	File Name	Storage	Logical Unit	More Information
Protection log	SIIN	tape disk	SYS010 *	Sequential log
Selected data	EXPA1–20	tape disk	SYS011–SYS030 *	Output by ADASEL
Associator	ASSORn	disk	*	
ADARUN parameters	SYSRDR CARD	reader tape/disk		*Operations Manual*
ADASEL parameters	SYSIPT	reader		
ADARUN messages	SYSLST	printer		*Messages and Codes*
ADASEL messages	SYS009	printer		*Messages and Codes*

***** Any programmer logical unit may be specified.

ADASEL JCS Example (VSE/ESA)

```
* $$ JOB JNM=ADASEL,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADASEL
*       SELECT PROTECTION DATA
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// PAUSE MOUNT LOAD INPUT PLOG FILE ON TAPE cuu
// ASSGN SYS010,TAPE
// TLBL SIIN,'EXAMPLE.DByyyyy.PLOG5'
// DLBL EXPA1,'EXAMPLE.ADAyyyyy.EXPA1'
// EXTENT SYS015,,,,ssss,nnnn
// ASSGN SYS015,DISK,VOL=vvvvvv,SHR
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADASEL,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
*    SELECT UPDATES FOR FILE NUMBER 1
*
SELECT ALL FROM FILE 1
  OUTPUT TO EXPA1
```

```
END
/*
/&
* $$ EOJ
```

Refer to member ADASEL.X for this example.

ADAULD : UNLOAD

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.

Note:
An interrupted ADAULD UNLOAD FILE run must be reexecuted from the beginning.

UNLOAD FILE : Unload Specified File

```
ADAULD  [UNLOAD]  FILE=file-number
                  [CODE=cipher-key]
                  [DDISN]
                  [ERRLIM={error-threshold-count|1}]
                  [ETID=multiclient-file-owner-id]
                  [LPB={prefetch-buffer-size|based-on-ADARUN-lu}]
                  [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 | ADARUN-device } ]
                      [TEMPSIZE=size ]
                  [SELCRIT='selection-criteria',
                            SELVAL='values-for-selection-criteria']
                     [STARTISN=value]
                  [SORTSEQ= { descriptor [,MU] [,NU]
                              ISN [,STARTISN=value]
                              physical-sequence }]
                  [TEST]
                  [UTYPE={EXU|EXF}]
```

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 DD/ISN Output File of Unloaded ISNs

Specifying the DDISN parameter instructs ADAULD to write the list of unloaded ISNs to the sequential output file DD/ISN. DD/ISN 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 DD/ISN 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 DD/ISN 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 (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 occurs. The default is 4000 bytes; the maximum allowed is 32760 bytes.

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 dataset.

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 an error is encountered while the function is running, the utility 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.

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 DD/OUT1 or DD/OUT2 output file selection. For additional information, see the *Adabas DBA Reference Manual*. 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 DD/PLOG sequential input dataset.

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.

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 blocksize 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. If the online save tape was created using ADASAV version 5.1, the parameters PLOGNUM and SYN1 or SYN4 must also be specified. PLOGNUM and SYN1 or SYN4 may be specified for online save tapes created using ADASAV version 5.2 or above to override the information included on the tape.

For more information, see the section **Processing a Save Tape as Input** on page 425.

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** on page 428.

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 Manual.

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 Manual*.

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.

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* manual.

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’
```

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, superdescriptor, 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 Manual* 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 DD/PLOG sequential input dataset.

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.

TEMPDEV : Temporary Storage Device Type

When SAVETAPE is specified and an “online” save tape is to be used as input to ADAULD, a temp dataset is used to store intermediate data during processing. The TEMPDEV parameter indicates the device type to be used for the temp dataset. This parameter is required only if the device type to be used is different from the standard device type assigned to Temp by the ADARUN DEVICE parameter.

The blocksize of the temp dataset must be at least as large as the largest Data Storage blocksize of the file to be unloaded, plus 16 bytes.

TEMPSIZE : Temporary Storage Size

TEMPSIZE specifies the size of the temp dataset for the file. The size can be either in cylinders or blocks (followed by “B”).

The temp dataset 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 dataset. If the temp dataset is too small, ADAULD error–136 (temp dataset too small) is returned.

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.

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.

ADAULD Input Processing

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 DD/SAVE 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 hyperexits must be specified in the ADARUN parameters for ADAULD. If the hyperexit routines are no longer available, the file must be unloaded with MODE=SHORT specified. See the *Adabas DBA Reference Manual* for more information about hyperexits.

For an “online” save tape:

- the corresponding protection log is expected as a DD/PLOG sequential input dataset.
- a temp (DD/TEMPR1) dataset 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 DD/DRUCK 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 U L D   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    Vvr                         *
*  DBID                 nnnnn                       *
*  DSID                 1 / 0 / yyyy-mm-dd  hh:mm:ss *
*  PLOG SESSION NR      17                          *
*  SYN1/4 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 dataset. The location of a Data Storage block on the temp dataset 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 dataset, they are actually unloaded to DD/OUT1 or DD/OUT2. 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 DD/SAVE and DD/PLOG input datasets, 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).

ADAULD Output Processing

ADAULD unloads the records in the specified sequence. The unloaded records are written to one or both of two sequential datasets: DD/OUT1 and DD/OUT2. Writing to these output datasets 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 DD/ISN. Only one DD/ISN file is created, containing the superset of ISNs written to either or both of DD/OUT1 and DD/OUT2. ISNs that are rejected by userexit 9 are not written to DD/ISN.

DD/ISN 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 DD/ISN is displayed in the ADAULD statistics on the DD/DRUCK 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          =        1000
NUMBER OF RECORDS WRITTEN       =        1000
RECORDS WRITTEN TO DDOUT1       =        1000
RECORDS WRITTEN TO DDOUT2       =           0
RECORDS REJECTED BY USEREXIT-9  =           0
NUMBER OF ISNS WRITTEN TO DDISN =        1000
```

The number of ISNs written to DD/ISN should always be the number of records read minus the number of records rejected by user exit 9.

The ISNs on the DD/ISN file are ISNs as visible to applications; that is, the internal ISN as stored in a Data Storage record plus MINISN–1.

ADAULD User Exit 9

User exit 9 is called (when present) for each record selected before writing the record to the output dataset. The user exit is supplied with the record address, and returns an action code as follows:

1 Write record to DD/OUT1;
2 Write record to DD/OUT2;
3 Write record to DD/OUT1 and DD/OUT2;
I Ignore this record.

The above datasets must have the same blocksize. See the *Adabas DBA Reference Manual* for more information about user exits.

JCL/JCS Requirements and Examples

This section describes the job control information required to run ADAULD with BS2000, OS/390 or z/OS, VM/ESA or z/VM, and VSE/ESA systems and shows examples of each of the job streams.

BS2000

Dataset	Link 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
ADARUN parameters	SYSDTA/DDCARD		*Operations Manual*
ADAULD parameters	SYSDTA/DDKARTE		
ADARUN messages	SYSOUT/DDPRINT		*Messages and Codes*
ADAULD messages	SYSLST/DDDRUCK		*Messages and Codes*

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 (BS2000)

Unload from Database

In SDF Format:

```
/.ADAULD LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK * A D A U L D NON-SAVETAPE FUNCTIONS
/REMARK *
/DELETE-FILE ADAyyyyy.OUT1
/SET-JOB-STEP
/CREATE-FILE ADAyyyyy.OUT1,PUB(SPACE=(480,48))
/SET-JOB-STEP
/ASS-SYSLST L.ULD
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDOUT1,ADAyyyyy.OUT1
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADAULD,DB=yyyyy,IDTNAME=ADABAS5B
ADAULD FILE=1,SORTSEQ=AA
/LOGOFF SYS-OUTPUT=DEL
```

In ISP Format:

```
/.ADAULD LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK * A D A U L D NON-SAVETAPE FUNCTIONS
/REMARK *
/SYSFILE SYSLST=L.ULD
/FILE ADA.MOD ,LINK=DDLIB
/FILE ADAyyyyy.OUT1 ,LINK=DDOUT1 ,SPACE=(480,48)
/EXEC (ADARUN,ADA.MOD)
ADARUN PROG=ADAULD,DB=yyyyy,IDTNAME=ADABAS5B
ADAULD FILE=1,SORTSEQ=AA
/LOGOFF NOSPOOL
```

Unload from Offline Save Tape

In SDF Format:

```
/.ADAULD LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK * A D A U L D SAVETAPE FUNCTION
/REMARK *
/DELETE-FILE ADAyyyyy.OUT1
/SET-JOB-STEP
/CREATE-FILE ADAyyyyy.OUT1,PUB(SPACE=(480,48))
/SET-JOB-STEP
/ASS-SYSLST L.ULD
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDOUT1,ADAyyyyy.OUT1
/SET-FILE-LINK DDSAVE,ADAyyyyy.SAVE
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADAULD,DB=yyyyy,IDTNAME=ADABAS5B
ADAULD FILE=1,SAVETAPE
/LOGOFF SYS-OUTPUT=DEL
```

In ISP Format:

```
/.ADAULD LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK * A D A U L D SAVETAPE FUNCTION
/REMARK *
/SYSFILE SYSLST=L.ULD
/FILE ADA.MOD ,LINK=DDLIB
/FILE ADAyyyyy.OUT1 ,LINK=DDOUT1 ,SPACE=(480,48)
/FILE ADAyyyyy.SAVE ,LINK=DDSAVE
/EXEC (ADARUN,ADA.MOD)
ADARUN PROG=ADAULD,DB=yyyyy,IDTNAME=ADABAS5B
ADAULD FILE=1,SAVETAPE
/LOGOFF NOSPOOL
```

OS/390 or z/OS

Dataset	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 Manual*
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 (OS/390 or z/OS)

Unload a File

```
//ADAULD    JOB
//*
//*   ADAULD:
//*      UNLOAD A FILE
//*
//ULD       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
//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
00000100
/*
//
```

