Adabas Manager

Adabas Utilities

Version 9.5.0

October 2026

ADABAS & NATURAL

This document applies to Adabas Manager Version 9.5.0 and all subsequent releases.

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

Copyright © 2014-2026 Software GmbH, Darmstadt, Germany and/or its subsidiaries and/or its affiliates and/or their licensors.

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

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

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

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

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

Document ID: AMN-AADAOSUTILITIES-950-20261005

Table of Contents

Adabas Utilities vii
1 About this Documentation 1
Document Conventions 2
Online Information and Support 2
Data Protection 3
2 About the Utilities Documentation 5
3 Overview 7
4 ADABCK (Dump And Restore Database Or Files) 11
Functional Overview 12
Procedure Flow 15
Checkpoints 17
Control Parameters 18
Restart Considerations 40
5 ADACLP (Command Log Report) 41
Functional Overview 42
Procedure Flow 43
Checkpoints 43
Control Parameters 44
Specifying Multiple Selection Criteria 50
6 ADACMP (Compression Of Data) 51
Functional Overview 52
Procedure Flow 53
Checkpoints 54
Control Parameters 54
Output 66
Report 67
Restart Considerations 67
7 ADADBM (Database Modification) 69
Functional Overview 70
Procedure Flow 72
Checkpoints 74
Control Parameters 75
Restart Considerations 98
RBAC Protection 99
8 ADADCU (Decompression Of Data) 101
Functional Overview 102
Procedure Flow 103
Checkpoints 104
Control Parameters 104
Input and Output Data 112
Restart Considerations 113
9 ADAERR (Error File Report) 115
Functional Overview 116

Procedure Flow 117
Checkpoints 117
Control Parameter 117
Example 118
Rejected Data Records 118
10 ADAFDU (File Definition) 119
Functional Overview 120
Procedure Flow 121
Checkpoints 123
Control Parameters 123
Examples 134
11 ADAFIN (File Information Report) 135
Functional Overview 136
Procedure Flow 137
Checkpoints 138
Control Parameters 138
12 ADAFRM (Format And Create A New Database) 151
Functional Overview 152
Procedure Flow 154
Checkpoints 155
Control Parameters 155
Restart Considerations 159
Control Statement Examples 160
13 ADAINV (Creating, Removing And Verifying Inverted Lists) 161
Functional Overview 162
Procedure Flow 164
Checkpoints 165
Control Parameters 166
Restart Considerations 176
Examples 177
14 ADAKMS (Key Management Service) 183
Functional Overview 184
Procedure Flow 185
Checkpoints 185
Control Parameters 186
15 ADAMUP (Mass Add And Delete) 193
Functional Overview 194
Procedure Flow 195
Checkpoints 198
Control Parameters 199
Restart Considerations 205
SORT Data Set Placement 206
TEMP Data Set Placement 206
Examples 206
16 ADAOPR (Operator Utility) 209

Functional Overview 210
Procedure Flow 211
Checkpoints 211
Control Parameters 212
RBAC Protection 261
17 ADAORD (Reorder Database Or Files, Export/Import Files) 263
Functional Overview 264
Procedure Flow 265
Checkpoints 267
Control Parameters 267
Restart Considerations 276
Examples 276
18 ADAPLP (Protection Log Printout) 279
Functional Overview 280
Procedure Flow 281
Checkpoints 282
Control Parameters 282
ADAPLP Output 291
19 ADAPRI (Print Adabas Blocks) 295
Functional Overview 296
Procedure Flow 297
Checkpoints 298
Control Parameters 298
20 ADAREC (Recovery Of Database Or Files) 301
Functional Overview 302
Procedure Flow 304
Checkpoints 305
ADAREC Input Data 305
Control Parameters 305
REGENERATE Examples 313
BACKOUT Examples 319
ADAREC Restart Considerations 322
21 ADAREP (Database Report) 323
Functional Overview 324
Procedure Flow 325
Checkpoints 325
Control Parameters 326
22 ADASEL (PLOG Selection Tool) 343
Functional Overview 344
Procedure Flow 345
Checkpoints 346
Control Parameters 346
Query Syntax 350
Examples and Special Cases 364
23 ADAULD (File Unloading) 367

Functional Overview 368
Procedure Flow 370
Checkpoints 372
Control Parameters 373
Examples 378
TEMP Data Set Space Estimation 380
Restart Considerations 380
24 ADAVFY (Database Consistency Check) 381
Functional Overview 382
Procedure Flow 383
Checkpoints 384
Control Parameters 384
Examples 388

Adabas Utilities

This documentation describes the Adabas utilities.

The *Overview* provides a summary of the utilities available and their purpose.

1 About this Documentation

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

Document Conventions

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

Online Information and Support

Product Documentation

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

Product Training

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

Tech Community

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

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

Product Support

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

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

Data Protection

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

2 About the Utilities Documentation

The database administrator (DBA) uses the Adabas utilities to create and maintain Adabas databases. For each utility, the following information is provided:

- a description of the purpose of the utility;
- a functional overview of the utility;
- a description of the utility's control parameters;
- examples to illustrate the use of the utility, where appropriate.

3 Overview

This chapter gives an overview of the Adabas utilities, which provide all of the functions necessary to manage an Adabas database.

ADABCK

Backup and restore database or files

The Adabas backup utility dumps/restores the contents of the database (or a specific file or files) to/from a sequential data file. The utility can also be used to copy an Adabas backup copy.

ADACMP

Compression of data

The compression utility compresses user data. The compressed data is used as input for the mass update utility ADAMUP. The input for this utility is the raw data together with the data definitions that describe the structure of the data provided.

ADADBM

Database modification

The ADADBM utility consists of the following functions which can be used to make modifications to the database:

- The ADD_CONTAINER function adds a new container file to the Associator or Data Storage data set;
- The ADD_FIELDS function appends one or more new fields to the end of a file's FDT;
- The ALLOCATE functions increase the Normal Index, Upper Index, Address Converter or Data Storage space assigned to a file. The DEALLOCATE functions are the inverse;
- The CHANGE function changes the standard length of a field in the FDT;
- The CHANGE_FIELDS function changes a field definition;
- The DEFINE_REFINT function defines a new referential constraint;

- The DELCP function deletes old checkpoint records from the checkpoint file in the specified range of dates;
- The DELETE function deletes an Adabas file or a range of files from the database;
- The DELETE_DATABASE function deletes a database. Depending on the keyword specified, either just the containers are deleted, or the database directory and its content are deleted.
- The DISPLAY function displays the UCB;
- The DROP_FIELDS function marks the specified fields as not existing, which means that they can no longer be accessed;
- The DROP_LOBFILE function is the inverse function of ADAFDU ADD_LOBFILE;
- The DROP_REFINT function drops an existing referential constraint;
- The EXTEND_CONTAINER function extends the last container file defined for the database;
- The NEW_DBID function changes the identifier of the database in use;
- The NEWWORK function allocates and formats a new Adabas WORK data set;
- The PGM_REFRESH function is used to disable or enable refreshing an Adabas file inside an application program with an E1 command;
- The RECOVER function returns lost space to the FST;
- The REDUCE_CONTAINER function reduces the size of the last container file defined for the database;
- The REFRESH function resets a file or a range of files to the state of zero records loaded;
- The REMOVE_CONTAINER function deletes a container file;
- The REMOVE_DROP function, used in conjunction with a subsequent REFRESH, removes dropped fields from the FDT;
- The REMOVE_REPLICATION function stops all replication processing and deletes the replication system files;
- The RENAME function changes the database name or the name of a loaded file;
- The RENUMBER function renumbers a loaded file or exchanges the numbers of loaded files;
- The REPLICATION_FILES function creates the systems files required for Adabas - Adabas replication;
- The RESET function removes entries from the UCB;
- The RESET_REPLICATION_TARGET function resets the replication target flag of Adabas files;
- The REUSE function controls the reuse of data storage space or ISNs by Adabas;
- The SECURITY function sets the security mode of the database;
- The SYFMAX function specifies the maximum number of values generated for a system generated multiple-value field in the file specified.

ADADCU

Decompression of data

The ADADCU utility decompresses records to be used with a non-Adabas application program, or as input for the compression utility ADACMP. The file to be decompressed must be unloaded from the database (unload utility ADAULD) before it can be used as input for this utility. With ADADCU, complete records can be decompressed, fields can be rearranged within a record, default lengths can be changed, some types of fields can be truncated, formats can be changed and space can be allocated for the addition of new fields.

ADAFDU

File definition

The file definition utility ADAFDU defines a file in the database. It only loads the FCB and the FDT into the database and allocates the requested space for ASSO and DATA for the specified file.

ADAFIN

File information report

The ADAFIN utility displays information about one or more files, e.g. FDT, descriptor statistics and the fill percentage of blocks in the Data Storage, Normal Index and Upper/Main Index.

ADAMUP

Mass add and delete

The ADAMUP utility adds or deletes large numbers of records to/from a file in the database.

ADAOPR

Operator utility

The operator utility is used to operate the Adabas nucleus.

ADAORD

Reorder database or files, export/import files

The reorder utility ADAORD provides functions to reorganize a database or files within a database (REORDER function) and to migrate files between databases (EXPORT and IMPORT functions).

ADAREP

Database report

The ADAREP utility produces the database status report. This report contains information about the current physical layout and logical contents of the database.

The information in this report includes the following: the amount and location of the space currently allocated for the Associator and Data Storage; the amount and location of unused space available for Associator and Data Storage; database file summary; checkpoint information;

Security information; information about each file in the database (space allocation, space available, number of records loaded, MAXISN setting, field definitions).

ADASEL

This utility selects records from Protection log files and WORK containers.

ADAULD

File unloading

The unload utility ADAULD unloads a file from a database or an Adabas backup copy and produces compressed records with the same format as those produced by the compression utility ADACMP. Unloaded records may be used as input for the decompression utility ADADCU or with the mass update utility ADAMUP. Records can be unloaded from a database in the sequence in which they are currently stored in Data Storage, in the sequence of a descriptor or in ISN sequence. However, records can only be unloaded from a backup copy in the order in which they were stored by the utility.

4 ADABCK (Dump And Restore Database Or Files)

- Functional Overview 12
- Procedure Flow 15
- Checkpoints 17
- Control Parameters 18
- Restart Considerations 40

This chapter describes the utility "ADABCK".

Functional Overview

The backup utility ADABCK provides protection against database corruption by creating Adabas backup copies. ADABCK should be used at regular intervals.

The utility dumps or restores a database or selected files from/to a database.

ADABCK is able to process input files that were created on either the same platform or on a platform with a different endian mode. The format in which the file was written is recognized by the restore operation. During the restore, all endian-mode dependent data are converted to the requirements of the target platform.

Making use of the internal structure of the database, this utility provides optimum performance. Unused blocks do not have to be read and can be omitted when dumping. Even though such blocks are not included in the Adabas backup copy, they can be re-created during a restore.

Furthermore, a backup copy may be directed to stdout in order to support the piping of the backup data (this feature is only available on Linux platforms). This feature is enabled by setting the environment variable (BCK001) to '-' (minus). In this case, the output messages are directed to stderr. The RESTORE and OVERLAY functions can also be used in this way, i.e. a backup copy can be read from stdin. In this case, the ADABCK control statements must be given in the command line. See *Adabas Basics*, *Using Utilities* in the Adabas documentation for more information.

The following functions are available:

- The COPY function copies an Adabas backup copy. A backup data set can only be duplicated on a machine with the same endian mode - attempting to duplicate a backup on a machine with a different endian mode will be rejected;
- The DUMP function dumps a database or selected files from a database to one or more sequential files, which is called an Adabas backup copy. The nucleus may be active and parallel updates are permitted on the files to be dumped while the dump is in progress. The DUMP function writes data in the endian mode of the processor;
- The EXU_DUMP function dumps a database or selected files from a database to one or more sequential files, which is called an Adabas backup copy. Only ACC users are permitted on the files to be dumped while the dump is in progress;
- The IOSTAT function prints information about the data transfer rate and the I/O waiting times.
- The OVERLAY function restores selected files or a database. The files to be restored may already be loaded in the database: ADABCK performs an implicit delete before restoring such files;

- The READ_CHECK function checks the readability (i.e. absence of parity errors) and completeness of the Adabas backup copy. These checks ensure that the dump file can be read by the RESTORE or OVERLAY function;
- The RESTORE function restores a database or selected files from an existing Adabas backup copy. If there are no security definitions for the files in the target database, the corresponding entries (as they were defined at the time the files were dumped) are set up in the security table when the file is restored;
- The list functions CONTENTS, FILES and SUMMARY display information about an Adabas backup copy. When the list functions are used, the DBID does not have to be entered first.

The functions DUMP, EXU_DUMP, OVERLAY and RESTORE are mutually exclusive and only one of them may be executed during a single run of this utility. The list functions can only be used together with the READ_CHECK, RESTORE or OVERLAY function.

If you perform the RESTORE or OVERLAY function and the database is too small or database containers are missing, ADABCK will automatically increase the size of the database or create the missing containers.

The functions RESTORE and OVERLAY allow you to create an encrypted database from an Adabas backup copy of a non-encrypted database, and vice versa. The target database is restored with the encryption settings provided by the control parameters ENCRYPTION and KMSTARGET. The new target database is restored or overlayed completely and must not already exist. The ENCRYPTION and KMSTARGET parameters are not permitted for existing databases because the encryption settings of the existing target database are always retained. An encrypted backup copy is restored unencrypted if the existing target database is not encrypted. In contrast, unencrypted backup copies are always restored in encrypted form if the existing target database is encrypted. An encrypted backup copy will be restored using the encryption settings of the existing target database. Different encryption algorithms and keys in the backup copy and database are possible but of course require access to both keys via the configured KMS. To use the encryption functionality, the Adabas Encryption for Linux (AEL) license is required.