Refer to ADAULD in the MVSJOBS dataset 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.Vvrs.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
//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
00000100
/*
//
```

Refer to ADAULDS in the MVSJOBS dataset for this example.

VM/ESA or z/VM

Dataset	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	disk/terminal/printer	*Messages and Codes*
ADARUN messages	DDPRINT	disk/terminal/printer	*Messages and Codes*
ADARUN parameters	DDCARD	disk/terminal/reader	*Operations Manual*
ADAULD parameters	SYSIPT	disk/terminal/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.

Examples (VM/ESA or z/VM)

Unload from Database

```
DATADEF DDOUT1,DSN=ADABASVv.ULD1,UNIT=181,VOL=ULDF1
DATADEF DDPRINT,DSN=ADAULD.DDPRINT,MODE=A
DATADEF DUMP,DUMMY
DATADEF DDDRUCK,DSN=ADAULD.DDDRUCK,MODE=A
DATADEF DDCARD,DSN=RUNULD.CONTROL,MODE=A
DATADEF DDKARTE,DSN=ADAULD.CONTROL,MODE=A
ADARUN
```

Contents of RUNULD CONTROL A1:

```
ADARUN PROG=ADAULD,DEVICE=dddd,DB=yyyyy
```

Contents of ADAULD CONTROL A1:

```
ADAULD FILE=1,SORTSEQ=AA
```

Unload from “Offline” Save Tape

```
DATADEF DDOUT1,DSN=ADABASVv.ULD1,UNIT=181,VOL=ULDF1
DATADEF DDSAVE,DSN=ADABASVv.ULD1,UNIT=181,VOL=ULDF1
DATADEF DDPRINT,DSN=ADAULD.DDPRINT,MODE=A
DATADEF DUMP,DUMMY
DATADEF DDDRUCK,DSN=ADAULD.DDDRUCK,MODE=A
DATADEF DDCARD,DSN=RUNULD.CONTROL,MODE=A
DATADEF DDKARTE,DSN=ADAULD.CONTROL,MODE=A
ADARUN
```

Contents of RUNULD CONTROL A1:

```
ADARUN PROG=ADAULD,DEVICE=dddd,DB=yyyyy
```

Contents of ADAULD CONTROL A1:

```
ADAULD FILE=1,SAVETAPE
```

VSE/ESA

File	Symbolic Name	Storage	Logical Unit	More Information
Unloaded data	OUT1	tape disk	SYS010 see note 1	Output by ADAULD (see note 2)
Unloaded data	OUT2	tape disk	SYS011 see note 1	Output by ADAULD (see note 2)
Unloaded ISNs	ISN	tape disk	SYS012 see note 1	Required with DDISN
Save tape	SAVE	tape disk	SYS013 see note 1	Required with SAVETAPE
Sequential PLOG	PLOG	tape disk	SYS014 see note 1	Required for online save tapes
Temp area	TEMPR1	disk	see note 1	Required for online save tapes
Recovery log (RLOG)	RLOGR1	disk	see note 1	Required for ADARAI
Messages	SYSLST	printer		*Messages and Codes*
ADARUN parameters	SYSRDR CARD	reader/ tape/disk		*Operations Manual*
ADAULD parameters	SYSIPT	reader		

Notes:

1. *Any programmer logical unit can be used.*
2. *OUT1 and OUT2 must have the same block size; otherwise, an ADAULD error will occur. OUT2 is required only if NUMOUT=2 is specified.*

ADAULD JCS Examples (VSE/ESA)

See appendix B for descriptions of the VSE procedures (PROCs).

Unload a File from a Database

```
* $$ JOB JNM=ADAULD,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
```

```
// JOB ADAULD
*       UNLOAD A FILE
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// ASSGN SYS004,DISK,VOL=vvvvvv,SHR
// DLBL OUT1,'EXAMPLE.ADA99.OUT1'
// EXTENT SYS004,,,,ssss,nnnn
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADAULD,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADAULD FILE=1,SORTSEQ=AA
/*
/&
* $$ EOJ
```

Refer to member ADAULD.X for this example.

Unload a File from Save Tape Created Offline

```
* $$ JOB JNM=ADAULDS,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADAULDS
*       UNLOAD A FILE FROM AN OFFLINE SAVE TAPE
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// ASSGN SYS004,DISK,VOL=vvvvvv,SHR
// DLBL OUT1,'EXAMPLE.ADA99.OUT1'
// EXTENT SYS004,,,,ssss,nnnn
// ASSGN SYS013,TAPE
// PAUSE MOUNT LOAD SAVE FILE ON TAPE cuu
// TLBL SAVE,'EXAMPLE.DByyyyy.SAVE'
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADAULD,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADAULD FILE=1,SAVETAPE
/*
/&
* $$ EOJ
```

Refer to member ADAULDS.X for this example.

ADAVAL : VALIDATE THE DATABASE

Functional Overview

The ADAVAL utility validates any or all files within an Adabas database except the checkpoint and security files.

ADAVAL compares the actual descriptor values contained in the records in Data Storage with the corresponding values stored in the Associator to ensure that the Associator and Data Storage are synchronized, and that there are no values missing from the Associator.

Before running ADAVAL, the consistency of the inverted lists should be checked with the ADAICK utility.

Note:
If ADAICK has been run and errors occurred, ***do not*** *run ADAVAL until the cause of the ADAICK error has been corrected. This ADAVAL run restriction applies for any ADAICK error except ADAICK WARNING-163.*

The Adabas nucleus must be running when executing ADAVAL. ADAVAL assigns EXF (exclusive use) status to all files to be validated, making them unavailable to other utilities or users. If ADAVAL specifies a file currently in use, an error message is issued and operation stops. ADAVAL returns condition code 4 if any errors are found.

ADAVAL prints a list of all fields compared and the ISNs rejected during validation on SYSOUT (DD/DRUCK). The normal ADAVAL output is shown under **Example of ADAVAL Output** on page 443.

If desired, rejected ISNs can also be output to a sequential dataset (DD/FEHL). The first record on DD/FEHL is always as follows:

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’ADAVAL’)
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 DD/FEHL records have the following format (items shown with "*" are also in the normal SYSOUT and DD/DRUCK 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
15, on*	Descriptor value

VALIDATE : Validate Data Storage and Associator

```
ADAVAL  VALIDATE     FILE=file-list
                     SORTSIZE=size
                     TEMPSIZE=size
                     [CODE=cipher-code]
                     [DESCRIPTOR='descriptor-list']
                     [LPB={prefetch-buffer-length|based-on-ADARUN-lu}]
                     [LRECL={record-buffer-length|4000}]
                     [LWP={work-pool-size|1048576}]
                     [NOUSERABEND]
                     [PASSWORD=password]
                     [SORTDEV={device-type|ADARUN -device}]
                     [TEMPDEV={device-type|ADARUN- device}]
```

The VALIDATE function validates the contents of the Data Storage against the values in the Associator. This is done by issuing commands to create a DVT that is validated against each corresponding value in the indices.

Note:
ADAVAL VALIDATE cannot be performed on the checkpoint or security files.

Essential Parameters

FILE : Files to Be Validated

FILE specifies a one or more Adabas file numbers and/or file ranges. A maximum of 1000 files may be specified.

Continuation for a file list is as follows:

```
ADAVAL  VALIDATE  FILE=1–10,15
ADAVAL            FILE=13,31–35
```

ADAVAL will concatenate the file list for each specification of the FILE parameter.

SORTSIZE : Sort Area Size

SORTSIZE specifies the number of blocks or cylinders available for the sort dataset. If specifying blocks, the value must be followed by “B” (for example, 2000B). A block value is automatically “rounded up” to the next full cylinder. See the *Adabas DBA Reference Manual* for information about estimating the SORTSIZE value.

TEMPSIZE : Temporary Storage Area Size

TEMPSIZE specifies the number of blocks or cylinders available for the temp dataset. If specifying blocks, the value must be followed by “B” (for example, 2000B). A block value is automatically “rounded up” to the next full cylinder. See see the section **LOAD File Space Allocation** on page NO TAG of the *Adabas Utilities Manual, Volume I*, for information about estimating the TEMPSIZE value.

Optional Parameters

CODE : Cipher Code

The CODE parameter is required if the file or file(s) being validated are enciphered.

DESCRIPTOR : List of Descriptors to Validate

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

The following is an example of DESCRIPTOR use:

```
ADAVAL  VALIDATE  FILE=5,DESCRIPTOR='AA,CC,BB'
```

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 current ADARUN LU value. ADAVAL VALIDATE may reduce the LPB value below that specified if the LU value is too small.

LRECL : Maximum Descriptor Value

LRECL specifies the maximum length of all descriptor values in any record of the file being validated. This length is used by ADAVAL to create a temporary record buffer. If the LRECL value is too small, response code 53 occurs when an oversized record is found. The default for LRECL is 4000 bytes; the maximum length allowed is 32760 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 ADAVAL 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 dataset, a smaller LWP value should be specified when validating relatively small files.

The minimum work pool size depends on the Sort dataset's device type:

Sort Device	Minimum LWP Bytes	Minimum LWP Kilobytes
2000	106496	104K
2314	090112	88K
3375	131072	128K
3380	139264	136K
3390	159744	156K

NOUSERABEND : Termination without ABEND

When an error is encountered while the function is running, the utility 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.

PASSWORD : Files Password

This parameter is required if the file or file(s) to be validated are password-protected.

SORTDEV : Sort Device Type

The SORTDEV parameter indicates the device type to be used for the sort dataset that ADAVAL uses to sort descriptor values (the sort dataset size is specified with SORTSIZE). This parameter is required only if the device type to be used is different from that specified by the ADARUN DEVICE parameter.

TEMPDEV : Temporary Storage Device Type

The TEMPDEV parameter indicates the device type to be used for the temp dataset that ADAVAL uses to store intermediate data. The dataset size is specified with the TEMPSIZE parameter. This parameter is required only if the device type to be used is different from that specified by the ADARUN DEVICE parameter.

Example of ADAVAL Output

ADAVAL output provides a SYSOUT (DD/DRUCK) table listing, by file and descriptor, of all data storage and Associator entries and their status. The following is an example of ADAVAL 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 ***
```

—where

- In the F (flag) column, “–” indicates that an inverted list entry is missing for the specified Data Storage descriptor; and “+” 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.

Note:
*The “*** NO INCONSISTENCIES ***” entry occurs for every successful descriptor validation.*

JCL/JCS Requirements and Examples

This section describes the job control information required to run ADAVAL with BS2000, OS/390 or z/OS, VM/ESA, and VSE/ESA systems and shows examples of each of the job streams.

Collation with User Exit

If a collation user exit is to be used during ADAVAL execution, the ADARUN CDXnn parameter must be specified for the utility run.

Used in conjunction with the universal encoding subsystem (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 *DBA Reference Manual* for more information.

Sorting Large Files

When sorting large files, performance can be improved if either the sort dataset occupies two volumes or two sort datasets are specified. Both datasets must be on the same device type (SORTDEV parameter), and each must be exactly half the size specified by the SORTSIZE parameter.

BS2000

Dataset	Link Name	Storage	More Information
Sort area	DDSORTR1	disk	
Sort area	DDSORTR2	disk	Split the sort area across two volumes when using large files (see note)
Temp area	DDTEMPR1	disk	
Associator area	DDASSOR1	disk	
ADARUN parameters	SYSDTA/DDCARD		*Operations Manual*
ADAVAL parameters	SYSDTA/DDKARTE		
ADARUN messages	SYSOUT/DDPRINT		*Messages and Codes*
ADAVAL messages	SYSLST/DDDRUCK		*Messages and Codes*
Rejected data	DDFEHL	tape/disk	

ADAVAL JCL Example (BS2000)

In SDF Format:

```
/.ADAVAL LOGON
/MODIFY-TEST-OPTIONS DUMP=YES
/REMARK *
/REMARK * A D A V A L ALL FUNCTIONS
/REMARK *
/DELETE-FILE VAL.FEHL
/SET-JOB-STEP
/CREATE-FILE VAL.FEHL,PUB(SPACE=(48,48))
/SET-JOB-STEP
/ASS-SYSLST L.VAL
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADAvrs.MOD
/SET-FILE-LINK DDASSOR1,ADAyyyyy.ASSO
/SET-FILE-LINK DDSORTR1,ADAyyyyy.SORT
/SET-FILE-LINK DDTEMPR1,ADAyyyyy.TEMP
/SET-FILE-LINK DDFEHL1,VAL.FEHL
/START-PROGRAM *M(ADA.MOD,ADARUN),PR-MO=ANY
ADARUN PROG=ADAVAL,DB=yyyyy,IDTNAME=ADABAS5B
ADAVAL VALIDATE FILE=30,SORTSIZE=3,TEMPSIZE=5
/LOGOFF SYS-OUTPUT=DEL
```

In ISP Format:

```
/.ADAVAL LOGON
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK * A D A V A L ALL FUNCTIONS
/REMARK *
/SYSFILE SYSLST=L.VAL
/FILE ADAyyyyy.TEMP ,LINK=DDTEMPR1
/FILE ADAyyyyy.SORT ,LINK=DDSORTR1
/FILE ADAyyyyy.ASSO ,LINK=DDASSOR1
/FILE ADA.MOD,LINK=DDLIB
/FILE VAL.FEHL,LINK=DDFEHL,SPACE=(48,48)
/EXEC (ADARUN,ADA.MOD)
ADARUN PROG=ADAVAL,DB=yyyyy,IDTNAME=ADABAS5B
ADAVAL VALIDATE FILE=30,SORTSIZE=3,TEMPSIZE=5
/LOGOFF NOSPOOL
```

OS/390 or z/OS

Dataset	DD Name	Storage	More Information
Sort area	DDSORTR1	disk	
Sort area	DDSORTR2	disk	Split the sort area across two volumes when using large files (see note)
Temp area	DDTEMPR1	disk	
Associator area	DDASSOR1	disk	
ADARUN parameters	DDCARD	reader	*Operations Manual*
ADAVAL parameters	DDKARTE	reader	
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADAVAL messages	DDDRUCK	printer	*Messages and Codes*
Rejected data	DDFEHL	tape/disk	

ADAVAL JCL Example (OS/390 or z/OS)

```
//ADAVAL    JOB
//*
//*   ADAVAL: VALIDATE DATA BASE
//*
//VAL      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
//DDWORKR1  DD   DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <=== WORK
//DDSORTR1  DD   DISP=OLD,DSN=EXAMPLE.DByyyyy.SORTR1 <=== SORT
//DDTEMPR1  DD   DISP=OLD,DSN=EXAMPLE.DByyyyy.TEMPR1 <=== TEMP
//DDDRUCK   DD   SYSOUT=X
//DDPRINT   DD   SYSOUT=X
//SYSUDUMP  DD   SYSOUT=X
//DDCARD    DD   *
ADARUN PROG=ADAVAL,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
```

```
//DDKARTE   DD   *
ADAVAL VALIDATE FILE=1,TEMPSIZE=ttt,SORTSIZE=sss
/*
//
```

Refer to ADAVAL in the MVSJOBS dataset for this example.

VM/ESA or z/VM

Dataset	DD Name	Storage	More Information
Sort area	DDSORTR1	disk	
Sort area	DDSORTR2	disk	Split the sort area across two volumes when using large files (see note)
Temp area	DDTEMPR1	disk	
Associator area	DDASSOR1	disk	
ADARUN parameters	DDCARD	disk/terminal/reader	*Operations Manual*
ADAVAL parameters	DDKARTE	disk/terminal/reader	
ADARUN messages	DDPRINT	disk/terminal/printer	*Messages and Codes*
ADAVAL messages	DDDRUCK	disk/terminal/printer	
Rejected data	DDFEHL	tape/disk	

Example (VM/ESA or z/VM)

```
DATADEF DDPRINT,DSN=ADAVAL,DDPRINT,MODE=A
DATADEF DUMP,DUMMY
DATADEF DDDRUCK,DSN=ADAVAL.DDDRUCK,MODE=A
DATADEF DDTEMPR1,DSN=ADABASVv.TEMP,VOL=TEMPV1
DATADEF DDSORTR1,DSN=ADABASVv.SORT,VOL=SORTV1
DATADEF DDASSOR1,DSN=ADABASVv.ASSO,VOL=ASSOV1
DATADEF DDCARD,DSN=RUNVAL.CONTROL,MODE=A
DATADEF DDKARTE,DSN=ADAVAL.CONTROL,MODE=A
DATADEF DDFEHL,DSN=FILE030.VALERR,MODE=A
ADARUN
```

Contents of RUNVAL CONTROL A1:

```
ADARUN PROG=ADAVAL,DEVICE=dddd,DB=yyyyy
```

Contents of ADAVAL CONTROL A1

```
ADAVAL VALIDATE FILE=30,SORTSIZE=3,TEMPSIZE=5
```

VSE

File	File Name	Storage	Logical Unit	More Information
Sort area	SORTR1	disk		
Sort area	SORTR2	disk		When using large files, split the sort area across two volumes (see note)
Temp area	TEMPR1	disk*		
Associator area	ASSOR1	disk		
ADARUN parameters	— CARD CARD	reader tape disk	SYSRDR SYS000 *	
ADAVAL parameters	–	reader	SYSIPT	*Utilities Manual*
ADARUN messages	–	printer	SYSLST	*Messages and Codes*
ADAVAL messages	–	printer	SYS009	*Messages and Codes*
Rejected data	FEHL	tape disk	SYS010 *	

* Any programmer logical unit can be used.

ADAVAL JCS Example (VSE/ESA)

See appendix B for descriptions of the VSE/ESA procedures (PROCs).

```
* $$ JOB JNM=ADAVAL,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADAVAL
*       VALIDATE DATABASE
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADAVAL,MODE=SINGLE,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
ADAVAL VALIDATE FILE=1,TEMPSIZE=ttt,SORTSIZE=sss
/*
/&
* $$ EOJ
```

Refer to member ADAVAL.X for this example

23

ADAZAP : DISPLAY OR MODIFY ASSO, DATA, AND WORK DATASETS

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 datasets.

Because a significant element of risk is involved in modifying physical database blocks, the use of this utility is restricted. Software AG will provide the mastercode 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, Software AG strongly recommends that you use your external security system to protect ADAZAP just as you protect other ZAP programs.

Software AG 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.

A version of ADAZAP running with different syntax was unofficially distributed with previous releases of Adabas. No documentation was or is provided for this earlier version and it was and is not supported.

Syntax

```
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 ]
```

Essential Parameters

MCODE

For security purposes, a mastercode is required to run the ADAZAP utility. Software AG provides the 8-byte mastercode on written request to authorized individuals.

ASSO | DATA | WORK

It is necessary to specify the physical dataset 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 dataset.

JCL/JCS Requirements and Examples

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.