Notes:

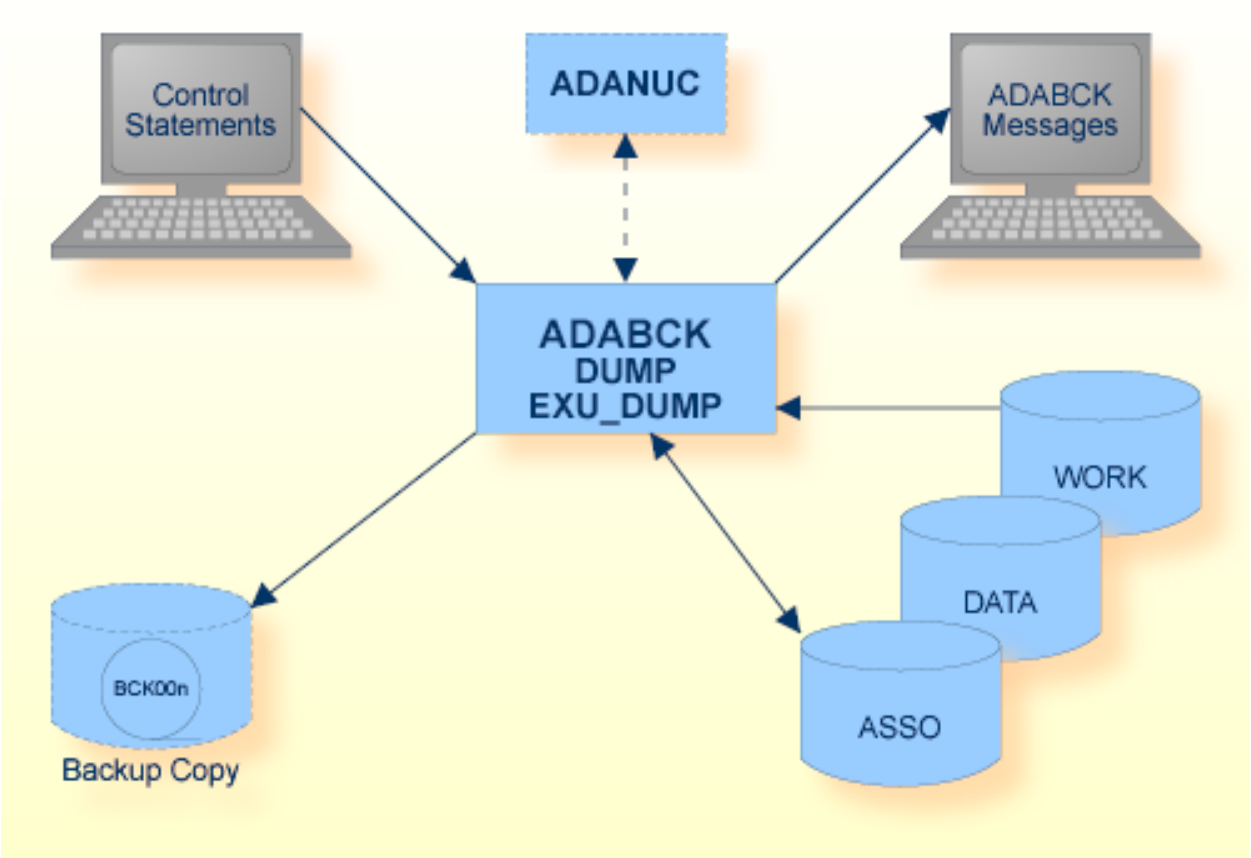
1. The RESTORE and OVERLAY functions can process backup files created with earlier Adabas versions, but usually not backup files created with later Adabas versions. However, it is possible to restore with earlier Adabas versions if the structures did not change. For example, version 6.6 backup files can be restored with version 6.5, unless superdescriptors with character-set encoding are used. If features are not supported by the earlier version, the structure level check will fail.
2. The RESTORE option PARALLEL=MULTIPROCESS is not supported for backup files created on a platform with a different endian mode. However, a backup that is split into several extents can be loaded.

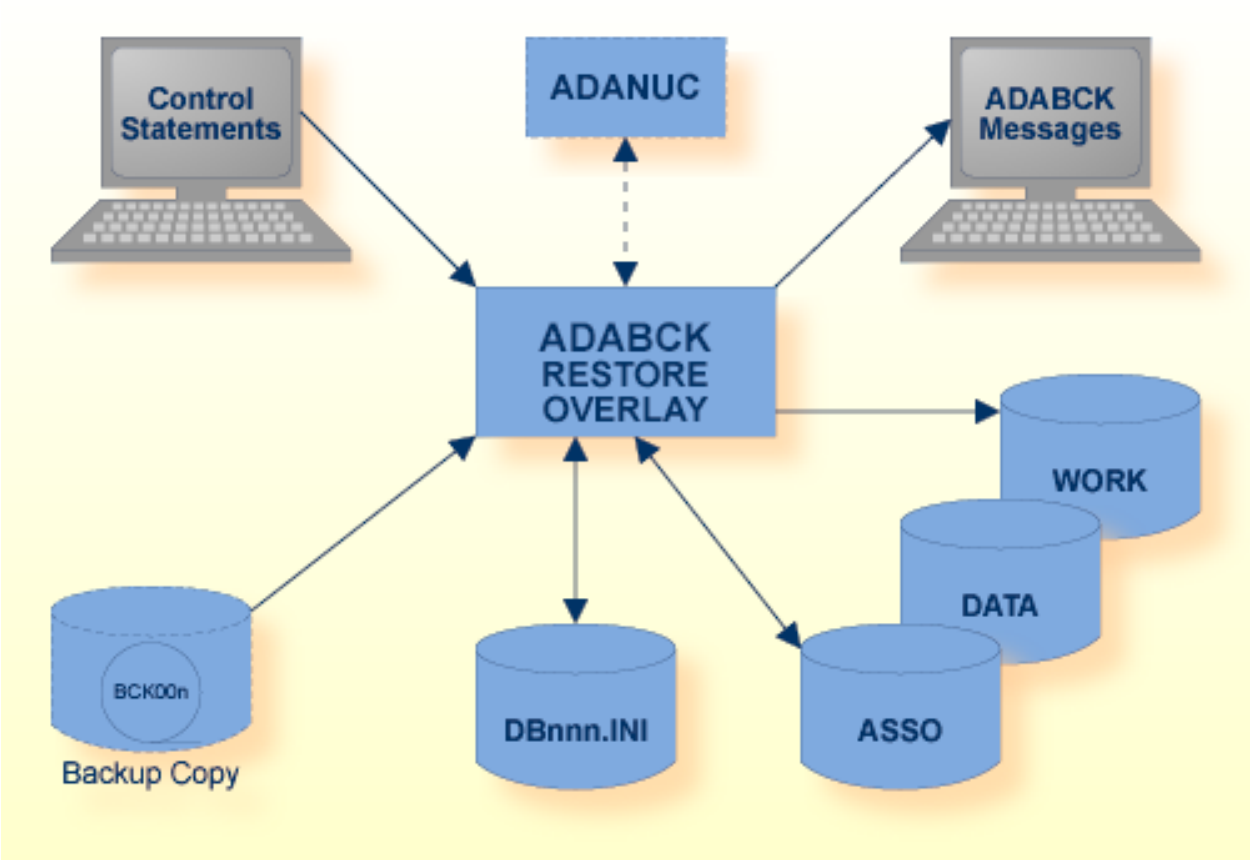
Caution: If you do not use the Adabas INI files, but instead use environment variables to specify the container file names, and if you forget to assign the environment variables/logical names before you start ADABCK, a copy of the database will be created in the database directory. If you perform a file overlay or restore when the Adabas nucleus is active, and the database has to be extended, the database is extended by the nucleus, and not by ADABCK. In this case, the nucleus extends the database even if OPTIONS=AUTO_EXPAND was NOT specified. If you use environment variables to specify the database containers, you must consider the following when a new container has to be created for the restore/overlay: it is important that the nucleus was started with the correct environment variable settings for the new container - because the new containers are created by the nucleus, specifying the environment variable for the ADABCK process has no effect.

Any online backup operation will fail if the running nucleus is canceled. Be aware that calling adastop may result in canceling the nucleus.

This utility is a single-function utility. For more information about single- and multi-function utilities. See *Adabas Basics*, *Using Utilities* in the Adabas documentation for more information.

Procedure Flow





Data Set	Environment Variable / Logical Name	Storage Medium	Additional Information
Associator	ASSOx	Disk	
Backup copy	BCK00n	Disk, Tape (see note 1), stdin (see note 2), stdout (see note 3)	Output of DUMP/EXU_DUMP function,input for other functions
	BCKOUT	Disk (see note 1)	Output of COPY function
Data storage	DATAx	Disk	
DBnnn.INI		Disk	*Adabas Extended Operations*
Control statements	stdin		*Utilities*
ADABCK messages	stdout (see note 4), stderr (see note 5)		*Messages and Codes*
Work	WORK1	Disk	

Notes:

1. A named pipe can be used for this sequential file (Linux platforms only). See *Adabas Basics, Using Utilities* in the Adabas documentation.

2. For functions other than DUMP or EXU_DUMP (BCK001 only).
3. For DUMP or EXU_DUMP (BCK001 only).
4. If BCK001 is not stdout.
5. If BCK001 is stdout.

The sequential files BCK00n can have multiple extents. For detailed information about sequential files with multiple extents, see *Adabas Basics*, *Using Utilities* in the Adabas documentation.

Checkpoints

The following table shows the nucleus requirements for each function and the checkpoint written:

Function	Nucleus must be active	Nucleus must NOT be active	Nucleus is NOT required	Checkpoint written
CONTENTS			X	-
COPY			X	-
DUMP	X(see note 1)		X	SYNX
EXU_DUMP	X(see note 1)		X	SYNX
FILES			X	-
NEW_PLOG				SYNC
OVERLAY		X(see note 2)	X(see note 3)	SYNP
READ_CHECK			X	-
RESTORE		X(see note 2)	X(see note 3)	SYNP
SUMMARY			X	-

Notes:

1. Nucleus only required when AUTORESTART is pending at the end of this function.
2. For restore of database or system files.
3. For restore of files.

Control Parameters

The following control parameters are available:

```
     CONTENTS

     COPY [= number]

M    DBID = number

     DUMP = {*|(number[-number][,number[-number]]...)}
            [,BLOCKSIZE = number [K|M]]
D           [{,DRIVES = number} |
D           {,[NO]DUAL } ]
            [,ET_SYNC_WAIT = number]
D           [,[NO]NEW_PLOG]
            [,REPLICATION]
            [PRIMARY_ONLY]
     EXU_DUMP = {*|(number[-number][,number[-number]]...)}
          [[,BLOCKSIZE = number [K|M]]
D         [{,DRIVES = number} |
D         {,[NO]DUAL} ]
D         [,[NO]NEW_PLOG]
                [,REPLICATION]

     FILES = { * | (number[-number][,number[-number]]...)}

     IOSTAT

     OVERLAY = {*|(number[-number][,number[-number]]...)}
               [,ENCRYPTION = keyword]
         [,FMOVE [=(number [,number [-number]]...)]]
         [,FORMAT = (keyword [,keyword])]
         [,KEEP_FILE_ALLOC]
               [,KMSTARGET = string]
         [,NEW_DBID = number]
         [,RENUMBER = (number[-number] [,number [-number]]...)]]
               [,REPLICATION]

     PARALLEL = keyword

     READ_CHECK

     RESTORE = {*|(number[-number][,number[-number]]...)}
               [,ENCRYPTION = keyword]
         [,FMOVE [=(number [,number [-number]]...)]]
         [,FORMAT = (keyword [,keyword])]
         [,KMSTARGET = string]
         [,NEW_DBID = number]
```

```
        [,RENUMBER = (number[-number] [,number [-number]]...)]]
              [,REPLICATION]

     SUMMARY
```

- CONTENTS
- COPY
- DBID
- DUMP
- EXU_DUMP
- FILES
- IOSTAT
- OVERLAY
- PARALLEL
- READ_CHECK
- RESTORE
- SUMMARY

CONTENTS

```
CONTENTS
```

This parameter displays a list of files in an Adabas backup copy created with the DUMP or EXU_DUMP function.

Example

```
adabck cont
%ADABCK-I-STARTED,      30-OCT-2015 11:42:37, <Version number>
Files dumped on 30-OCT-2015 10:51:14

Database 34, GENERAL-DATABASE

File     4, Update-log      , loaded on 17-SEP-2014 14:44:19
File     9, EMPLOYEES       , loaded on  8-OCT-2008 17:59:40
File    14, miscellaneous   , loaded on 11-JUN-2015 13:22:19
File    17, Timezone        , loaded on 19-SEP-2014 11:44:42
File    19, LARGE           , loaded on  2-SEP-2014 15:37:18
File    51, PCA24SYSF1      , loaded on 14-APR-2014 16:55:22
File    91, ADAOS-2544      , loaded on  8-APR-2015 13:19:27
File    95, P299255         , loaded on 20-MAR-2014 11:35:30
File    98, ADAOS-4591      , loaded on 16-JUL-2015 10:03:16
File  1009, LOBFILE of 9    , loaded on  8-OCT-2008 17:59:40



%ADABCK-I-IOCNT, 1 IOs on dataset BCK001
%ADABCK-I-TERMINATED,   30-OCT-2015 11:42:37, elapsed time: 00:00:00
```

COPY

```
COPY [= number]
```

This function creates a new file from an existing Adabas backup copy. The input file (BCK0xx) and the output file (BCKOUT) must be on disk, where xx is either the specified number, or 01 if no number is explicitly specified.