BS2000

Dataset	Link Name	Storage	More Information
Associator	DDASSORn	disk	required if ASSO is being zapped
Data Storage	DDDATARn	disk	required if DATA is being zapped
Work	DDWORKRn	disk	required if WORK is being zapped
ADARUN parameters	DDCARD	reader	*Operations Manual*
ADAZAP parameters	DDKARTE	reader	
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADAZAP messages	DDDRUCK	printer	*Messages and Codes*

ADAZAP JCL Example (BS2000)

In SDF Format:

```
/BEGIN-PROC A
/REMA
/REMA SAMPLE JCL FOR ADAZAP
/REMA
/ASS-SYSOUT L.ADAZAP.OUT
/ASS-SYSLST L.ADAZAP.LST
/ASS-SYSDTA *SYSCMD
```

```
/SET-FILE-LINK DDLIB,ADABAS.Vvrs.MOD
/SET-FILE-LINK DDASSOR1,DByyyyyy.ASSOR1,SUP=DISK(SHARE-UPD=YES)
/SET-FILE-LINK DDDATAR1,DByyyyyy.DATAR1,SUP=DISK(SHARE-UPD=YES)
/SET-FILE-LINK DDWORKR1,DByyyyyy.WORKR1,SUP=DISK(SHARE-UPD=YES)
/STA-PROG *M(ADABAS.Vvrs.MOD,ADARUN),RUN-MODE=*ADV(ALT=Y)
ADARUN PROG=ADAZAP,DB=yyyyyy
ADAZAP MCODE=xxxxxxxxxx  <<--- 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
/SET-JOB-STEP
/ASS-SYSDTA *PRIM
/ASS-SYSLST *PRIM
/ASS-SYSOUT *PRIM
/END-PROC
```

in ISP Format:

```
/.ADAZAP PROC
/OPTION MSG=FH,DUMP=YES
/REMARK *
/REMARK * SAMPLE JCL FOR ADAZAP
/REMARK *
/SYSFILE SYSLST=L.ZAP
/SYSFILE SYSDTA=(SYSCMD)
/FILE ADAyyyyy.TEMP ,LINK=DDTEMPR1
/FILE ADAyyyyy.SORT ,LINK=DDSORTR1
/FILE ADAyyyyy.ASSO ,LINK=DDASSOR1
/EXEC (ADARUN,ADA.MOD)
ADAZAP MCODE=xxxxxxxxxx <<    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
/STEP
/SYSFILE SYSDTA=()
/SYSFILE SYSLST=()
/ENDP
```

OS/390 or z/OS

Dataset	Link Name	Storage	More Information
Associator	DDASSORn	disk	required if ASSO is being zapped
Data Storage	DDDATARn	disk	required if DATA is being zapped
Work	DDWORKRn	disk	required if WORK is being zapped
ADARUN parameters	DDCARD	reader	*Operations Manual*
ADAZAP parameters	DDKARTE	reader	
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADAZAP messages	DDDRUCK	printer	*Messages and Codes*

Example (OS/390 or z/OS)

```
//ADAZAP    JOB
//*
//ZAP       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
//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=mmmmmmmmm  <<--- MASTER PASSWORD
  ADAZAP ASSO RABN=1,OFFSET=X'10',LENGTH=16    <===DISPLAY ASSO RABN 1
  ADAZAP ASSO RABN=1,OFFSET=X'10',VER='C1C2',REP=X'C2C1'
  ADAZAP ASSO RABN=1,OFFSET=X'10',LENGTH=16    <===DISPLAY ASSO RABN 1
/*
//
```

Refer to ADAZAP in the MVSJOBS dataset for this example.

VM/ESA or z/VM

Dataset	Link Name	Storage	More Information
Associator	DDASSORn	disk	required if ASSO is being zapped
Data Storage	DDDATARn	disk	required if DATA is being zapped
Work	DDWORKRn	disk	required if WORK is being zapped
ADARUN parameters	DDCARD	reader	*Operations Manual*
ADAZAP parameters	DDKARTE	reader	
ADARUN messages	DDPRINT	printer	*Messages and Codes*
ADAZAP messages	DDDRUCK	printer	*Messages and Codes*

Example (VM/ESA or z/VM)

```
DATADEF DDASSOR1,DSN=ADABASVv.ASSO,VOL=ASSOV1     <=== ASSO
DATADEF DDDATAR1,DSN=ADABASVv.DATA,VOL=DATAV1     <=== DATA
DATADEF DDWORKR1,DSN=ADABASVv.WORK,VOL=WORKV1     <=== WORK
DATADEF DDPRINT,DSN=ADAZAP.DDPRINT,MODE=A
DATADEF DUMP,DUMMY

DATADEF DDDRUCK,DSN=ADAZAP.DDDRUCK,MODE=A
DATADEF DDCARD,DSN=RUNZAP.CONTROL,MODE=A
DATADEF DDKARTE,DSN=ADAZAP.CONTROL,MODE=A
ADARUN
```

Contents of RUNREP CONTROL A1

```
ADARUN PROG=ADAZAP,DEVICE=dddd:
```

Contents of ADAREP CONTROL A1:

```
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='C1C2',REP=X'C2C1'
ADAZAP ASSO RABN=1,OFFSET=X'10',LENGTH=16    <===DISPLAY ASSO RABN 1
```

VSE/ESA

Dataset	Link Name	Storage	More Information
Associator	ASSORn	disk	required if ASSO is being zapped
Data Storage	DATARn	disk	required if DATA is being zapped
Work	WORKRn	disk	required if WORK is being zapped
ADARUN parameters	CARD	reader	*Operations Manual*
ADAZAP parameters	KARTE	reader	
ADARUN messages	PRINT	printer	*Messages and Codes*
ADAZAP messages	DRUCK	printer	*Messages and Codes*

Example (VSE/ESA)

```
* $$ JOB JNM=ADAZAP,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADAZAP
*
// EXEC PROC=ADAVvLIB
// EXEC PROC=ADAVvFIL
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADAZAP,MODE=MULTI,SVC=xxx,DEVICE=dddd,DBID=yyyyy
/*
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='C1C2',REP=X'C2C1'
ADAZAP ASSO RABN=1,OFFSET=X'10',LENGTH=16    <===DISPLAY ASSO RABN 1
/*
/&
* $$ EOJ
```

Refer to member ADAZAP.X for this example.

See appendix B for descriptions of the VSE/ESA procedures (PROCs).

A

APPENDIX A : ADABAS SEQUENTIAL FILES

Sequential File Table

This appendix 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	VSE Tape SYS	Out	In	BLKSIZE by device	Concatenation
ADACDC	DD/SIIN	10		x		Yes
ADACMP	DD/AUSBA	12	x			Yes
	DD/EBAND	10		x		
	DD/FEHL	14	x			
ADALOD	DD/EBAND	10		x		Yes
	DD/FILEA	12	x	x	Yes	
	DD/ISN	16		x		Yes
	DD/OLD	14	x			
ADAMER	DD/EBAND	10		x		
ADAORD	DD/FILEA	10	x	x	Yes	
ADAPLP	DD/PLOG	14		x		Yes
ADAREP	DD/SAVE	10		x		Yes
	DD/PLOG	11		x		Yes
ADARES	DD/BACK	20		x		Yes
	DD/SIAUS1	21	x			
	DD/SIAUS2	22	x			
	DD/SIIN	20		x		Yes
ADASAV	DD/DEL1	31		x		Yes
	DD/DEL2	32		x		Yes
	DD/DEL3	33		x		Yes
	DD/DEL4	34		x		Yes

Utility	File Name	VSE Tape SYS	Out	In	BLKSIZE by device	Concatenation
	DD/DEL5	35		x		Yes
	DD/DEL6	36		x		Yes
	DD/DEL7	37		x		Yes
	DD/DEL8	38		x		Yes
	DD/DUAL1	21	x			
	DD/DUAL2	22	x			
	DD/DUAL3	23	x			
	DD/DUAL4	24	x			
	DD/DUAL5	25	x			
	DD/DUAL6	26	x			
	DD/DUAL7	27	x			
	DD/DUAL8	28	x			
	DD/FULL	30		x		Yes
	DD/PLOG	10		x		Yes
	DD/REST1	11		x		Yes
	DD/REST2	12		x		
	DD/REST3	13		x		
	DD/REST4	14		x		
	DD/REST5	15		x		
	DD/REST6	16		x		
	DD/REST7	17		x		
	DD/REST8	18		x		
	DD/SAVE1	11	x			
	DD/SAVE2	12	x			
	DD/SAVE3	13	x			
	DD/SAVE4	14	x			

Utility	File Name	VSE Tape SYS	Out	In	BLKSIZE by device	Concatenation
	DD/SAVE5	15	x			
	DD/SAVE6	16	x			
	DD/SAVE7	17	x			
	DD/SAVE8	18	x			
ADASEL	DD/EXPA1	11	x			
	DD/EXPA2	12	x			
	DD/EXPA3	13	x			
	DD/EXPA4	14	x			
	DD/EXPA5	15	x			
	DD/EXPA6	16	x			
	DD/EXPA7	17	x			
	DD/EXPA8	18	x			
	DD/EXPA9	19	x			
	DD/EXPA10	20	x			
	DD/EXPA11	21	x			
	DD/EXPA12	22	x			
	DD/EXPA13	23	x			
	DD/EXPA14	24	x			
	DD/EXPA15	25	x			
	DD/EXPA16	26	x			
	DD/EXPA17	27	x			
	DD/EXPA18	28	x			
	DD/EXPA19	29	x			
	DD/EXPA20	30	x			
	DD/SIIN	10		x		Yes

Utility	File Name	VSE Tape SYS	Out	In	BLKSIZE by device	Concatenation
ADAULD	DD/OUT1	10	x		Yes	
	DD/OUT2	11	x		Yes	
	DD/ISN	12	x		Yes	
	DD/SAVE	13		x		Yes
	DD/PLOG	14		x		Yes
	DD/FULL	30		x		Yes
	DD/DEL1–8	31–38		x		Yes
ADAVAL	DD/FEHL	10	x		Yes	

Files that are both output and input are first written and then read by the indicated program. BS2000, VM/ESA or z/VM, OS/390 or z/OS, and OS-compatible files have “DD...” names (DDSIIN, DDFEHL, etc.); VSE/ESA file names are without “DD”.

Operating System Dependencies

The following sections describe characteristics of file and device definition by operating system.

BS2000 Systems

Note:
This discussion uses SPF format. In ISP format:

SPF Format	***ISP Format***
BUFF–LEN	*BLKSIZE defined by BLKSIZE=(STD,16)*
REC–FORM	*RECFM*
REC–SIZE	*RECSIZE*
SET–FILE–LINK	*FILE*

The LINK name by which a file is referenced is determined as follows:

- The characters DD are prefixed to the file name to form the LINK name.
- If files for which the column “Concatenation” contains “Yes” are on tape, they may be concatenated as follows: the first file is read using the indicated LINK name; at the first end-of-file, 01 is appended to the LINK name; and, if there is a /SET–FILE–LINK (in ISP format /FILE) statement for that LINK name, reading continues.
- Each subsequent end-of-file adds 1 to the LINK name, and as long as there is a /SET–FILE–LINK (in ISP format /FILE) statement for that LINK name, reading continues through a maximum of 99. For LINK names longer than six characters, the excess characters will be overlaid with the file number increment (e.g., DDEBAND becomes DDEBAN01).
- BS2000 does not support the backward reading of multivolume tape files; therefore, all volumes of the ADARES DDBACK file must be specified in the reverse order in which they were written on /SET–FILE–LINK (in ISP format /FILE) statements using the LINK names DDBACK, DDBACK01, DDBACK02, and so on.

The BUFF–LEN of a sequential file is determined as follows:

1. The BUFF–LEN is obtained from the /SET–FILE–LINK statement or the dataset's catalog entry, if present.
2. If the BUFF–LEN cannot be obtained from the /SET–FILE–LINK statement and/or catalog, the value of the ADARUN QBLKSIZE parameter is used, if specified.
3. Otherwise, the BUFF–LEN depends on the device type as follows:

Tape:	32760
Disk:	32768 (BUFF–LEN=(STD,16))

The REC–SIZE and REC–FORM should be as follows:

Tape:	REC–SIZE = BUFF–LEN – 4; REC–FORM = V;
Disk:	REC–SIZE = BUFF–LEN – 20; RECFORM = V;
Input:	Obtained from the /SET–FILE–LINK statement or the dataset's catalog entry.

Note:
Do not specify REC–FORM, REC–SIZE, or BUFF–LEN for input datasets unless the TAPE dataset contains no REC–FORM, REC–SIZE, or BUFF–LEN values in HDR2.

The SPACE parameter for primary and secondary allocations must specify a multiple of three (3) times the number of PAM blocks specified in the BUFF–LEN parameter. Otherwise, I/O errors will occur. For the default /CREATE–FILE ...,PUB(SPACE(48,48)) and /SET–FILE–LINK ...,BUFF–LEN=STD(16) (in ISP format, BLKSIZE=(STD,16), SPACE=(48,48)) is the smallest valid value.

The portions of the DDDRUCK and DDPRINT datasets already written to disk can be accessed during either a regular nucleus or utility session for reading. This includes the following BS2000 read accesses:

- SHOW–FILE
- @READ dataset
- /COPY–FILE (in ISP format, /COPY)

Concatenation of Sequential Input Files for BS2000

For using more than one dataset as input medium to an ADABAS utility, some operating systems (such as OS/390) provide a concatenation feature.

For BS2000 this feature is simulated by adding /SET–FILE–LINK (in ISP format, /FILE) statements with modified LINK names created from the original and a two-digit increment (ranging from 01 to 99):

```
/SET-FILE-LINK DDTEST,firstfile
/SET-FILE-LINK DDTEST01,secondfile
/SET-FILE-LINK DDTEST02,thirdfile
...
/SET-FILE-LINK DDTEST99,lastfile
```

In ISP format:

```
/FILE firstfile ,LINK=DDTEST
/FILE secondfile,LINK=DDTEST01
/FILE thirdfile ,LINK=DDTEST02
...
/FILE lastfile ,LINK=DDTEST99
```

For those original LINK names that are 7 or 8 characters long, the incremental number occupies the 7th and 8th position. For example:

```
/SET-FILE-LINK DDEBAND,firstfile
/SET-FILE-LINK DDEBAND01,lastfile
```

In ISP format:

```
/FILE firstfile ,LINK=DDEBAND
/FILE secondfile,LINK=DDEBAN01
```

When processing input files that have the concatenation option at end-of-file of one input file, a check is made to determine whether a /SET–FILE–LINK (in ISP format, /FILE) statement exists for the next dataset. If none exists, the sequential GET call returns EOF; otherwise, the dataset currently open is closed, and an open is tried for the next file.

Files concatenated in this way must have the same file characteristics (block size, record format and record size).

This concatenation feature applies also to files that are processed backwards. The order of the LINK names is the reverse of the creation order. For example, ADARES with DDBACK:

```
/SET-FILE-LINK DDBACK,lastfile
/SET-FILE-LINK DDBACK01,filebeforelast
/...
/SET-FILE-LINK DDBACKnn,firstfile
```

In ISP format:

```
/FILE lastfile ,LINK=DDBACK
/FILE filebeforelast,LINK=DDBACK01
/....
/FILE firstfile ,LINK=DDBACKnn
```

Note that this feature can also be used to process a multivolume file backwards, if each volume is specified with a separate /SET–FILE–LINK (in ISP format, /FILE) statement.

The following list is of LINK names/utilities with the concatenation option:

DDDELn (where n = 1–8)	ADASAV
DDEBAND	ADACMP ADALOD ADAMER
DDFULL	ADASAV
DDISN	ADALOD
DDPLOG	ADAPLP ADASAV
DDBACK	ADARES
DDSIIN	ADARES ADASEL
DDREST1	ADASAV (LINK names used are DDREST1, DDREST01, DDREST02, and so on.)

Example for Use of the Concatenation Feature with ADARES

During the last nucleus session, three protection log files were produced with ADARES PLCOPY named F1, F2, F3.

When backing out the session to a specific point, use the following /SET–FILE–LINK (in ISP format, /FILE) statements for the ADARES BACKOUT function:

```
/SET-FILE-LINK DDBACK,F3
/SET-FILE-LINK DDBACK01,F2
/SET-FILE-LINK DDBACK02,F1
```

In ISP format:

```
/FILE F3,LINK=DDBACK
/FILE F2,LINK=DDBACK01
/FILE F1,LINK=DDBACK02
```

To regenerate the database from the protection log that was produced during the session, use the following /SET–FILE–LINK (in ISP format, /FILE) statements for the ADARES REGENERATE function:

```
/SET-FILE-LINK DDSIIN,F1
/SET-FILE-LINK DDSIIN01,F2
/SET-FILE-LINK DDSIIN02,F3
```

In ISP format:

```
/FILE F1,LINK=DDSIIN
/FILE F2,LINK=DDSIIN01
/FILE F3,LINK=DDSIIN02
```

Control Statement Read Procedure in Version 11.2 (OSD 2.0)

With BS2000 version 11.2 (OSD 2.0), the SYSIPT system file is no longer available. Beginning with version 5.3.3, ADABAS can read all control statements from the SYSDTA system file.

When running on BS2000 Versions 10.0 or 11.0, the SYSIPT assignment can still be used; however, Software AG recommends adapting all ADABAS utility and Entire Net-Work job control to indicate the SYSDTA system file before migrating to BS2000 version 11.2 (OSD 2.0).

ADARUN TAPEREL : Tape Release Option

The ADARUN parameter TAPEREL is required to perform the tape handling control for utilities that access files on tape. See the *ADABAS Operations Manual* for more information.

OS/390 or MVS/ESA Systems

The DDNAME is formed by prefixing the characters DD to the file name.

To allow utilities to access dataset information after closing, the DD statement for sequential datasets 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:	32760
3330 disk:	13030
3340 disk:	8368
3350 disk:	19069
3375 disk:	17600
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 dataset 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 dataset 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 dataset label.

If the DCB BUFNO parameter is not provided on the DD statement, the operating system default will be used.

VM/ESA Systems

The DATADEF name is formed by prefixing the characters DD to the file name.

The BLKSIZE of a sequential file is determined as follows:

- If the column, “BLKSIZE by device” specifies Yes for a file, the BLKSIZE depends on the device type as follows:

Tape:	32760
FBA disk:	32760
3330 disk:	13030
3340 disk:	8368
3350 disk:	19069
3375 disk:	17600
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 dataset 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 dataset label, the value of the ADARUN QBLKSIZE parameter is used, if specified.

For all sequential files except ADACMP EBAND, the RECFM is VB and LRECL is (BLKSIZE – 4). For ADACMP EBAND, RECFM and LRECL must be available from the DATADEF statement and/or dataset label.

VSE/ESA Systems

The following items determine how a file is referenced by the utilities running under VSE/ESA:

- The file name is used as the filename on the DLBL or TLBL statement.
- If files for which the column “Concatenation” contains Yes are on tape, they may be concatenated as follows:
  - The file is first read using the indicated file name.
  - At the first end-of-file, 01 is appended to the file name and, if there is a TLBL statement for that filename, reading continues.
  - At each subsequent end-of-file, 1 is added to the file name and reading continues as long as there is a TLBL statement for that filename, up through a maximum of 99.
- Since VSE does not support reading multivolume tape files backward, each volume of the ADARES BACK file must be specified in reverse order from the way it was written on TLBL statements using the filenames BACK, BACK01, BACK02, and so on.

Any programmer logical unit may be used for sequential files on disk. The **VSE Tape SYS** number must be used for sequential files on tape; any or all of these numbers may be changed using procedures defined in the *ADABAS Installation Manual*.

The BLKSIZE of a sequential file is determined as follows:

- If the column “BLKSIZE by Device” specifies Yes for a file, the BLKSIZE depends on the device type as follows:

Tape:	32760
FBA disk:	32760
3330 disk:	13030
3340 disk:	8368
3350 disk:	19069
3375 disk:	17600
3380 disk:	23476
3390 disk:	27998

- If the column “BLKSIZE by Device” does not specify Yes for a file, the value of the ADARUN QBLKSIZE parameter is used, if specified.

For ADACMP EBAND, this BLKSIZE is checked and may then be changed to an actual BLKSIZE, depending on the RECFM and LRECL parameters as specified on ADACMP control cards, as follows:

If RECFM= ...	then the actual BLKSIZE= ...
F	LRECL.
FB	BLKSIZE/LRECL*LRECL, where the remainder of the division is discarded before the multiplication.
U	LRECL, which must not be greater than BLKSIZE.
V	LRECL+4, which must not be greater than BLKSIZE.
VB	BLKSIZE, which must not be less than LRECL+4.

The RECFORM of all sequential files except ADACMP EBAND is VARBLK. For ADACMP EBAND, it is provided by the RECFM parameter of a control statement.