DBID

```
DBID = number
```

This parameter selects the database to be used.

DUMP

```
DUMP = { * | (number[-number][,number[-number]]...)}
       [,BLOCKSIZE = number [K|M]]
       [ {,DRIVES = number} |
       {, [NO]DUAL } ]
       [,ET_SYNC_WAIT = number ]
       [,[NO]NEW_PLOG ]
       [,REPLICATION] ↩
```

At the file level, this function dumps the files specified by the numbers in the list. LOB files specified are ignored, but the LOB files assigned to all base files are dumped too. An asterisk '*' specifies that the complete database is to be dumped. Parallel updates are permitted on the files to be dumped while the dump is in progress.

If the nucleus is running in parallel (online backup), ADABCK must ensure that all transactions affecting the dumped files are completed by all users before ADABCK terminates. This is called ET synchronization - please refer to the section *ET Synchronization* in *Administration* for further information. If you perform a dump at the file level with the option NONEW_PLOG, the ET synchronization is performed at the file level; otherwise the ET synchronization is performed for the complete database.

If you specify files with referential constraints, all files connected to these files via referential constraints must also be specified in order to maintain referential integrity.

- BLOCKSIZE = number[K|M]
- DRIVES = number
- [NO]DUAL
- ET_SYNC_WAIT = number
- [NO]NEW_PLOG
- REPLICATION

- PRIMARY_ONLY

BLOCKSIZE = number[K|M]

This parameter can be specified to change the I/O transfer blocksize. If PARALLEL is specified, the default blocksize is 512 KB. The following values can be specified: 64KB, 128KB, 256KB, 512KB, 1MB, 2MB, ... 12MB. The blocksize specified will be used in a subsequent RESTORE function.

DRIVES = number

This parameter limits the maximum number of output devices to be operated in parallel. It can be used to split a backup file into several extents. The output is sent to BCK0xx.

The default value is 1 and the maximum value is 10.

The parameters DRIVES and DUAL are mutually exclusive, and only one of them may be specified in a given call of the DUMP function.

[NO]DUAL

DUAL specifies that two physical copies of the dumped information are to be created. The output is sent to BCK001 and BCK002.

The default is NODUAL.

The parameters DUAL and DRIVES are mutually exclusive, and only one of them may be specified in a given call of the DUMP function.

ET_SYNC_WAIT = number

This parameter defines the time (in seconds) that ADABCK waits for ET-logic users to come to ET status at the end of the DUMP function: if a transaction is already active for the number of seconds (or longer) specified by ET_SYNC_WAIT when the ET synchronization begins, its wait time is 0. Otherwise, the wait time for a transaction, in seconds, is the value specified for the parameter minus the number of seconds that the transaction is already active when the ET synchronization begins. Transactions not yet terminated at the end of their wait times are rolled back.

If this parameter is omitted, the ET synchronization waits until all open transactions are terminated using the normal Adabas timeout logic (ADANUC parameter TT).

The minimum value is 1 and the maximum value is 32767.

Notes:

1. If you forgot to specify the ET_SYNC_WAIT parameter for ADABCK, and ADABCK is hanging because of open transactions, you can do one of the following to let ADABCK continue: tempor-

arily set TT to a small value -after ADABCK terminates, you can set the value back to its original value; or stop the user(s) that have open transactions (with ADAOPR STOP).

2. If updates were performed in the database during the ET synchronization, all modified blocks must be written to the database containers and to the backup copy when all open transactions have been committed or rolled back. Therefore, the total time for the ET synchronization, which can be displayed via the IOSTAT parameter, may be longer than the time specified with the ET_SYNC_WAIT parameter.

[NO]NEW_PLOG

This option specifies whether or not to close the protection log file and create a new log file at the end of the DUMP function.

The default for a database dump is NEW_PLOG, and for a file dump it is NONEW_PLOG.

If NEW_PLOG is specified, it is safe to remove the protection log files after the DUMP function. See *Adabas Basics > Locations of Database Containers, Backup Files, and Protection Logs* in the *Adabas for Linux and Cloud* documentationfor specific notes on protection logs.

Caution: Before V6.3 SP1 Fix 13, the default for a file dump was NEW_PLOG. In most cases, this change is of no consequence, but if you really need the PLOG switch, you must specify NEW_PLOG explicitly.

REPLICATION

The parameter REPLICATION is relevant only for customers who are using the Adabas Event Replicator with Adabas - Adabas Replication.

This parameter should be specified if you want to use ADABCK for the Adabas - Adabas replication initial state processing. If you specify this parameter, the status of the replications of the files to be dumped is automatically updated.

For further information refer to ADAOPR CHANGE_REPLICATION_STATUS.

PRIMARY_ONLY

The Parameter PRIMARY_ONLY is for customer who are using Adabas Cluster.

This parameter allows only to take dump of primary database, if the secondary database id is provided, ADABCK will terminate with success but no BCKxxx file will be created.

EXU_DUMP

```
EXU_DUMP = {*|(number[-number][,number[-number]]...)}
           [,BLOCKSIZE = number [K|M]]
           [ {,DRIVES = number} |
           {,[NO]DUAL} ]
           [,[NO]NEW_PLOG]
           [,REPLICATION] ↩
```

At the file level, this function dumps the files specified by the numbers in the list. LOB files specified are ignored, but the LOB files assigned to all base files are dumped too. An asterisk '*' specifies that the complete database is to be dumped. Only ACC users are permitted on the files to be dumped while the dump is in progress. ET-synchronization is not required.

If you specify files with referential constraints, all files connected to these files via referential constraints must also be specified in order to maintain referential integrity.

BLOCKSIZE = number[K|M]

This parameter can be specified to change the I/O transfer blocksize. If PARALLEL is specified, the default blocksize is 512 KB. The following values can be specified: 64KB, 128KB, 256KB, 512KB, 1MB, 2MB, ... 12MB. The blocksize specified will be used in a subsequent RESTORE function.

DRIVES = number

This parameter limits the maximum number of output devices to be operated in parallel. It can be used to split a backup file into several extents. The output is sent to BCK0xx.

The default value is 1 and the maximum value is 10.

The parameters DRIVES and DUAL are mutually exclusive, and only one of them may be specified in a given call of the DUMP function.

[NO]DUAL

DUAL specifies that two physical copies of the dumped information are to be created. The output is sent to BCK001 and BCK002.

The default is NODUAL.

The parameters DUAL and DRIVES are mutually exclusive, and only one of them may be specified in a given call of the DUMP function.

[NO]NEW_PLOG

This option specifies whether or not to close the protection log file and create a new log file at the end of the EXU_DUMP function.

This option must not be used if dumping single files.

The default is NEW_PLOG for EXU_DUMP=*.

REPLICATION

The parameter REPLICATION is relevant only for customers who are using the Adabas Event Replicator with Adabas - Adabas Replication.

This parameter should be specified if you want to use ADABCK for the Adabas - Adabas replication initial state processing. If you specify this parameter, the status of the replications of the files to be dumped is automatically updated.

For further information refer to ADAOPR CHANGE_REPLICATION_STATUS.

Examples for DUMP/EXUDUMP

Example 1

The database is dumped to three output devices in parallel.

```
adabck db=34 parallel=multi_process dump=\* drives=3
%ADABCK-I-STARTED,      30-OCT-2015 11:05:25, <version number>
%ADABCK-I-DBOFF, database 34 accessed offline

Database dumped on 30-OCT-2015 11:05:25

Database 34, GENERAL-DATABASE

File     1, CHECKPOINT-FILE , loaded on  4-SEP-2014 13:52:43
File     2, SECURITY-FILE   , loaded on  4-SEP-2014 13:52:43
File     3, USER-DATA-FILE  , loaded on  4-SEP-2014 13:52:43
File     4, Update-log      , loaded on 17-SEP-2014 14:44:19
File     9, EMPLOYEES       , loaded on  8-OCT-2008 17:59:40
File    14, miscellaneous   , loaded on 11-JUN-2015 13:22:19
File    17, Timezone        , loaded on 19-SEP-2014 11:44:42
File    19, LARGE           , loaded on  2-SEP-2014 15:37:18
File    30, FILE-30         , loaded on 31-MAR-2015 13:40:33
File    51, PCA24SYSF1      , loaded on 14-APR-2014 16:55:22
File    80, P295170         , loaded on  4-AUG-2015 12:42:43
File    91, ADAOS-2544      , loaded on  8-APR-2015 13:19:27
File    95, P299255         , loaded on 20-MAR-2014 11:35:30
File    98, ADAOS-4591      , loaded on 16-JUL-2015 10:03:16
File   101, COLLATION-TESTS , loaded on 14-APR-2014 16:47:30
```

```
File   111, TESTOPT         , loaded on 29-NOV-2011 10:34:23
File   113, lob_LB          , loaded on 23-JUN-2015 15:48:48
File   146, XMA-REPOSITORY  , loaded on 10-DEC-2014 09:39:52
File   215, ADAOS-4647      , loaded on 27-APR-2012 15:52:49
File  1009, LOBFILE of 9    , loaded on  8-OCT-2008 17:59:40
File  1080, LOBFILE of 80   , loaded on  4-AUG-2015 12:42:43
File  1113, LOBFILE of 113  , loaded on 23-JUN-2015 15:48:48
File 22111, LOBFILE of 111  , loaded on 29-NOV-2011 10:34:23


%ADABCK-I-IOCNT, 2 IOs on dataset WORK
%ADABCK-I-IOCNT, 135 IOs on dataset DATA
%ADABCK-I-IOCNT, 275 IOs on dataset ASSO
%ADABCK-I-IOCNT, 41 IOs on dataset BCK001
%ADABCK-I-IOCNT, 34 IOs on dataset BCK002
%ADABCK-I-IOCNT, 80 IOs on dataset BCK003
%ADABCK-I-TERMINATED,   30-OCT-2015 11:05:26, elapsed time: 00:00:01
```

Example 2

File 215 is dumped, and two physical copies of the backup are created. Only ACC users are allowed on file 30 while the dump is in progress.

```
adabck db=34 exu_dump=215 dual
%ADABCK-I-STARTED,      30-OCT-2015 10:45:43, <version number>
%ADABCK-I-DBON, database 34 accessed online

Files dumped on 30-OCT-2015 10:45:44

Database 34, GENERAL-DATABASE

File   215, ADAOS-4647      , loaded on 27-APR-2012 15:52:49


%ADABCK-I-IOCNT, 51 IOs on dataset DATA
%ADABCK-I-IOCNT, 29 IOs on dataset ASSO
%ADABCK-I-IOCNT, 40 IOs on dataset BCK001
%ADABCK-I-IOCNT, 40 IOs on dataset BCK002
%ADABCK-I-TERMINATED,   30-OCT-2015 10:45:44, elapsed time: 00:00:01
```

Example 3

All base files in the database with a file number between 91 and 99 or equal to 51 or between 4 and 19 are dumped (including the corresponding LOB files, even if they are not in the specified file ranges). ADABCK allows a maximum of 10 seconds for ET logic users to come to ET status.

```
adabck db=34 dump=\(91-99,51,4-19\) et_sync_wait=10
%ADABCK-I-STARTED,      30-OCT-2015 10:51:14, <version number>
%ADABCK-I-DBON, database 34 accessed online

Files dumped on 30-OCT-2015 10:51:14

Database 34, GENERAL-DATABASE

File     4, Update-log      , loaded on 17-SEP-2014 14:44:19
File     9, EMPLOYEES       , loaded on  8-OCT-2008 17:59:40
File    14, miscellaneous   , loaded on 11-JUN-2015 13:22:19
File    17, Timezone        , loaded on 19-SEP-2014 11:44:42
File    19, LARGE           , loaded on  2-SEP-2014 15:37:18
File    51, PCA24SYSF1      , loaded on 14-APR-2014 16:55:22
File    91, ADAOS-2544      , loaded on  8-APR-2015 13:19:27
File    95, P299255         , loaded on 20-MAR-2014 11:35:30
File    98, ADAOS-4591      , loaded on 16-JUL-2015 10:03:16
File  1009, LOBFILE of 9    , loaded on  8-OCT-2008 17:59:40


%ADABCK-I-IOCNT, 715 IOs on dataset DATA
%ADABCK-I-IOCNT, 1145 IOs on dataset ASSO
%ADABCK-I-IOCNT, 1195 IOs on dataset BCK001
%ADABCK-I-TERMINATED,   30-OCT-2015 10:51:16, elapsed time: 00:00:02
```

Files 1, 2, 4, 6, 8, 10, 11 and 13 are dumped. ADABCK allows a maximum of 5 seconds for ET-logic users to come to ET status.

FILES

```
FILES = { * | (number[-number][,number[-number]]...)}
```

This parameter displays status information of the specified files in a dump file.

IOSTAT

```
IOSTAT
```

If this parameter is specified, the data transfer rate and the I/O (waiting) times on the various devices are printed at the end of ADABCK processing.