To distinguish whether VSE message 4140D refers to the first or a subsequent volume of a multivolume tape file, message ADAI31 is written to the operator whenever a tape file is opened, but not at end-of-volume.

Concatenation of Sequential Input Files for VSE/ESA

In those cases where it is desired to use more than one dataset as input medium for an ADABAS utility, a concatenation feature is provided by some operating systems (OS/390 or z/OS, for example).

For VSE, this feature is simulated by adding FILE statements with modified LINK names created from the original and a two-digit increment (ranging from 01 to 99):

```
// DLBL TEST  ,'firstfile'
// EXTENT ...
// DLBL TEST01,'secondfile'
// EXTENT ...
...
// DLBL TEST99,'lastfile'
// EXTENT ...
```

When processing input files that have the concatenation option at end-of-file (EOF) of one input file, a check is made to determine whether a FILE statement exists for the next dataset. If it does not exist the Sequential Get call returns EOF; otherwise, the dataset currently open is closed and an open is tried for the next file.

Files concatenated in this way must have the same file characteristics (block size, record format, and record size).

This concatenation feature applies also to files that are processed backwards. The order of the LINK names is the reverse of the creation order; for example, ADARES with BACK:

```
// DLBL BACK  ,'lastfile'
// EXTENT ...
// DLBL BACK01,'filebeforelast'
// EXTENT ...
...
// DLBL BACKnn,'firstfile'
// EXTENT ...
```

Note that this feature could also be used to process a multivolume file backwards, if each volume is specified with a separate FILE statement.

The following are the LINK names/utilities with the concatenation option:

DELn (where n=1–8)	ADASAV
EBAND	ADACMP ADALOD ADAMER
FULL	ADASAV
ISN	ADALOD
PLOG	ADAPLP ADASAV
BACK	ADARES
SIIN	ADARES ADASEL
REST1	ADASAV (LINK names used are REST1, REST101, REST102, and so on.)

Example for Use of the Concatenation Feature with ADARES

During the last nucleus session, three protection log files were produced with ADARES PLCOPY named F1, F2, F3.

When deciding to back out the session to a specific point, the following FILE statements should be used for the ADARES BACKOUT function:

```
// DLBL BACK  ,'F3'
// EXTENT ...
// DLBL BACK01,'F2'
// EXTENT ...
// DLBL BACK02,'F1'
// EXTENT ...
```

To regenerate the database from the protection log that was produced during the session, the following FILE statements should be used for the ADARES REGENERATE function:

```
// DLBL SIIN  ,'F1'
// EXTENT ...
// DLBL SIIN01,'F2'
// EXTENT ...
// DLBL SIIN02,'F3'
// EXTENT ...
```

B

APPENDIX B : PROCEDURES FOR VSE/ESA EXAMPLES

The VSE/ESA examples assume that the procedures for defining Adabas libraries (ADAVvLIB) and Adabas files (ADAVvFIL) have been cataloged into an accessible procedure library.

For information about cataloging these procedures, refer to the section **Catalog Procedures for Defining Libraries and the Database** in the VSE/ESA chapter of the *Adabas Installation Manual*.

Information about cataloging procedures for use with the Delta Save Facility are documented in the *Adabas Delta Save Facility Manual*.

Adabas Libraries (ADAVvLIB)

```
// PROC
* ******************************************************** *
* LIBRARY DEFINITIONS AND CHAINING FOR ADABAS               *
* ******************************************************** *
// SETPARM VERS=vrs                  <- CURRENT VERSION
// SETPARM ADALIB=SAGLIB             <- SAG PRODUCT LIBRARY
// SETPARM ADASUB=ADA&VERS           <- ADABAS SUBLIBRARY
// DLBL SAGLIB,'SAG.PRODUCT.LIBRARY'
// EXTENT ,vvvvvv
// LIBDEF *,SEARCH=&ADALIB..&ADASUB,TEMP
// LIBDEF PHASE,CATALOG=&ADALIB..&ADASUB,TEMP
// ASSGN SYS009,PRINTER
```

—where

vrs	is the Adabas version, revision, and system maintenance (SM) level
vvvvvv	is the programmer logical unit assigned

Adabas Files (ADAVvFIL)

```
// ASSGN SYS031,dddd,VOL=ADA001,SHR
// ASSGN SYS032,dddd,VOL=ADA002,SHR
// ASSGN SYS033,dddd,VOL=ADA003,SHR
// ASSGN SYS034,dddd,VOL=ADA004,SHR
// DLBL ASSOR1,'EXAMPLE.ADAyyyyy.ASSOR1',99/365,DA
// EXTENT SYS031,ADA001,,,15,1500
// DLBL DATAR1,'EXAMPLE.ADAyyyyy.DATAR1',99/365,DA
// EXTENT SYS032,ADA002,,,15,3000
// DLBL WORKR1,'EXAMPLE.ADAyyyyy.WORKR1',99/365,DA
// EXTENT SYS033,ADA003,,,15,600
// DLBL PLOGR1,'EXAMPLE.ADAyyyyy.PLOGR1',99/365,DA
// EXTENT SYS034,ADA004,,,15,600
// DLBL PLOGR2,'EXAMPLE.ADAyyyyy.PLOGR2',99/365,DA
// EXTENT SYS034,ADA004,,,615,600
// DLBL CLOGR1,'EXAMPLE.ADAyyyyy.CLOGR1',99/365,DA
// EXTENT SYS034,ADA004,,,1215,750
// DLBL CLOGR2,'EXAMPLE.ADAyyyyy.CLOGR2',99/365,DA
// EXTENT SYS034,ADA004,,,1965,750
// DLBL TEMPR1,'EXAMPLE.ADAyyyyy.TEMPR1',99/365,DA
// EXTENT SYS032,ADA002,,,3015,1500
// DLBL SORTR1,'EXAMPLE.ADAyyyyy.SORTR1',99/365,DA
// EXTENT SYS033,ADA003,,,615,375
// EXTENT SYS034,ADA004,,,2715,375
// DLBL RLOGR1,'EXAMPLE.ADAyyyyy.RLOGR1',99/365,DA
// EXTENT SYS033,ADA003,,,990,150
```

C

APPENDIX C : SUPPLIED UES ENCODINGS

The tables in this appendix list the encodings available with universal encoding support (UES) when executing Adabas utilities or issuing Adabas commands. Encodings represent single-byte character sets (Latin–1 or not) or double-/multiple-character sets. If encodings

- have a character set in common with other encodings so that conversion between them is accomplished without loss of data, they are “interoperable”.
- are not interoperable, they are “coexistent” with other encodings.

Columns used in the following tables are described as follows:

Key	Entire Conversion Services (ECS) key number in decimal and hexadecimal	
CS	character set as identified by IBM	
CP	code page usually identical to the “key”. It is **not** identical when • CP does not fit into the ECS key number range 1–4095; or • the encoding is constructed from two or more code pages.	
Size	F	encoding uses all allocated graphical character space
	M	maximum for given ESID
	S	subset
ESID	encoding scheme identifier with the following entries:	
	1100	EBCDIC fixed 1
	1301	EBCDIC mixed DBCS modal
	2100	IBM PC data fixed 1
	2200	IBM PC data DBCS only
	2300	IBM PC data mixed DBCS nonmodal
	4100	ISO–8 fixed 1
	4105	ISO–8 fixed 1 graphic characters in the range x’80’– x‘9F’ reserved for control codes
Fill	the hexadecimal value representing the fill character used in the encoding.	
Sub	the hexadecimal value representing the substitution character used in the encoding.	

Interoperable Encodings

Single-Byte Character Sets (Latin–1)

Key		CS	CP	F/M/S	ESID	Fill	Sub	Description
Dec	Hex			Size				
37	025	697	37	F 190	1100	40	3F	CECP: USA, Canada (ESA*), Netherlands, Portugal, Brazil, Australia, New Zealand
273	111	697	273	F 190	1100	40	3F	CECP: Austria, Germany, de_deu
277	115	697	277	F 190	1100	40	3F	CECP: Denmark, Norway
278	116	697	278	F 190	1100	40	3F	CECP: Finland, Sweden
280	118	697	280	F 190	1100	40	3F	CECP: Italy, it_ita
284	11C	697	284	F 190	1100	40	3F	CECP: Spain, Latin America (Spanish), es_esp
285	11D	697	285	F 190	1100	40	3F	CECP: United Kingdom, en_gbr
297	129	697	297	F 190	1100	40	3F	CECP: France, fr_fra
500	1F4	697	500	F 190	1100	40	3F	CECP: Belgium, Canada (AS/400*), Switzerland, international Latin–1
819	333	697	819	F 190	4100	20	1A	ISO 8859–1: Latin alphabet, Latin–1
871	367	697	871	F 190	1100	40	3F	CECP: Iceland, is_ISL
923	39B	**	923	F 190	4100	20	1A	ISO 8859–15 with euro sign
924	39C	**	924	F 190	1100	40	3F	IBM EBCDIC with same char. set as 923.
1140	474	**	1140	F 190	1100	40	3F	IBM EBCDIC 37 with euro sign
1141	475	**	1141	F 190	1100	40	3F	IBM EBCDIC 273 with euro sign
1142	476	**	1142	F 190	1100	40	3F	IBM EBCDIC 277 with euro sign
1143	477	**	1143	F 190	1100	40	3F	IBM EBCDIC 278 with euro sign
1144	478	**	1144	F 190	1100	40	3F	IBM EBCDIC 280 with euro sign
1145	479	**	1145	F 190	1100	40	3F	IBM EBCDIC 284 with euro sign
1146	47A	**	1146	F 190	1100	40	3F	IBM EBCDIC 285 with euro sign
1147	47B	**	1147	F 190	1100	40	3F	IBM EBCDIC 297 with euro sign
1148	47C	**	1148	F 190	1100	40	3F	IBM EBCDIC 285 with euro sign

Key		CS	CP	F/M/S	ESID	Fill	Sub	Description
Dec	Hex			Size				
1149	47D	**	1149	F 190	1100	40	3F	IBM EBCDIC 871 with euro sign
*3946	F6A	697	850	S 190	2100	20	7F	PC data–190: Latin alphabet, Latin–1 (CCSID=4946)

***** This code page is not yet available but may be at a later time.

******This information is not yet available.

Single-Byte Character Sets (Non-Latin–1)

Key		CS	CP	F/M/S	ESID	Fill	Sub	Description
Dec	Hex			Size				
420	1A4	235	420	M 181	1100	40	3F	IBM, Arabic (all presentation shapes)
424	1A8	941	424	M 152	1100	40	3F	IBM, Hebrew, iw_isr
813	32D	925	813	M 183	4100	20	1A	ISO 8859–7: Greek/Latin, el_GRC
870	366	959	870	F 190	1100	40	3F	IBM, Latin–2
875	36B	925	875	M 184	1100	40	3F	IBM, Greek, el_GRC
912	390	959	912	F 190	4100	20	1A	ISO 8859–2: Latin–2
914	392		914	F 190	4100	20	1A	ISO 8859–4
915	393	1150	915	F 190	4100	20	1A	ISO 8859–5: Cyrillic, 8-bit
916	394	941	916	M 152	4100	20	1A	ISO 8859–8: Hebrew, iw_ISR
918	396	1160	918	F 190	1100	40	3F	IBM, Urdu
919	397		919	F190	4100	20	1A	ISO 8859–10
920	398	1152	920	F 190	4100	20	1A	ISO 8859–9 Latin–5 (ECMA–128, Turkey TS–5881), tr_TUR
921	399	1305	921	F 190	4100	20	1A	IBM, Baltic, 8-bit
922	39A	1307	922	F 190	4100	20	1A	IBM, Estonia, 8-bit
1006	3EE	1160	1006	F 190	4100	20	1A	IBM, Urdu
1025	401	1150	1025	F 190	1100	40	3F	IBM, Cyrillic multilingual
1026	402	1152	1026	F 190	1100	40	3F	IBM, Turkey Latin–5, tr_TUR
1112	458	1307	1112	F 190	1100	40	3F	IBM, Baltic, multilingual (EBCDIC)
1122	462	1307	1122	F 190	1100	40	3F	IBM, Estonia (EBCDIC)

Double- and Multiple-Byte Character Sets

Key		CS	CP	F/M/S	ESID	Fill	Sub	Description
Dec	Hex			Size				
290	122	1172	290	M 162	1100	40	3F	Japanese Katakana host extended SBCS, ja_JPN
836	344	1174	836	M 99	1100	40	3F	Simplified Chinese host extended SBCS (EBCDIC), zh_CHN
1027	403	1172	1027	M 162	1100	40	3F	IBM, Japanese Latin host extended SBCS, ja_JPN
1041	411	1172	1041	S 162	2100	20	1A	IBM, Japanese PC data extended SBCS, ja_JPN
1043	413	1175	1043	M 97	2100	20	1A	IBM, traditional Chinese PC data extended SBCS, zh_TWN
1115	45B	1174	1115	M 99	2100	20	1A	IBM, simplified Chinese PC data single-byte (IBM GB) including 5 SAA SB characters, zh_CHN
1380	564	937	1380	M 9355	2200	A1A1	FEFE	IBM, simplified Chinese DBCS PC (IBM GB) including 1880 UDC and 31 IBM-selected, zh_CHN
1381	565	1174 937	1115 1380	M 9454	2300	20 A1A1	1A FEFE	IBM, simplified Chinese PC data mixed (IBM GB) including 1880 UDC, 31 IBM-selected and 5 SAA SB characters, zh_CHN
3026	BD2	1172 370	290 300	S 9306	1301	40 4040	3F FEFE	IBM CCSID 5026, Japanese Katakana-Kanji host mixed including 1880 UDC, extended SBCS, ja_JPN
3035	BDB	1172 370	1027 300	S 9306	1301	40 4040	3F FEFE	IBM CCSID 5035, Japanese Latin-Kanji host mixed including 1880 UDC, extended SBCS, ja_JPN
3396	D44	370	300	S 9144	1200	4040	FEFE	IBM, CCSID 4396, Japanese host double-byte including 1880 UDC, ja_JPN

Key		CS	CP	F/M/S	ESID	Fill	Sub	Description
Dec	Hex			Size				
3709	E7D	1175	3709	S 97	1100	40	3F	IBM CCSID 28709: traditional Chinese host extended SBCS (EBCDIC), zh_TWN
4091	FFB			F 63456		0020	001A	UTF–8: with user/compatibility area; without surrogates

Coexistent Encodings

Single-Byte Character Sets

Key		CS	CP	F/M/S	ESID	Fill	Sub	Description
Dec	Hex			Size				
3	3	103	367	F 94	5100	20	1A	US–ASCII
6	6			F 190	4100	20	1A	ISO 8859–3, Latin–3, Afrikaans, Catalan, Dutch, English, German, Italian, Maltese, Spanish, Turkish
9	9			M 145	4100	20	1A	ISO 8859–6, Arabic
437	1B5	697	437	F 222	2100	20	1A	Microsoft MSDOS US, en_USA
850	350	1106	950	F 222	2100	20	1A	Microsoft multilingual code page 850, Latin–1
858	350	1106	950	F 222	2100	20	1A	Microsoft multilingual code page 850, Latin–1, euro-ready
1250	4E2	1400	1250	M 217	4105	20	1A	Microsoft, Windows ANSI, Latin–2, euro-ready
1251	4E3	1401	1251	M 220	4105	20	1A	Microsoft, Windows ANSI, Cyrillic, euro-ready
1252	4E4	1402	1252	M 215	4105	20	1A	Microsoft, Windows ANSI, Latin–1, euro-ready
1253	4E5	1403	1253	M 206	4105	20	1A	Microsoft, Windows ANSI, Greek, euro-ready
1254	4E6	1404	1254	M 215	4105	20	1A	Microsoft, Windows ANSI, Turkey, euro-ready

Key		CS	CP	F/M/S	ESID	Fill	Sub	Description
Dec	Hex			Size				
1255	4E7	1405	1255	M 194	4105	20	1A	Microsoft, Windows ANSI, Hebrew, euro-ready
1256	4E8	1406	1256	M 214	4105	20	1A	Microsoft, Windows ANSI, Arabic, euro-ready
1257	4E9	1407	1257	M 203	4105	20	1A	Microsoft, Windows ANSI, Baltic rim, euro-ready
2084	824			M 222	3100	20	1A	RELCOM, koi8–r, RFC–1489, Russian internet character set
2087	827		775	M 222	3100	20	1A	Baltic
2258	8D2		1258	M 222	4105	20	1A	Microsoft, Windows ANSI, Vietnam
4092	FFC			3	–	20	A1	Software AG, used for ECS internally
4093	FFD				4100	20	1A	Software AG, old TS default, ASCII
4094	FFE	–	–		1100	40	3F	Software AG, old TS default, EBCDIC
4095	FFF			–	7200	0020	001A	Unicode

Double- and Multiple-Byte Character Sets

Key		CS	CP	F/M/S	ESID	Fill	Sub	Description
Dec	Hex			Size				
18	12	103 1061 1121 1062	367 952 896 953	S 13102	4403	20 8140	1A	IBM CCSID 1350, EUC–JP, ja_JPN composed of US–ASCII, JIS–X–0208, HW Katakana, and JIS–X–0212–90
36	24				5404	20	1A	ISO–2022–JP, US–ASCII, JIS–Roman, JIS–0208–1983, JIS_C_6226–1978
300	12C	1001	300	M 11634	1200	4040	FEFE	IBM Japanese Latin host double-byte including 4370 UDC, ja_JPN
301	12D	370	301	M 9144	2200	8140	FCFC	IBM Japanese PC double-byte including 1880 UDC, SJIS, ja_JPN

Key		CS	CP	F/M/S	ESID	Fill	Sub	Description
Dec	Hex			Size				
835	343	935	835	M 20263	1200	4040	FEFE	IBM traditional Chinese host double-byte including 6204 UDC (EBCDIC) zh_TWN
837	345	937	837	M 9355	1200	4040	FEFE	Simplified Chinese host double-byte including 1880 UDC (EBCDIC), zh_CHN
927	39F	935	927	M 20263	2200	8140	FCFC	IBM traditional Chinese PC data double-byte including 6204 UDC, zh_TWN
932	3A4	11223 70	897 301	M 9301	2300	20 8140	1A	Microsoft, JIS Roman, JIS–X–208, half-width Katakana, ja_JPN
935	3A7	1174 937	836 837	M 9454	1301	40 4040	3F FEFE	Simplified Chinese host mixed including 1880 UDC, extended SBCS (EBCDIC), zh_CHN
936	3A8	1185 937	1185 937	M 9449	2300	20 1A1A	1A	Microsoft, GB Roman, GB 2312–80, zh_CHN
937	3A9	1175 935	903 937	S 20360	1301	40 4040	3F FEFE	IBM traditional Chinese host mixed including 6204 UDC, extended SBCS (EBCDIC), zh_TWN
942	3AE	1172 370	1041 301	M 9306	2300	20 8140	1A FCFC	IBM, Japanese PC data mixed including 1880 UDC, extended SBCS, ja_JPN
948	3B4	1175 935	948	M 20360	2300	20 8140	1A FCFC	IBM, traditional Chinese PC data mixed including 6204 UDC, extended SBCS, zh_TWN
949	3B5	1278 1050	949	M 10197	2300	20 A1A1	1A AFFE	IBM, Korean IBM KS code – PC data mixed including 1880 UDC, ko_KOR
950	3B6	103 935	950	M 20357	2300	20 A140	1A	Microsoft, Big Five, zh_TWN

Key		CS	CP	F/M/S	ESID	Fill	Sub	Description
Dec	Hex			Size				
951	3B7	1050	951	M 10103	2200	A1A1	AFFE	IBM, Korean IBM KS code – PC data double-byte including 1880 UDC, ko_KOR
3001		10001	3001			20 8140	1A	MAC Japanese, JIS–Roman, JIS–X–208, HW Katakana, ja_JPN