Example:

```
adabck db=36 parallel=multi_process dump=\* drives=3 iostat
...
-----------------------------------------------------------------
Dump Method     :  parallel
Blocksizes      :  DB:             512 KB     BCK:          512 KB
DB I/O time     :  total:        27.09 sec    average:     8084 us
BCK  1 I/O time :  total:         1.16 sec    average:     7606 us
BCK  2 I/O time :  total:         0.00 sec    average:      944 us
BCK  3 I/O time :  total:         1.24 sec    average:     1375 us
Wait rates      :     waits    nowaits        rate   mreq
DB              :      1439       1898         43%    8
Transfer rate   :  15215 KB/sec
-----------------------------------------------------------------

%ADABCK-I-IOCNT, 2 IOs on dataset WORK
%ADABCK-I-IOCNT, 3147 IOs on dataset DATA
%ADABCK-I-IOCNT, 229 IOs on dataset ASSO
%ADABCK-I-IOCNT, 153 IOs on dataset BCK001
%ADABCK-I-IOCNT, 2 IOs on dataset BCK002
%ADABCK-I-IOCNT, 906 IOs on dataset BCK003
```

The IOSTAT statistics display the following information:

Dump Method
Either parallel or non-parallel, depending on the setting of the PARALLEL parameter.

DB I/O time
The total I/O time in seconds and the average time per I/O operation in microseconds for the access to the ASSO and DATA containers.

BCK n I/O time
The total I/O time in seconds and the average time per I/O operation in microseconds for the access to the backup files.

Note: The I/O time measured is the time required for the I/O system functions. This may be different from the physical I/O times actually required to accessing the disks because of caches in the operating system or in the storage system and because of usage of asynchronous I/O.

Wait rates (only for dump method parallel)
For a parallel backup/restore, the I/Os for the database containers are performed asynchronously. The wait rate shows for how many ASSO or DATA I/Os a wait operation is required. mreq is the maximum number of parallel I/O requests for database containers.

Note: Only the I/Os for the real backup or restore are counted. During the startup phase of ADABCK, some additional I/Os are required; therefore the sum of wait and nowait I/Os is less than the sum of ASSO and DATA I/Os.

BF sync count (only for a backup in online mode)
In the case of a backup in online mode during a buffer flush, synchronization with the nucleus is required in order to guarantee that the modified database blocks written to disk by the buffer flush are also written to the backup file(s). The BF sync count is the number of these buffer flush synchronizations.

ET sync time (only for a backup in online mode)
At the end of a backup in online mode, an ET synchronization is required, i.e. ADABCK must wait until all ET logic users come to ET status. The ET sync time is the time required for this ET synchronization.

Transfer rate
This is the number of kilobytes read from or written to the backup file(s) per second.

Notes:

1. For the transfer rate, only the pure backup/restore time is taken into consideration, but not the time required for the preparation of the backup/restore. Therefore, the transfer rate may be higher than the transfer rate you would get if you compute the transfer rate based on the total elapsed time of ADABCK.
2. In the case of small backups, rounding errors may occur in the computation. Therefore, for very small backups the transfer rate is not displayed, because the value would be too inaccurate.
3. Because usually many database blocks are not filled completely, and because only the net data are copied to the backup file(s), the transfer rate is less than the rate you would get if you consider the processed database space.

OVERLAY

```
OVERLAY = {*|(number[-number][,number[-number]]...)}
          [,ENCRYPTION = keyword]
          [,FMOVE [=(number [,number [-number]]...)]]
          [,FORMAT = (keyword [,keyword] ) ]
          [,KEEP_FILE_ALLOC]
          [,KMSTARGET = string]
          [,NEW_DBID = number]
          [,RENUMBER = (number[-number] [,number [-number]]...)]]
          [,REPLICATION]
```

This function restores the files specified by the numbers in the list at file level. LOB files specified are ignored, but the LOB files assigned to all base files are restored too. The files to be restored may already be loaded in the database. ADABCK performs an implicit delete before restoring such files. If only one file of a LOB group is overlaid, the other file of the LOB group is also deleted. An asterisk ('*') specifies that a restore is to be made at the database level. Exclusive control over the database container files is required.

Only the specified files are overlayed, even if there are referential integrity constraints to other files; these referential integrity constraints are removed.

ENCRYPTION = keyword

This parameter specifies that the created database is encrypted and assigns the encryption algorithm. The keyword can take the values AES_256_XTS, AES_128_XTS and NO. Depending on the keyword specified, the ASSO and DATA container files are encrypted using XTS Advanced Encryption Standard with a key length of 256 bits (AES_256_XTS), a key length of 128 bits (AES_128_XTS), or they are not encrypted (NO).

The default value is NO.

Notes:

1. Database encryption cannot be disabled.
2. The ENCRYPTION parameter can only be used for a full database restore (overlay=* or restore=*) and the database must not exist. If the database exists, the encryption settings of the database are retained.
3. File restore/overlay operations is applied according to the encryption settings of the existing database.

FMOVE [=(number [,number [-number]]...)]

If this keyword is specified, ADABCK reallocates all files to be overlayed or the specified subset rather than attempting to restore them in the same block ranges as in the backup. Using this keyword reduces the number of file extents as much as possible.

FORMAT = (keyword [,keyword])

The keywords ASSO and/or DATA may be specified. This parameter is used to format Associator and/or Data Storage blocks. When restoring at the file level, only blocks contained in the unused areas of the files' extents are formatted.

KEEP_FILE_ALLOC

If this parameter is specified, ADABCK tries to keep the allocation of the file as it currently is in the database, as opposed to restoring it with the same block ranges as on the backup. This keyword can, for example, be used when a file has been reorganized since the backup was made or also if more space has since been preallocated to the file. If the file on the backup has more blocks allocated than are currently available in the database, the remaining blocks will be allocated in an arbitrary location. This keyword can only be used in conjunction with a file list.

KMSTARGET = string

This parameter specifies the key management system that is used if the created database is encrypted. See *Adabas Security Features* > *Adabas Encryption for Linux* > *Key Management System* in the Adabas documentation for all supported key management systems.

The default value is FILE.

Note: You must specify the KMSTARGET parameter before the ENCRYPTION parameter.

Examples for KSMTARGET

Please see **Examples for RESTORE/OVERLAY using ENCRYPTION and KMSTARGET**.

NEW_DBID = number

This parameter can be used to change the identifier of the database to be restored. This parameter can only be specified when restoring a complete database.

A new identifier can be used to restore a backup copy of an active database into a different set of container files. The new identifier may not be identical to that of another active database.

If this parameter is omitted, the database identifier remains unchanged.

Important: If a backup of e.g. database 100 is restored into an existing database 100, but a NEW_DBID parameter with a different dbid is used, the database 100 will become inaccessible. When accessing the database the Adabas utilities will report a DBID mismatch because of a wrong dbid in the GCB RABN.

In such a case the following steps need to be performed to get access again to the database:

1. Rename the database directory from db100 to db101, change the container extensions from 100 to 101, and rename the DB100.INI file to DB101.INI.
2. Update DB101.INI: Edit the content by changing the path and file name of each environment variable value from 100 to 101.
3. Update assign.bsh: Edit the content by changing the path and file name of each environment variable value from 100 to 101.
4. Update ADABAS.INI: In the DB_LIST section edit all values from the old dbid value 100 to the new dbid val-ue 101.

RENUMBER = (number[-number] [,number [-number]]...)

RENUMBER is used to renumber the files to be overlayed in the target database. The following restrictions and requirements apply:

- There must be a 1:1 relationship between the files specified in the OVERLAY file list and the RENUMBER file list.
- If you specify a range in the OVERLAY file list, the corresponding range in the RENUMBER file list must be the same size.
- Normally it is not necessary to specify LOB files in the OVERLAY file list. However, if the LOB file is also to be renumbered, the LOB file must also be specified.
- Files may occur more than once in the OVERLAY file list, for example: (11-55),(44-99). In this case, you are not allowed to specify different target file numbers for the same source file numbers. For the example file list, it is correct to specify RENUMBER=(1011-1055,1044-1099), whereas RENUMBER=(1011-1055,2044-2099) is incorrect.
- It is not allowed to renumber more than one file to the same target file number.

REPLICATION

The parameter REPLICATION is relevant only for customers who are using the Adabas Event Replicator with Adabas - Adabas Replication.

This parameter should be specified if you want to use ADABCK for the Adabas - Adabas replication initial state processing. If you specify this parameter, the Adabas file is automatically marked as a replication target file.

For further information refer to ADAOPR CHANGE_REPLICATION_STATUS.

PARALLEL

```
PARALLEL = keyword
```

This parameter can be specified to increase processing speed when creating/restoring from backups on slow devices by using parallel devices. The keyword MULTI_PROCESS can be used. If PARALLEL=MULTI_PROCESS is specified, the default value of the BLOCKSIZE parameter changes to 512 KB.

The ADABCK operation is only performed in parallel if the number of backup files (ADABCK subparameter DRIVES for DUMP or EXU_DUMP) is greater than 1.

Notes:

1. In the case of ADABCK RESTORE or ADABCK OVERLAY, PARALLEL must be specified before the OVERLAY or RESTORE parameter.
2. The PARALLEL parameter is not supported on Windows platforms.

3. It is possible to pass the output of ADABCK DUMP or ADABCK EXU_DUMP to named pipes, which can be directly used as input for an ADABCK RESTORE or ADABCK OVERLAY in order to copy a database or some files from one database to another database.
4. The PARALLEL parameter does not improve the performance of the READ_CHECK function.
5. The use of PARALLEL=MULTI_PROCESS is not recommended for the DUMP operation if the data is to be restored on a computer with a different endian mode - the RESTORE operation will reject backup files created with PARALLEL=MULTI_PROCESS on a computer with a different endian mode.

READ_CHECK

```
READ_CHECK
```

This function checks the readability (i.e. absence of parity errors) and completeness of the Adabas backup copy. These checks are made to ensure that the dump file can be used to restore the database or files with the RESTORE or OVERLAY function of this utility.

RESTORE

```
RESTORE = {*|(number[-number][,number[-number]]...)}
          [,ENCRYPTION = keyword]
          [,FMOVE [=(number [,number [-number]]...)]]
          [,FORMAT = (keyword [,keyword] ) ]
          [,KMSTARGET = string]
          [,NEW_DBID = number]
          [,RENUMBER = (number[-number] [,number [-number]]...)]]
          [,REPLICATION]
```

This function restores the files specified by the numbers in the list at the file level. LOB files specified are ignored, but the LOB files assigned to all base files are restored too. If a file list is given, the files to be restored must not be loaded in the database. An '*' specifies that a restore is to be made at the database level. In this case, the files may already be loaded in the database and will implicitly be deleted or substituted by files in the dump with identical file numbers. Exclusive control over the database container files is required.

Only the specified files are restored, even if there are referential integrity constraints to other files; these referential integrity constraints are removed.

Notes:

1. You can use `RESTORE=*` only if the dump file was created with `DUMP=*` or `EXU_DUMP=*`.
2. A backup created on a platform with a different endian mode cannot be restored if the backup was created with the `PARALLEL=MULTI_PROCESS` option.
3. During a full database restore, ADABCK checks the database entry in the *ADABAS.INI* file. If the restored database is not present in the database list, ADABCK attempts to recover the entry

using the information from the dump file. ADABCK also checks whether the corresponding *DBnnn.INI* file exists. If the file does not exist, ADABCK creates the *DBnnn.INI* file with the default database settings and restored container entries.

4. After the full database restore, verify the *ADABAS.INI* and *DBnnn.INI* files generated during the process because their values may reflect the settings at the time of the backup.

ENCRYPTION = keyword

This parameter specifies that the created database is encrypted and assigns the encryption algorithm. The keyword can take the values AES_256_XTS, AES_128_XTS and NO. Depending on the keyword specified, the ASSO and DATA container files are encrypted using XTS Advanced Encryption Standard with a key length of 256 bits (AES_256_XTS), a key length of 128 bits (AES_128_XTS), or they are not encrypted (NO).

The default value is NO.

Notes:

1. Database encryption cannot be disabled.
2. The `ENCRYPTION` parameter can only be used for a full database restore (overlay=* or restore=*) and the database must not exist. If the database exists, the encryption settings of the database are retained.
3. File restore/overlay operations will be applied according to the encryption settings of the existing database.

FMOVE [=(number [,number [-number]]...)]

If this keyword is specified, ADABCK reallocates all files to be restored or the specified subset rather than attempting to restore them in the same block ranges as in the backup. Using this keyword reduces the number of file extents as much as possible.

FORMAT = (keyword [,keyword])

The keywords ASSO and/or DATA may be specified. This parameter is used to format Associator and/or Data Storage blocks. When restoring at the file level, only blocks contained in the unused areas of the files' extents are formatted.

KMSTARGET = string

This parameter specifies the key management system that is used if the created database is encrypted. See *Adabas Security Features* > *Adabas Encryption for Linux* > *Key Management System* in the Adabas documentation for all supported key management systems.

The default value is FILE.

Note: You must specify the KMSTARGET parameter before the ENCRYPTION parameter.

Examples for KSMTARGET

Please see **Examples for RESTORE/OVERLAY using ENCRYPTION and KMSTARGET**.

NEW_DBID = number

This parameter can be used to change the identifier of the database to be restored. This parameter can only be specified when restoring a complete database.

A new identifier can be used to restore a backup copy of an active database into a different set of container files. The new identifier may not be identical to that of another active database.

If this parameter is omitted, the database identifier remains unchanged.

Important: If a backup of e.g. database 100 is restored into an existing database 100, but a NEW_DBID parameter with a different dbid is used, the database 100 will become inaccessible. When accessing the database the Adabas utilities will report a DBID mismatch because of a wrong dbid in the GCB RABN.

In such a case the following steps need to be performed to get access again to the database:

1. Rename the database directory from db100 to db101, change the container extensions from 100 to 101, and rename the DB100.INI file to DB101.INI.
2. Update DB101.INI: Edit the content by changing the path and file name of each environment variable value from 100 to 101.
3. Update assign.bsh: Edit the content by changing the path and file name of each environment variable value from 100 to 101.
4. Update ADABAS.INI: In the DB_LIST section edit all values from the old dbid value 100 to the new dbid val-ue 101.