INDEX

A

Adabas Delta Save Facility, database status report, change flags per file, 206
Adabas files, sequential, list by utility, 459
ADAORD utility, 11–88
  function directory, 13
  JCL requirements and examples
    BS2000, 76
    OS/390 or z/OS, 80
    VM/ESA or z/VM, 83
    VSE/ESA, 86
  REORASSO function, 13
  REORDATA function, 20
  REORDB function, 25
  REORFASSO function, 36
  REORFDATA function, 42
  REORFILE function, 47
  RESTRUCTUREDB function, 55
  RESTRUCTUREF function, 60
  STORE function, 65
ADAPLP utility, 89–104
  examples, 93
  IPLOGPRI function, 89
  JCL requirements and examples
    BS2000, 93
    OS/390 or z/OS, 97
    VM/ESA or z/VM, 100
    VSE/ESA, 102
  PLOGPRI function, 89
  SPLOGPRI function, 89
  WORKPRI function, 89
ADAPRI utility, 105–115
  ASSOPRI function, 105
  CLOGPRI function, 105
  DATAPRI function, 105
  examples, 106
  JCL requirements and examples
    BS2000, 108
    OS/390 or z/OS, 110
    VM/ESA or z/VM, 112
    VSE/ESA, 114
  PLOGPRI function, 105
  RLOGPRI function, 105
  SORTPRI function, 105
  TEMPPRI function, 105
  WORKPRI function, 105
ADAR2E utility, for recovering BS2000 pubset members, 173
ADARAI utility, 117–184
  CHKDB function, 120
  DISABLE function, 121
  function directory, 119
  function of, 117
  JCL requirements and examples
    BS2000, 158
    OS/390 or z/OS, 175
    VM/ESA or z/VM, 181
    VSE/ESA, 183
  LIST function, 122
    output examples
      BS2000, 126
      OS/390 or z/OS, 127
      VSE/ESA, 133
  PREPARE function, 135
  RECOVER function, 138
    executing, 143
    file-level recovery, 145
    input datasets, 142
    output job stream, 143
    restarting after interruption, 156
    skeleton job control input, 151
      for BS2000, 163, 167
      for OS/390 or z/OS, 177
  REMOVE function, 157
  special considerations for using

under BS2000, 158
under VM/ESA or z/VM, 181
under VSE/ESA, 183
ADAREP utility, 185–223
examples, 189
JCL requirements and examples
BS2000, 217
OS/390 or z/OS, 219
VM/ESA or z/VM, 221
VSE/ESA, 222
REPORT function, 185
ADARES utility, 225–274
BACKOUT DPLOG function, 238
BACKOUT function, 231
BACKOUT MPLOG function, 238
CLCOPY function, 246
COPY function, 248
function directory, 225
JCL requirements and examples
BS2000, 271
OS/390 or z/OS, 279
VM/ESA or z/VM, 288
VSE/ESA, 293
MERGE CLOG function, 252
PLCOPY function, 253
REGENERATE function, 256
REPAIR function, 266
ADASAV utility, 299–372
function directory, 300
JCL requirements and examples
BS2000, 364
OS/390 or z/OS, 371
VM/ESA or z/VM, 377
VSE/ESA, 380
RESTONL FILES function, 311
RESTONL FMOVE function, 318
RESTONL function, 301
overview, 299, 300
RESTONL GCB function, 305
RESTORE FILES function, 337
RESTORE FMOVE function, 343
RESTORE function, 329
overview, 299
RESTORE GCB function, 333
RESTPLOG function, 352
overview, 300
SAVE FILES function, 358
SAVE function, 355
overview, 300
ADASEL utility, 385–412
global parameters, 407
JCL requirements and examples
BS2000, 409
OS/390 or z/OS, 411
VM/ESA or z/VM, 412
VSE/ESA, 413
syntax, 385–406
additional selection criteria, 388
date/time, 389
IF statement, 391
output instruction, 398
DISPLAY, 399
NEWPAGE, 406
OUTPUT, 403
SKIP, 406
SELECT statement, 387
value criterion, 392
WITH clause, 390
ADAULD utility, 415–427
input processing, 425
JCL requirements and examples
BS2000, 429
OS/390 or z/OS, 432
VM/ESA or z/VM, 434
VSE/ESA, 436
LOAD FILE function, DDISN parameter, 417
output processing, 427
UNLOAD FILE function, 416–437
user exit 9 processing, 428
ADAVAL utility, 439–450
JCL requirements and examples
BS2000, 445
OS/390 or z/OS, 447
VM/ESA, 448

VSE/ESA, 449
output example, 443
rejected ISNs, 439
VALIDATE function, 440
ADAZAP utility, 451–462
JCL requirements and examples
BS2000, 454
OS/390 or z/OS, 456
VM/ESA or zVM, 457
VSE/ESA, 458
Associator
print blocks from, using utility, 105
reorder, using utility, 11
reorder for a file, using utility, 36
reorder for database, using utility, 13
validate values against Data Storage contents, using utility, 440

B

BS2000
ADAR2E utility, for recovering lost pubset members, 173
additional recovery log information for, 123
special considerations for using ADARAI, 158

C

Checkpoints
database status report, 212
written by Adabas nucleus/utilities, 214
Command log
copy a dual dataset to a sequential dataset, using utility, 246
merging multiple, 252, 284
multiple, print blocks from, using utility, 105
Coparameters, specifying, 8

D

Data Storage
print blocks from, using utility, 105
reorder, using utility, 11
reorder for a file, using utility, 42
reorder for database, using utility, 20
repair blocks, using utility, 266
validate against Associator values, using utility, 440
Database
build job stream needed to restore, using utility, 138
check active/inactive status, using utility, 120
load files into, using utility, 11
modify physical blocks, using utility, 451
produce status report, using utility, 185
recover, using utility, 138, 225
recovery, automate and optimize, using utility, 117
reorder Associator for, using utility, 13
reorder Data Storage and Associator for, using utility, 25
reorder Data Storage for, using utility, 20
restore from offline source, using utility, 329
restore from online source, using utility, 301
restore incremental from offline source, using utility, 333
restore incremental from online source, using utility, 305
restore updates performed between two checkpoints, using utility, 256
save, using utility, 355
save/restore, using utility, 299
unload to sequential dataset, using utility, 11, 55
validate, using utility, 439
Datasets, print blocks from, using utility, 105
DD/JCLIN, requirement for ADARAI, 142

F

Field definition table, database status report, 208
Files
  Adabas sequential
    BS2000
      concatenate input, 465
      control statement read procedure, 467
      record formats, 463
      tape release option, 467
    determining names and blocks sizes, 459
    list by utility, 459
    OS/390 or z/OS, record formats, 468
    VM/ESA or z/VM, record formats, 469
    VSE/ESA
      concatenation of input, 471
      record formats, 470
  build job stream needed to restore, using utility, 138
  load into existing database, using utility, 11, 65
  recovering with ADARAI RECOVER, 145
  reorder
    Associator and Data Storage for a specified, using utility, 47
    Associator for a specified, using utility, 36
    Data Storage for a specified, using utility, 42
  restore
    to any RABNs from offline source, using utility, 343
    to any RABNs from online source, using utility, 318
    to original RABNs from offline source, using utility, 337
    to original RABNs from online source, using utility, 311
    updates performed between two checkpoints, using utility, 256
  save specified, using utility, 358
  save/restore, using utility, 299
  unload
    to sequential dataset, using utility, 11, 60
    using utility, 415

G

Generation, definition of, 118

I

ISNs, file of unloaded, 417

M

Multiclient files, unload, using utility, 417

P

Parameters, positional values, specifying, 8
Procedures, for defining Adabas libraries, VSE/ESA, 475
Protection log
  dataset types, 230
  dual
    backout updates between two checkpoints, using utility, 238
    copy one to sequential dataset, using utility, 253
  information described, 230
  merging multiple, 281
  multiple
    backout updates between two checkpoints, using utility, 238
    print blocks from, using utility, 105
    print records from, using utility, 89
  restore, using utility, 352
  select information from, using utility, 385
  sequential
    backout updates between two checkpoints, using utility, 231

copy using utility, 248
print records from, using utility, 89
sequential intermediate, print records from, using utility, 89
use of compressed data, 230

R

Recovery
check before starting, 155
job stream
building, 144
restarting after interruption, 156
skeleton, 151
skeleton for BS2000, 163
of lost BS2000 pubset members, 173
restarting after interruption, 156
using ADARAI RECOVER, 138
Recovery Aid, function of, 117
Recovery log
additional information for BS2000, 123
checking content with ADARAI LIST, 122
deactivate, using utility, 121
disable, using utility, 157
display contents, using utility, 122
function of, 117
generation, definition of, 118
initialize and start, using utility, 135
prepare, using utility, 119
preparing with ADARAI PREPARE, 135
print blocks from, using utility, 105
Report, database status
checkpoint information, 212
Delta Save change flags per file, 206
description of, 194
Field definition table per file, 208
file options, 199
file space allocations, 200
general characteristics per file, 203
general file status, 198
general information, 194
options per file, 205
physical layout, 201
produce using utility, 185
space allocation, 196
space allocation per file, 207
special descriptors per file, 210

S

Save tape, copy using utility, 248
Sort, print blocks from, using utility, 105
Subparameters, specifying, 8

T

Temp, print blocks from, using utility, 105
TSOSCAT (BS2000), reading members with ADAR2E utility, 173

U

Universal encoding support (UES), table of encodings, 477–484
coexistent, 481
interoperable, 478
User exits, 9, used with ADAULD utility, 428
Utility control statement
parameter values, 6
default, 9
value, 7
value list, 8
value range, 8
rules, 6
syntax, 2
mutually exclusive parameters, 4
parameter list, 2
repeating parameters and values, 5
required and optional parameters, 3
subparameters, 3

V

VM/ESA or z/VM, special considerations for using ADARAI, 181
VSE/ESA
  procedures for examples, 475
  special considerations for using ADARAI, 183

W

Work
  print blocks from, using utility, 105
  print data protection records from, using utility, 89