REPLICATION

The parameter REPLICATION is relevant only for customers who are using the Adabas Event Replicator with Adabas - Adabas Replication.

This parameter should be specified if you want to use ADABCK for the Adabas - Adabas replication initial state processing. If you specify this parameter, the Adabas file is automatically marked as a replication target file.

For further information refer to ADAOPR CHANGE_REPLICATION_STATUS.

Examples for RESTORE/OVERLAY

Example 1

The complete database is restored in parallel from several backup devices. Only database backups can be processed (backups created with DUMP=* or EXU_DUMP=*). The backup of example 1 for DUMP/EXUDUMP is used. The nucleus must be inactive.

```
adabck db=34 parallel=multi_process restore=\*
%ADABCK-I-STARTED,      30-OCT-2015 11:13:24, <version number>
%ADABCK-I-DBOFF, database 34 accessed offline

Restore database 34 dumped on 30-OCT-2015 11:10:29

Database 34, GENERAL-DATABASE

File     1, CHECKPOINT-FILE , loaded on  4-SEP-2014 13:52:43
File     2, SECURITY-FILE   , loaded on  4-SEP-2014 13:52:43
File     3, USER-DATA-FILE  , loaded on  4-SEP-2014 13:52:43
File     4, Update-log      , loaded on 17-SEP-2014 14:44:19
File     9, EMPLOYEES       , loaded on  8-OCT-2008 17:59:40
File    14, miscellaneous   , loaded on 11-JUN-2015 13:22:19
File    17, Timezone        , loaded on 19-SEP-2014 11:44:42
File    19, LARGE           , loaded on  2-SEP-2014 15:37:18
File    30, FILE-30         , loaded on 31-MAR-2015 13:40:33
File    51, PCA24SYSF1      , loaded on 14-APR-2014 16:55:22
File    80, P295170         , loaded on  4-AUG-2015 12:42:43
File    91, ADAOS-2544      , loaded on  8-APR-2015 13:19:27
File    95, P299255         , loaded on 20-MAR-2014 11:35:30
File    98, ADAOS-4591      , loaded on 16-JUL-2015 10:03:16
File   101, COLLATION-TESTS , loaded on 14-APR-2014 16:47:30
File   111, TESTOPT         , loaded on 29-NOV-2011 10:34:23
File   113, lob_LB          , loaded on 23-JUN-2015 15:48:48
File   146, XMA-REPOSITORY  , loaded on 10-DEC-2014 09:39:52
File   215, ADAOS-4647      , loaded on 27-APR-2012 15:52:49
File  1009, LOBFILE of 9    , loaded on  8-OCT-2008 17:59:40
File  1080, LOBFILE of 80   , loaded on  4-AUG-2015 12:42:43
File  1113, LOBFILE of 113  , loaded on 23-JUN-2015 15:48:48
File 22111, LOBFILE of 111  , loaded on 29-NOV-2011 10:34:23
```

```
%ADABCK-I-IOCNT, 1 IOs on dataset WORK
%ADABCK-I-IOCNT, 133 IOs on dataset DATA
%ADABCK-I-IOCNT, 244 IOs on dataset ASSO
%ADABCK-I-IOCNT, 41 IOs on dataset BCK001
%ADABCK-I-IOCNT, 34 IOs on dataset BCK002
%ADABCK-I-IOCNT, 80 IOs on dataset BCK003
%ADABCK-I-TERMINATED,   30-OCT-2015 11:13:26, elapsed time: 00:00:02
```

Example 2

File 215 is restored; the file must not already exist in the database. Database backups and file backups can be processed. If there is more than one backup file, the backup files are not processed in parallel, even if the backup was created with option PARALLEL. The nucleus may be either active or inactive.

```
adabck db=34 restore=215
%ADABCK-I-STARTED,      30-OCT-2015 11:18:34, <version number>

%ADABCK-I-DBOFF, database 34 accessed offline

Restore files from database 34 dumped on 30-OCT-2015 11:10:29

Database 34, GENERAL-DATABASE

File   215, ADAOS-4647      , loaded on 27-APR-2012 15:52:49


%ADABCK-I-IOCNT, 7 IOs on dataset DATA
%ADABCK-I-IOCNT, 28 IOs on dataset ASSO
%ADABCK-I-IOCNT, 41 IOs on dataset BCK001
%ADABCK-I-IOCNT, 34 IOs on dataset BCK002
%ADABCK-I-IOCNT, 80 IOs on dataset BCK003
%ADABCK-I-TERMINATED,   30-OCT-2015 11:18:34, elapsed time: 00:00:00
```

Example 3

All base files in a backup file with a file number between 91 and 99 or equal to 51 or between 11 and 19 are restored (including the corresponding LOB files, even if they are not in the specified file ranges). If a file already exists in the database, the file is overwritten. Database backups and file backups can be processed. The nucleus may be either active or inactive.

```
adabck db=34 over=\(91-99,51,11-19\)
%ADABCK-I-STARTED,      30-OCT-2015 11:38:12, <version number>
%ADABCK-I-DBON, database 34 accessed online

Overlay files dumped on 30-OCT-2015 10:51:14

Database 34, GENERAL-DATABASE

File    14, miscellaneous   , loaded on 11-JUN-2015 13:22:19
File    17, Timezone        , loaded on 19-SEP-2014 11:44:42
File    19, LARGE           , loaded on  2-SEP-2014 15:37:18
File    51, PCA24SYSF1      , loaded on 14-APR-2014 16:55:22
File    91, ADAOS-2544      , loaded on  8-APR-2015 13:19:27
File    95, P299255         , loaded on 20-MAR-2014 11:35:30
File    98, ADAOS-4591      , loaded on 16-JUL-2015 10:03:16


%ADABCK-I-IOCNT, 619 IOs on dataset DATA
%ADABCK-I-IOCNT, 1122 IOs on dataset ASSO
%ADABCK-I-IOCNT, 1195 IOs on dataset BCK001
%ADABCK-I-TERMINATED,   30-OCT-2015 11:38:13, elapsed time: 00:00:01
```

Example 4

Base file in a backup file with file number 98 will be restored as file 198. If that file already exists in the database, the file is overwritten. Database backups and file backups can be processed. The nucleus may be either active or inactive.

```
adabck db=34 overlay=98 renumber=198
%ADABCK-I-STARTED, 30-OCT-2015 11:40:12, <version number>
%ADABCK-I-DBON, database 34 accessed online

Overlay files dumped on 30-OCT-2015 10:51:14

Database 34, GENERAL-DATABASE

File 98 renumbered to file 198
File 198 loaded on 16-JUL-2015 10:03:16

%ADABCK-I-IOCNT, 619 IOs on dataset DATA
%ADABCK-I-IOCNT, 1122 IOs on dataset ASSO
%ADABCK-I-IOCNT, 1195 IOs on dataset BCK001
%ADABCK-I-TERMINATED, 30-OCT-2015 11:40:13, elapsed time: 00:00:01
```

Examples for RESTORE/OVERLAY using ENCRYPTION and KMSTARGET

Example 1

The complete database is to be restored as an encrypted database from an Adabas backup copy of a non-encrypted database. The ASSO and DATA container files are encrypted with algorithm AES_256_XTS. The encryption keys are created and managed by the Adabas file-based key management system (default: KMSTARGET=FILE).

```
adabck dbid=1 restore=* encryption=aes_256_xts ↩
```

Example 2

The complete database is to be overlayed as an encrypted database from an Adabas backup copy of a non-encrypted database. The ASSO and DATA container files are encrypted with algorithm AES_128_XTS. The encryption keys are created and managed by the Adabas file-based key management system.

```
adabck dbid=1 overlay=* kmstarget=file encryption=aes_128_xts
```

Example 3

The complete database is to be restored as a non-encrypted database from an Adabas backup copy of an encrypted database. The Adabas container files are not encrypted.

```
adabck dbid=1 restore=* encryption=no
```

SUMMARY

```
SUMMARY
```

This parameter displays general information and physical layout of the database in the Adabas backup copy created by a previous run of the DUMP/EXU_DUMP function.

Example

```
adabck summary
%ADABCK-I-STARTED,      30-AUG-2020 12:01:46, <version number>

Database dumped on 30-AUG-2020 11:57:04

Database 34, GENERAL-DATABASE

Summary of Database 34          30-AUG-2020 12:01:46


DATABASE NAME                     GENERAL-DATABASE
```

```
DATABASE ID                          34
MAXIMUM FILE NUMBER LOADED       22111
SYSTEM FILES                         1 (CHK),     2 (SEC),     3 (USR)
                                   150 (RBAC)
ACTUAL FILES LOADED                  23
CURRENT PLOG NUMBER                  99
CURRENT CLOG NUMBER                  10
SECURITY                         ACTIVE
ENCRYPTION                       AES_256_XTS
KMSTARGET                        FILE
KEKNAME                          ↩
1598876857F83BB64A01916FFA3ACF9D39DB18537E80E3E48FD8A2333A3D77C8

Container  Device      Extents in Blocks    Number of      Block    Total Size
  File     Type         from          to     Blocks        Size     (Megabytes)
-------------------------------------------------------------------------------

 ASS01     file            1      35,840       35,840      4,096         140.00
 ASS02     file       35,841     166,400      130,560      2,048         255.00
 ASS03     file      166,401     205,120       38,720     32,768       1,210.00
 ASS04     file      205,121     210,240        5,120     16,384          80.00
 ASS05     file      210,241     944,832      734,592      8,192       5,739.00

 DATA1     file            1       8,891        8,891      4,096          34.73
 DATA2     file        8,892      24,379       15,488      8,192         121.00
 DATA3     file       24,380      34,619       10,240     16,384         160.00
 DATA4     file       34,620     388,763      354,144     32,768      11,067.00

 WORK1     file            1     207,872      207,872      4,096         812.00

-------------------------------------------------------------------------------
                                                                      19,618.73
                                                                      =========


%ADABCK-I-IOCNT, 2 IOs on dataset BCK001
%ADABCK-I-TERMINATED,   30-AUG-2020 12:01:46, elapsed time: 00:00:00
```

The security information is only displayed if database security has been activated. Otherwise, the information is not displayed.

The encryption information about the encryption algorithm, KMS target (Key Management System) and KEK name (Key Encryption Key) is only displayed if the database is encrypted. Otherwise, the information is not displayed.

The RBAC system file is only displayed if it has been defined. Otherwise, the information is not displayed.

Restart Considerations

ADABCK has no restart capability. An abnormally-terminated ADABCK execution must be rerun from the beginning.

An interrupted RESTORE/OVERLAY of one or more files will result in lost RABNs which can be recovered by executing the RECOVER function of the utility ADADBM. An interrupted RESTORE/OVERLAY of a database results in a database that cannot be accessed.

5 ADACLP (Command Log Report)

- Functional Overview 42
- Procedure Flow 43
- Checkpoints 43
- Control Parameters 44
- Specifying Multiple Selection Criteria 50

This chapter describes the utility "ADACLP".

Functional Overview

The ADACLP utility prints the command log with a line width of 132 characters.

Note: ADACLP can only process command logs of nucleus sessions that were started with the ADANUC parameter CLOGLAYOUT=5 (5 is the default value). Please refer to *ADANUC, CLOGLAYOUT* in the documentation of Adabas for further information.

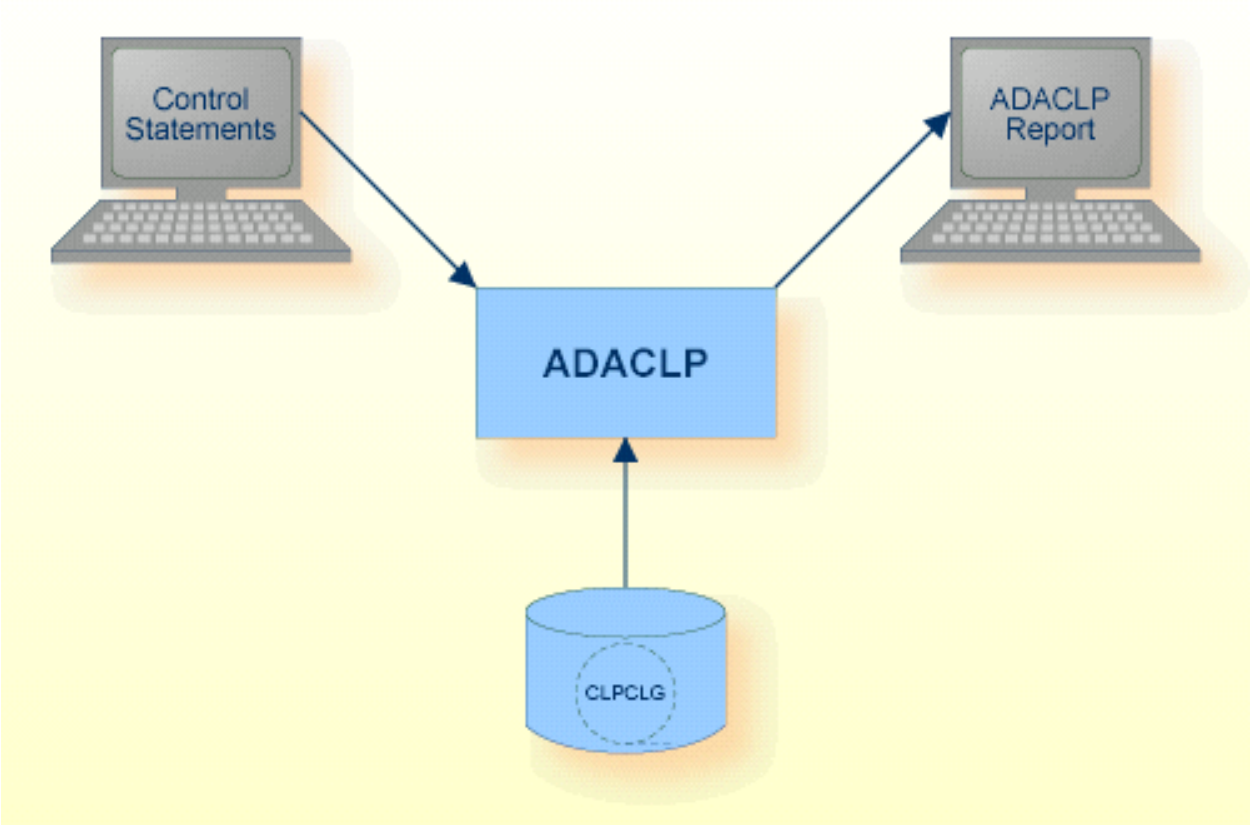
A record is written in the command log for each Adabas command issued. Command logging must be enabled during Adabas startup with the nucleus parameter LOGGING, or when the nucleus is already active with the ADAOPR parameter LOGGING.

Note: For performance reasons, the Adabas nucleus determines the command start timestamp only if command logging has been enabled. For this reason, the command start date and the command duration are not displayed for Adabas commands that are already active but not yet finished when command logging is switched on.

Any of the ADACLP parameters selects a subset of the command log information.

This utility is a single-function utility. For more information about single- and multi-function utilities, see *Adabas Basics*, *Using Utilities* in the Adabas documentation.

Procedure Flow



Data Set	Environment Variable / Logical Name	Storage Medium	Additional Information
Command log	CLPCLG	Disk (* see note)	Utilities Manual, ADACLP
Control statements	stdin		Utilities Manual
ADACLP report	stdout		Messages and Codes

Checkpoints

The utility writes no checkpoints.

Control Parameters

The following control parameters are available:

```
D    [NO]ADDITIONS_2

     CLASS = (keyword[,keyword]...)

     CLOG = (number[,number])

     COMMAND = (keyword[,keyword]...)

     DATE = ([absolute-date][,[absolute-date]])

     DBID  = number

D    DISPLAY = (keyword[,keyword]...)

D    ES_ID [=number]

D    FILE = (number[-number][,number[-number]]...)

D    [NO]HEXADECIMAL

D    LOGIN_ID = string

     NODE_ID = string

D    PAGE = number

D    RECORDS = number[-number]

     RESPONSE = (number[-number][,number[-number]]...)

D    USER_ID = string

D    WIDTH = number
```

[NO]ADDITIONS_2

```
[NO]ADDITIONS_2
```

This option can be used to display the Additions 2 field instead of the command ID.

The default is NOADDITIONS_2.

CLASS

```
CLASS = (keyword [,keyword]...)
```

This parameter selects the log records whose command codes belong to the specified command class. All records are selected if neither the CLASS parameter nor the COMMAND parameter is specified.

CLASS and COMMAND are mutually exclusive.

The following keywords can be used:

KEYWORD	USE
CONTROL	Selects control commands such as 'open' and 'close';
FIND	Selects find commands;
READ	Selects read commands;
UPDATE	Selects update commands.

Example:

```
adaclp: class = find
```

The log records of the commands S1, S2, S4, S8 and S9 are selected.

CLOG

```
CLOG = (number [,number])
```

It is optional if the command log is within a file system. The CLOG number and the extension count can be specified. If no extension count is specified, Adabas will open subsequent extents as necessary. If an extent count is specified, then only the specified extent will be processed.

Note: This parameter applies to Linux platforms only.

COMMAND

```
COMMAND = (keyword [,keyword]...)
```

This parameter selects the log records with an Adabas command code specified by the keywords. Up to ten keywords can be defined. If neither the COMMAND parameter nor the CLASS parameter is specified, all records are selected.

COMMAND and CLASS are mutually exclusive.

All valid Adabas commands (A1...S9) can be used as keywords (see the *Comman Reference* of the Adabas documentation for further information).

DATE

```
DATE = ([absolute-date] [,[absolute-date]])
```

This parameter selects the log records in the range specified by the optional date strings. The date strings must correspond to the following absolute date and time format:

Leading zeroes in the date and time specification may be omitted. Any numbers not specified are set to 0, for example 28-jul-2012 is equivalent to 28-jul-2012:00:00:00.

By default, all log records are selected.

Examples:

```
adaclp: date = 8-aug-2012
```

The log record written on 8-AUG-2012 00:00:00 is selected

```
adaclp: date = (8-aug-2012:12,)
```

All log records written from 8-AUG-2012 12:00:00 onwards are selected.

```
adaclp: date = (,8-aug-2012:12:34)
```

All log records written up to 8-AUG-2012 12:34:00 are selected.

DBID

```
DBID = number
```

This parameter selects the database to be used.

Note: This parameter applies to Linux platforms only.

DISPLAY

```
DISPLAY = (keyword [,keyword]...)
```

This parameter is used to display various kinds of information from the command log. The keywords shown in the following table are available. Information for these keywords can only be displayed if corresponding data was logged during the nucleus session.

KEYWORD	MEANING
CB	■ Command log record number ■ Starting date and time of the command ■ Duration of the command in microseconds ■ User ID specified in the corresponding OP command ■ Node ID ■ Login ID or, if ES_ID was specified, environment-specific ID ■ Selected fields of the control block ■ '*' in column 'X' indicates utility or exclusive file usage ■ The thread which processed the command ■ I/O statistics **Note:** For command logs created with versions lower than Version 6.3 SP1, the duration of the command is displayed in milliseconds.
FB	Format buffer.
FULL_CB	All fields of the control block. Other information shown for DISPLAY=CB is not shown here.
IB	ISN buffer.
IO	IO list.
NAT	Natural information. The following information is displayed in the output: NAT_APPL, NAT_PROG, NAT_UID, NAT_STMT, NAT_LVL, NAT_CNT, NAT_EXEC, NAT_LIB, NAT_RPCCLUID, NAT_RPCID, NAT_RPCCO, NAT_GRP. **Note:** This option requires additional configuration in the NATPARM module. For more information, see the appropriate Natural documentation.
RB	Record buffer.

KEYWORD	MEANING
SB	Search buffer.
STATISTICS	Command statistics of the selected records.
VB	Value buffer.

The default is DISPLAY = CB.

ES_ID

```
ES_ID [ = number]
```

This parameter causes the environment-specific ID to be displayed instead of the login ID.

If a number is specified, only records with information for the specified environment-specific ID (process ID) will be selected.

By default, all records are selected.

FILE

```
FILE = (number [- number] [,number [- number]]...)
```

This parameter selects the log records with commands that reference the file(s) specified by number or range of numbers. A maximum of 20 files may be specified.

By default, all records are selected.

[NO]HEXADECIMAL

```
[NO]HEXADECIMAL
```

If this parameter is set to HEXADECIMAL, the record buffer and value buffer are displayed in hexadecimal format (when DISPLAY=RB or DISPLAY=VB is specified).

The default is NOHEXADECIMAL.

LOGIN_ID

```
LOGIN_ID = string
```

This parameter selects all records with the specified login ID.

By default, all records are selected.

NODE_ID

```
NODE_ID = string
```

This parameter selects the log records from the specified node.

The node identification shown while processing ADAOPR with the parameter DISPLAY = UQ must be used.

This parameter is valid only if ENTIRE NET-WORK is installed.

PAGE

```
PAGE = number
```

This parameter defines the page size, in lines, used for the printout.

The default is 59 lines.

RECORDS

```
RECORDS = number [-number]
```

This parameter selects the log records in the specified range of log record numbers. Log record numbers start with 1 after the log is switched on.

By default, all records are selected.

RESPONSE

```
RESPONSE = (number [- number] [,number [- number]] ... )
```

This parameter selects the records with the specified response code or range of response codes.

USER_ID

```
USER_ID = string
```

This parameter selects the records with the user ID specified in 'string'.

By default, all users are selected.

Example

```
user_id = *adarep
```

All records that represent commands issued from the utility ADAREP are selected.

WIDTH

```
WIDTH = number
```

This parameter selects the output line width. Valid values are 80 and 132.

The default is 132.

Specifying Multiple Selection Criteria

If multiple selection criteria are specified, they are combined by a logical AND, e.g.

```
command = l3, file = 5
```

This selects all L3 commands on file 5.

6 ADACMP (Compression Of Data)

- Functional Overview 52
- Procedure Flow 53
- Checkpoints 54
- Control Parameters 54
- Output 66
- Report 67
- Restart Considerations 67

This chapter describes the utility "ADACMP".

Functional Overview

The compression utility ADACMP compresses user raw data into a form which can be used by the mass update utility ADAMUP.

The input data for this utility must be contained in a sequential file. LOB field values can also be provided in separate files.

The logical structure and characteristics of the input data are described by a field definition table (FDT). These statements specify the level number, field name, standard length and format together with any definition options that are to be assigned to the field (descriptor, unique descriptor, multiple-value field, null value suppression, fixed storage, periodic group).

Each field in the input record without the option SY (system generated) is compressed. Compression consists of removing trailing blanks from alphanumeric fields and leading zeros from numeric fields. Unpacked and packed fields are checked for correct data. Fields defined with the fixed storage option are not compressed. A user exit is provided to allow additional editing of each input record with a user-written routine.

System generated fields are either regenerated or decompressed, depending on the keyword specified for the ADACMP parameter SYFINPUT.

This utility creates three types of output files:

- Compressed data.
- Descriptor values.
- Records with errors.

The sizes of the descriptor values of all descriptors are listed at the end of execution.

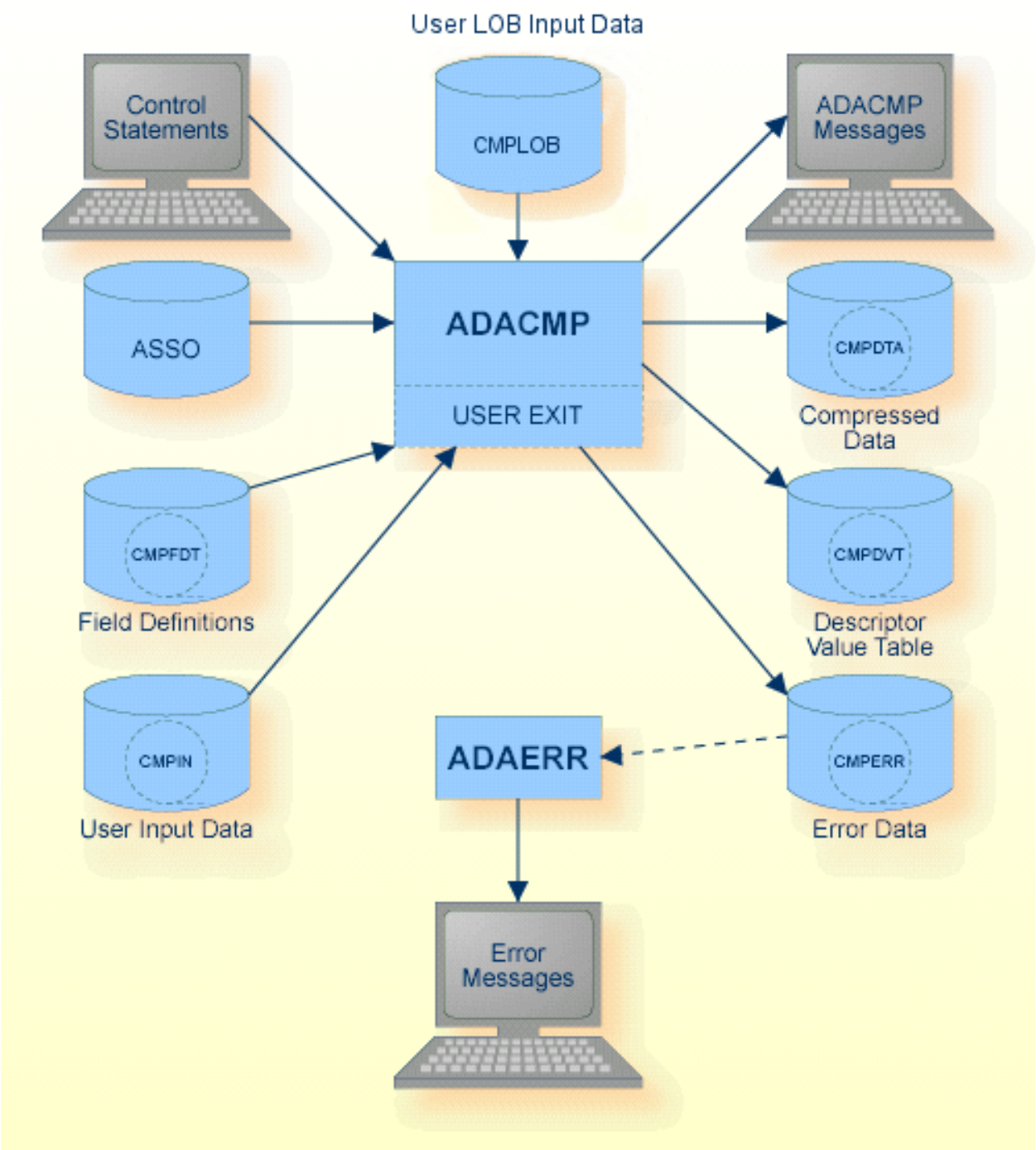
If the utility writes records to the error file, it will exit with a non-zero status.

Note: Please be careful if you want to add data to a file that still contains ICU 3.2 collation descriptors:

- If you specify the FDT with the parameters DBID and FILE, the FDT is taken unchanged from the database. This means the ICU version is still 3.2. You can add the data to the file, and the ICU version remains 3.2.
- If you take the FDT from the CMPFDT file, a new FDT is created from the CMPFDT file, where the ICU version is set to 5.4. This means you can only add the data to the file if it is empty, and if you specify the NEW_FDT option. The ICU version used is 5.4.

This utility is a single-function utility. For more information about single- and multi-function utilities, see *Adabas Basics, Using Utilities* in the Adabas documentation.

Procedure Flow



The sequential files CMPDTA, CMPDVT and CMPERR can have multiple extents. . CMPLOB is a directory that contains files which may be stored as LOB values in the database.

Data Set	Environment Variable / Logical Name	Storage Medium	Additional Information
Associator	ASSOx	Disk	
Compressed data	CMPDTA	Disk (* see note)	output by ADACMP
Descriptor Value Table	CMPDVT	Disk (* see note)	output by ADACMP
Rejected data	CMPERR	Disk (* see note)	output by ADACMP
Input data FDT	CMPFDT	Disk (* see note)	Utilities Manual
User input data	CMPIN	Disk (* see note)	Utilities Manual
User LOB input data	CMPLOB	Disk	Utilities Manual
ADACMP control statements	stdin		Utilities Manual
ADACMP messages	stdout		Messages and Codes

Note: (*) A named pipe can be used for this sequential file.

If the SINGLE_FILE option is set, the Descriptor Value Table (DVT) and the compressed user data are written together to the logical name CMPDTA.

Checkpoints

The utility writes no checkpoints.

Control Parameters

The following control parameters are available:

```
     DBID = number

D    [NO]DST

     FDT

     FIELDS {uncompressed_field_definition | FDT}...[END_OF_FIELDS | . ]

     FILE = number

D    [NO]LOBS

D    [NO]LOWER_CASE_FIELD_NAME
```

```
D    MAX_DECOMPRESSED_SIZE = number [K|M]

D    MUPE_C_L = {1|2|4}

D    [NO]NULL_VALUE

D    NUMREC = number

D    RECORD_STRUCTURE = keyword

     SEPARATOR = character | \character

D    [NO]SHORT_RECORDS

D    [NO]SINGLE_FILE

     SKIPREC = number

D    SOURCE_ARCHITECTURE = (keyword[,keyword][,keyword])

D    SYFINPUT = keyword

D    TZ {=|:} [timezone]

D    [NO]USEREXIT

D    [NO]USERISN

D    WCHARSET = char_set
```

DBID

```
DBID = number
```

This parameter selects the database that contains the file to be specified by the FILE parameter.

[NO]DST

```
[NO]DST
```

The parameter DST is required if a daylight saving time indicator is provided for date/time fields with the option TZ. The daylight saving time indicator must be appended behind the date/time value as a 2-byte integer value (format F) that contains the number of seconds to be added to the standard time in order to get the actual time (usually 0 or 3600).

Without the parameter DST, it is not possible to define time values in the hour before the time is switched back to standard time.

The default is NODST.

Notes:

1. The DST parameter is ignored if the FIELDS parameter is specified. In this case, you must specify a D element for fields with the daylight saving time indicator.
2. The DST parameter is not compatible with the RECORD_STRUCTURE = NEWLINE_SEPARATOR parameter because the daylight saving indicator in format F contains non-printable characters.

Example:

A DT field has the following definition: 1,DT,8,P,DT=E(DATE_TIME),TZ

The following values must then be specified for this field:

- The local date/time value corresponding to the edit mask DATE_TIME as an 8-byte packed value
- The daylight saving time indicator, usually 0 for standard time and 3600 for summer time as a 2-byte fixed point value

```
Case 1 (DT has a date/time value with daylight saving time): 0x0200910250230000E10
Case 2 (DT has a date/time value with standard time): 0x0200910250230000000
```

FDT

```
FDT
```

If this parameter is specified as the first parameter, or as the second parameter after [NO]LOWER_CASE_FIELD_NAMES, ADACMP reads the FDT information contained in the sequential file CMPFDT and displays the FDT.

Note: Alternatively, instead of FDT, you can specify DBID and FILE as the first parameters, or as the second parameters after [NO]LOWER_CASE_FIELD_NAMES (which is allowed before DBID and FILE). In this case, the FDT of the file is used as the base for the compression.

The FDT parameter can be specified several times, but if you have already determined the FDT to be used for the compression by specifying the FDT or DBID and FILE parameters, specifying the FDT parameter again will only display the FDT; the FDT is not overwritten by the CMPFDT file.

When handling standalone output from ADADCU, the corresponding DCUOUT and DCUFDT files are not encrypted and do not contain encryption information. In order to make up for that, ADACMP can accept a dbid. For example, the command "ADACMP dbid=100 fdt" will compress the data in DCUOUT with the FDT from DCUFDT. Additionally, ADACMP will read the encryption information stored in ASSO1 of database 100 and encrypt the output files if database 100 is encrypted.

FIELDS

```
FIELDS {uncompressed_field_definition | FDT}...[END_OF_FIELDS | . ]
```

This parameter is used to specify a subset of fields given in the FDT and their format and length. This means that the input records do not have to contain all of the fields given in the FDT, or that fields can be provided with a different format or length. The syntax and semantics are the same as for the format buffer, with the exception that you can also specify an R-element (for LOB references) if the decompressed record contains the name of a file containing the LOB value instead of the LOB value itself.

While entering the specification list, the FDT function can be used to display the FDT of the file to be decompressed. The specification list can be terminated or interrupted by entering END_OF_FIELDS or '.'. The '.' option is an implicit END_OF_FIELDS and is compatible with the format buffer syntax. FIELDS or END_OF_FIELDS must always be entered on a line by itself, whereas the '.' may be entered on a line by itself or at the end of the format buffer elements.

If the field definitions are terminated with the END_OF_FIELDS parameter, this parameter must be specified in upper case when the LOWER_CASE_FIELD_NAMES parameter is used. In addition, the FDT parameter must also be specified in upper case when the LOWER_CASE_FIELD_NAMES parameter is used.

FILE

```
FILE = number
```

This parameter specifies the file from which the FDT information is to be read. This parameter can only be specified after the DBID parameter.

Note: If you specified $CMPFDT, the FDT supplied through $CMPFDT and the FDT held in the database must have the same structure. Otherwise, the execution aborts when a discrepancy is detected. ADACMP stops on the first difference of the FDT field. If $CMPFDT is empty or if it points to a nonexistent file, the comparison is skipped.

[NO]LOWER_CASE_FIELD_NAMES

```
[NO]LOWER_CASE_FIELD_NAMES
```

If LOWER_CASE_FIELD_NAMES is specified, Adabas field names are not converted to upper case. If NOLOWER_CASE_FIELD_NAMES is specified, Adabas field names are converted to upper case. The default is NOLOWER_CASE_FIELD_NAMES.

If lower case field names in the FDT are not to be converted to upper case, the parameter must be specified as the first parameter before the FDT parameter; if lower case field names in the FIELDS parameter are not to be converted to upper case, the parameter must be specified before the FIELDS parameter.

Caution: If the LOWER_CASE_FIELD_NAMES parameter is specified for the CMPFDT file, not upper case conversion is done for the complete file. Lower case characters for field formats and field options will cause FDT syntax errors. This problem also exists for lower case characters in the FIELDS parameter.

[NO]LOBS

```
[NO]LOBS
```

This parameter specifies whether LA and LB field values are to be stored in a LOB file after loading the compressed data into the database:

- If the parameters DBID and file number have been specified, this parameter is ignored, and the field is handled as described below;
- If the parameters DBID and file number have not been specified and LOBS is specified, field values for LA and LB fields are prepared for storage in a LOB file, except the field is defined as a descriptor.
- If the parameters DBID and file number have not been specified and NOLOBS is specified, field values for LA and LB fields are prepared for storage in the base file. In this case, the length of field values for LA and LB fields must not exceed 16381 bytes and the compressed record must fit into a 32 KB DATA block.

Please note that LA and LB fields which are descriptors or parent fields of a derived descriptor, e.g. a super descriptor, are always handled as described for the NOLOBS parameter.

Default behavior is as follows:

- If the parameters DBID and file number have been specified and the file is a base file with corresponding LOB file, LOBS is default.
- If the parameters DBID and file number have been specified and the file is not a base file with corresponding LOB file, NOLOBS is default.
- If the parameters DBID and file number have not been specified, LOBS is default.

MAX_DECOMPRESSED_SIZE

```
MAX_DECOMPRESSED_SIZE = number  [K|M]
```

This parameter specifies the maximum size of a decompressed record in bytes, kilobytes or megabytes, depending on the specification of "K" or "M" after the number. This parameter is intended to recognize invalid CMPIN files as early as possible.

The default is 65536. This is also the minimum value.

Notes:

1. This parameter does not include the size of LOB values stored in separate files.
2. The exact definition of this parameter is the size of the I/O buffer required for the largest decompressed record. Only multiples of 256 bytes are used for the I/O buffers, which means that you must specify a value greater than or equal to the largest decompressed record (including the preceding length field) rounded up to the next multiple of 256.

MUPE_C_L

```
MUPE_C_L = {1|2|4}
```

If the uncompressed data contain multiple-value fields or periodic groups, they are preceded by a binary count field with the length of MUPE_C_L bytes.

The default is 1.

[NO]NULL_VALUE

```
[NO]NULL_VALUE
```

The parameter NULL_VALUE is required if you are compressing data according to the standard FDT and the status values of the NC option fields are given in the input data. Normally, such input data is generated by ADADCU with the NULL_VALUE option set.

The default is NONULL_VALUE.

Example

The definition in the FDT for the field AA is: 1, AA, 2, A, NC

Case 1 (AA has a non-NULL value): input record (hexadecimal) = 00004142

Case 2 (AA has a NULL value): input record (hexadecimal) = FFFF2020

NUMREC

```
NUMREC = number
```

This parameter specifies the number of input records to be processed. If this parameter is omitted, all input records contained on the input file are processed.

Use of this parameter is recommended for the initial execution of ADACMP if the input data file contains a large number of records. This avoids unnecessary processing of all records in cases where a data definition error or invalid input data results in a large number of rejected records.

This parameter is also useful for creating small files for test purposes.

RECORD_STRUCTURE

```
RECORD_STRUCTURE = keyword
```

This parameter specifies the type of record separation used in the input file with the environment variable CMPIN. The following keywords can be used:

Keyword	Meaning
ELENGTH_PREFIX	The records in the CMPIN file are separated by a two-byte exclusive length field.
E4LENGTH_PREFIX	The records in the decompressed data file are separated by a 4-byte exclusive length field.
ILENGTH_PREFIX	The records in the CMPIN file are separated by a two-byte inclusive length field.
I4LENGTH_PREFIX	The records in the decompressed data file are separated by a 4-byte inclusive length field.
NEWLINE_SEPARATOR	The records in the CMPIN file are separated by a new-line character. This keyword may only be specified if the field values do not contain characters interpreted as new-line (i.e. if there are only unpacked, alphanumeric and Unicode fields, and the alphanumeric and Unicode fields contain only printable characters). This keyword and the USERISN parameter are mutually exclusive.

RDW	The records in the CMPIN file contain data that has been transferred from an IBM host using the FTP *site rdw* option. ADACMP is able to process such data without having to use *cvt_fmt* first (in previous versions, the unsupported tool *cvt_fmt* was used for such format conversions). For example:
	
	`% ftp IBM-host`
	`ftp> binary`
	`200 Representation type is Image`
	`ftp> site rdw`
	`200 Site command was accepted`
	`ftp> get decomp`
	`% setenv CMPIN decomp`
	`% adacmp fdt record_structure=rdw source=(ebcdic,high)`

RDW_HEADER	Like RDW, for data decompressed on a mainframe with HEADER=YES.
HEADER	For data decompressed on a mainframe with HEADER=YES, if the decompressed data do not contain any additional information about block or record length.
VARIABLE_BLOCKED	The variable blocked format from BS2000 or IBM.

The default is ELENGTH_PREFIX.

SEPARATOR

```
SEPARATOR = character | \character
```

If you specify this option, ADACMP expects the fields in the raw data record to be separated by the character specified. You can omit the apostrophes round the character specification if the character has no special meaning for the Adabas utilities. The same fields in different records are then permitted to be of different lengths.

If a format buffer is specified using the FIELDS parameter, the order of the specified field names must correspond with the order in which the fields are specified in the FDT. A mismatch results if this is not the case.

If the FDT contains multiple value fields or periodic groups, a format buffer must be specified with the FIELDS parameter. Members of periodic groups must be ordered by 1) periodic group index and 2) field sequence in the FDT (see example 2 below).

Because no binary data is expected in the input file using the SEPARATOR option, the RECORD_STRUCTURE parameter will be set to NEWLINE_SEPARATOR.

Example 1

```
FDT:      1, AA, 2, U
          1, AB, 8, U
          1, AC, 2, A

CMPIN:    12;12345678;AA
          1234;5;BB

adacmp
fdt
separator=\;

or for Linux

adacmp fdt  separator=\\\;

or

adacmp fdt  separator='\;'
```

In this example, 2 records are compressed with the default FDT, the separator character is the semicolon, and the default record structure is NEWLINE_SEPARATOR. Note that the semicolon must be preceded by a backslash, otherwise it would be treated as the start of a comment. If you enter the parameters under Linux directly from the command line, it is necessary to precede the backslash and the semicolon by additional backslashes or to put them in quotes or double quotes since they are special characters.

Example 2

```
FDT:      1, XX, PE
          2, AA, 8, A
          2, AB, 8, U
          1, YY, 2, A

Correct:  CMPIN:    aaaa,1,bbbb,2,yy

          Command:  adacmp fdt separator=, fields AA1,AB1,AA2,AB2,YY.
          First, the field values for the periodic group index 1 are
          specified, and then the field values for periodic group index 2.

Invalid:  CMPIN:   aaaa,bbbb,1,2,yy
          Command: adacmp fdt separator=, fields AA1-2,AB1-2,YY.
          The fields specification is invalid because the 2nd value of
          AA is specified before the 1st value of AB; you will get
          the error SEPINV.
```

In this example, 1 record with fields given in the format buffer is compressed, the separator character is the comma.

Example 3

```
FDT:      1, AA, 8, A
          1, MA, 1, A, MU

CMPIN:    aaaa%2%A%B
          bbbb%3%C%D%E

adacmp  dbid=9  file=15  separator=%, fields "AA,MAC,1,U,MA1-N"
```

In this example, 2 records with fields given in the format buffer are compressed, the occurrence count or the multiple value field MA is different in different records. The separator character is the percent character.

[NO]SHORT_RECORDS

```
[NO]SHORT_RECORDS
```

If SHORT_RECORDS is specified, it is possible to omit fields at the end of the decompressed record that contain null values.

The default is NOSHORT_RECORDS.

You can only omit complete fields; it is not possible to truncate the last value:

Example

Assuming you have specified the parameters for a file containing alphanumeric fields AA and AB:

```
FIELDS
AA,20,AB,20
END_OF_FIELDS
SHORT_RECORDS
```

Then the following record is allowed:

```
"Field AA          "
```

The following record is not allowed:

```
"Field AA"
```

[NO]SINGLE_FILE

```
[NO]SINGLE_FILE
```

If the SINGLE_FILE option is set, ADACMP writes the Descriptor Value Table (DVT) and the compressed user data to a single file (CMPDTA) instead of writing them to separate files.

The default is NOSINGLE_FILE.

SKIPREC

```
SKIPREC = number
```

This parameter specifies the number of records to be skipped before compression is started.

SOURCE_ARCHITECTURE

```
SOURCE_ARCHITECTURE = ( keyword [,keyword [,keyword] ] )
```

This parameter specifies the format (character set, floating-point format and byte order) of the input data records. The following keywords can be used:

Keyword Group	Valid Keywords
Character set	ASCII EBCDIC
Floating-point format	IBM_370_FLOATING IEEE_FLOATING

Keyword Group	Valid Keywords
	VAX_FLOATING
Byte order	HIGH_ORDER_BYTE_FIRST LOW_ORDER_BYTE_FIRST

If no keyword of a keyword group is specified, the default for this keyword group is the keyword that corresponds to the architecture of the machine on which ADACMP is running.

Note: The FDT is always input in ASCII format.

Example

If the input records that are to be compressed are in IBM format, the user must specify the following:

```
SOURCE_ARCHITECTURE = (EBCDIC, IBM_370_FLOATING, HIGH_ORDER_BYTE_FIRST)
```

SYFINPUT

```
SYFINPUT = keyword
```

This parameter specifies the input used for the compression of system generated fields. The following keywords can be used:

Keyword	Meaning
SYSTEM	The system generated field values are regenerated by the system in ADACMP.
USER	The system generated field values are taken from the decompressed file.

The default is SYFINPUT = USER.

TZ

```
TZ  {=|:} [timezone]
```

The specified time zone must be a valid time zone name that is contained in the time zone database known as the Olson database (**https://www.iana.org/time-zones**). If a time zone has been specified, this time zone is used for time zone conversions of date/time fields with the option TZ.

The default is UTC, which is used internally to store date/time fields with option TZ; no conversion is required.

If you specify an empty value, no checks are made to ensure that date/time fields are correct.

Note: The time zone names are file names. Depending on the platform, these file names may or may not be case sensitive. Also, the time zone names, depending on the platform, may or may not be case sensitive.

Examples:

```
tz:Europe/Berlin
```

This is correct on all platforms.

```
TZ=Europe/Berlin
```

With this specification, TZ is converted to upper case EUROPE/BERLIN. This is correct on Windows, because file names are not case sensitive on Windows, but it is not correct on Linux, because Linux file names are case sensitive.

[NO]USEREXIT

```
[NO]USEREXIT
```

This option specifies whether a user exit is to be taken or not. If USEREXIT is specified, the environment variable ADAUEX_6/logical name ADABAS$USEREXIT_6 must point to a loadable user-written routine.

The default is NOUSEREXIT.

[NO]USERISN

```
[NO]USERISN
```

If this option is set to USERISN, the ISN for each record in the input file will be assigned by the user.

If USERISN is specified, the user must give the ISN to be assigned to each record as a four-byte binary number immediately preceding each data record.

ISNs may be assigned in any order and must be unique (for the file). The ISN must not exceed the maximum number of records (MAXISN) specified for the file.

ADACMP does not check for unique ISNs or for ISNs which exceed MAXISN. These checks are performed by the mass update utility ADAMUP (if an error is detected, the ADAMUP run terminates with an error message).

If this option is set to NOUSERISN, the ISN is assigned by Adabas.

The default is NOUSERISN.

WCHARSET

```
WCHARSET = char_set
```

This parameter specifies the default encoding used in the decompressed file based on the encoding names listed at **https://www.iana.org/assignments/character-sets** - most of the character sets listed there are supported by ICU, which is used by Adabas for internationalization support.

The default is UTF-8.

Output

The ADACMP utility outputs three files:

1. Compressed data
2. Descriptor values
3. Records with errors

Compressed Data Records

The data records which ADACMP has processed, modified and compressed are output together with the FDT information to a sequential file. This file is used as input for the mass update utility ADAMUP.

If the output file contains no records (no records on the input file or all records rejected), the output may still be used as input for the mass update utility ADAMUP.

Descriptor-Value Table File

This file contains the descriptor value tables (DVT).

Compressed data records and descriptor value tables are written to one file if the SINGLE_FILE option is specified.

Rejected Data Records

Any records rejected by ADACMP are written to the ADACMP error file. The contents of this error file should be displayed using the ADAERR utility. Do not print the error file using the standard operating system print utilities since the records contain unprintable characters.

Report

The ADACMP report begins with a display of the field definition entered if CMPFDT is used for input. Any statement which contains a syntax error will be printed with a message immediately following the statement.

Following the display of the data-definition statements, a descriptor summary, the number of input records processed, the number of input records rejected, and the number of input records compressed are printed.

Restart Considerations

ADACMP does not have a restart capability. An interrupted ADACMP run must be re-started from the beginning.

ADACMP does not change the database; therefore, no considerations need to be made concerning database status before restarting ADACMP.

7 ADADBM (Database Modification)

- Functional Overview 70
- Procedure Flow 72
- Checkpoints 74
- Control Parameters 75
- Restart Considerations 98
- RBAC Protection 99

This chapter describes the utility "ADADBM".

Functional Overview

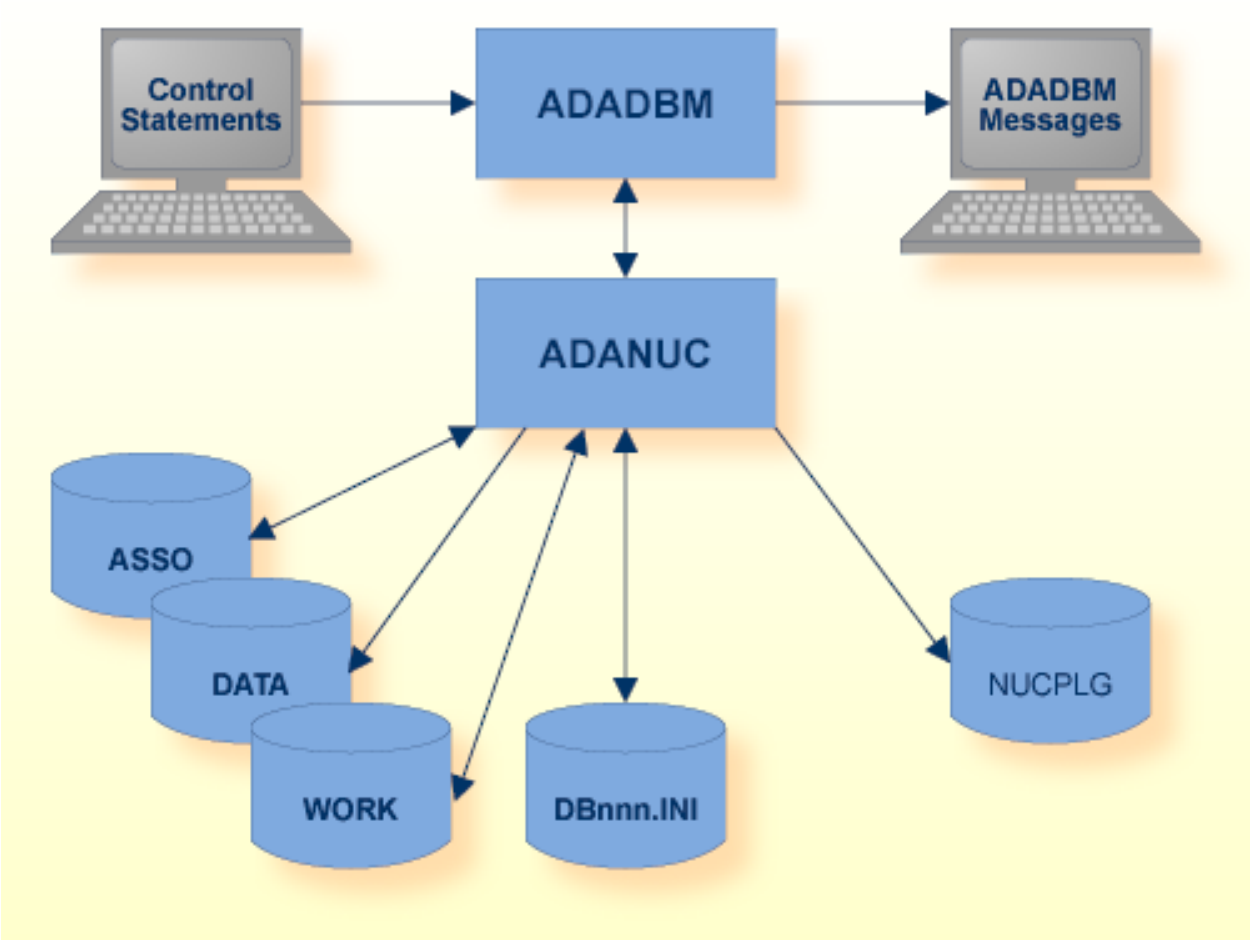
The ADADBM utility consists of the following functions which may be used to make modifications to the database:

- The ADD_CONTAINER function adds a new container file to the Associator or Data Storage data set;
- The ADD_FIELDS function adds new fields to the end of a file's FDT;
- The ALLOCATE NI, UI, AC or DS function increases the Normal Index, Upper Index, Address Converter or Data Storage space assigned to a file;
- The BT function sets/clears the NOBT flag for an existing file;
- The CHANGE function changes the standard length of a field in the Field Definition Table (FDT);
- The CHANGE_FIELDS function modifies one or more field specifications in a file;
- The DEALLOCATE functions are the inverse functions of ALLOCATE. The NI, UI, AC or DS function returns the Normal Index, Upper Index, Address Converter or Data Storage space which is no longer required by a file to the free space table (FST);
- The DELCP function deletes old checkpoint records from the checkpoint file in the specified range of dates;
- The DELETE function deletes a single Adabas file or a range of Adabas files from the database;
- The DELETE_DATABASE function deletes a database. Depending on the keyword specified, either just the containers are deleted, or the database directory and its content are deleted.
- The DISPLAY function displays the utility communication block (UCB);
- The DROP_FIELDS function marks the specified fields as not existing, which means that they can no longer be accessed ;
- The DROP_LOBFILE function is the inverse function of ADAFDU ADD_LOBFILE;
- The DROP_REFINT function drops an existing referential constraint;
- The EXTEND_CONTAINER function extends the last container file defined for the database;
- The NEW_DBID function changes the identifier of the database in use;
- The NEWWORK function allocates and formats a new Adabas WORK data set;
- The PGM_REFRESH function enables or disables refreshing an Adabas file inside an application program with an E1 command;
- The RECORDSPANNING function enables/disables record spanning for a file;
- The RECOVER function returns lost space to the free space table;

- The REDUCE_CONTAINER function reduces the size of the last container file defined for the database;
- The REFRESH function resets a single file or a range of files to the state of zero records loaded;
- The REMOVE_CONTAINER function removes a container file from the Associator, or Data Storage data set;
- The REMOVE_DROP function, used in conjunction with a subsequent REFRESH, removes dropped fields from the FDT;
- The REMOVE_REPLICATION function stops all replication processing and deletes the replication system files;
- The RENAME function changes the database name or names of loaded files;
- The RENUMBER function renumbers a loaded file or exchanges the numbers of loaded files;
- The REPLICATION_FILES function creates the system files required for Adabas - Adabas replication;
- The RESET function removes entries from the UCB;
- The RESET_REPLICATION_TARGET function resets the replication target flag of Adabas files;
- The REUSE function controls the reusage of Data Storage space or ISNs by Adabas;
- The SECURITY function sets the security mode of the database;
- The SYFMAX function specifies the maximum number of values generated for a system generated multiple-value field in the file specified.

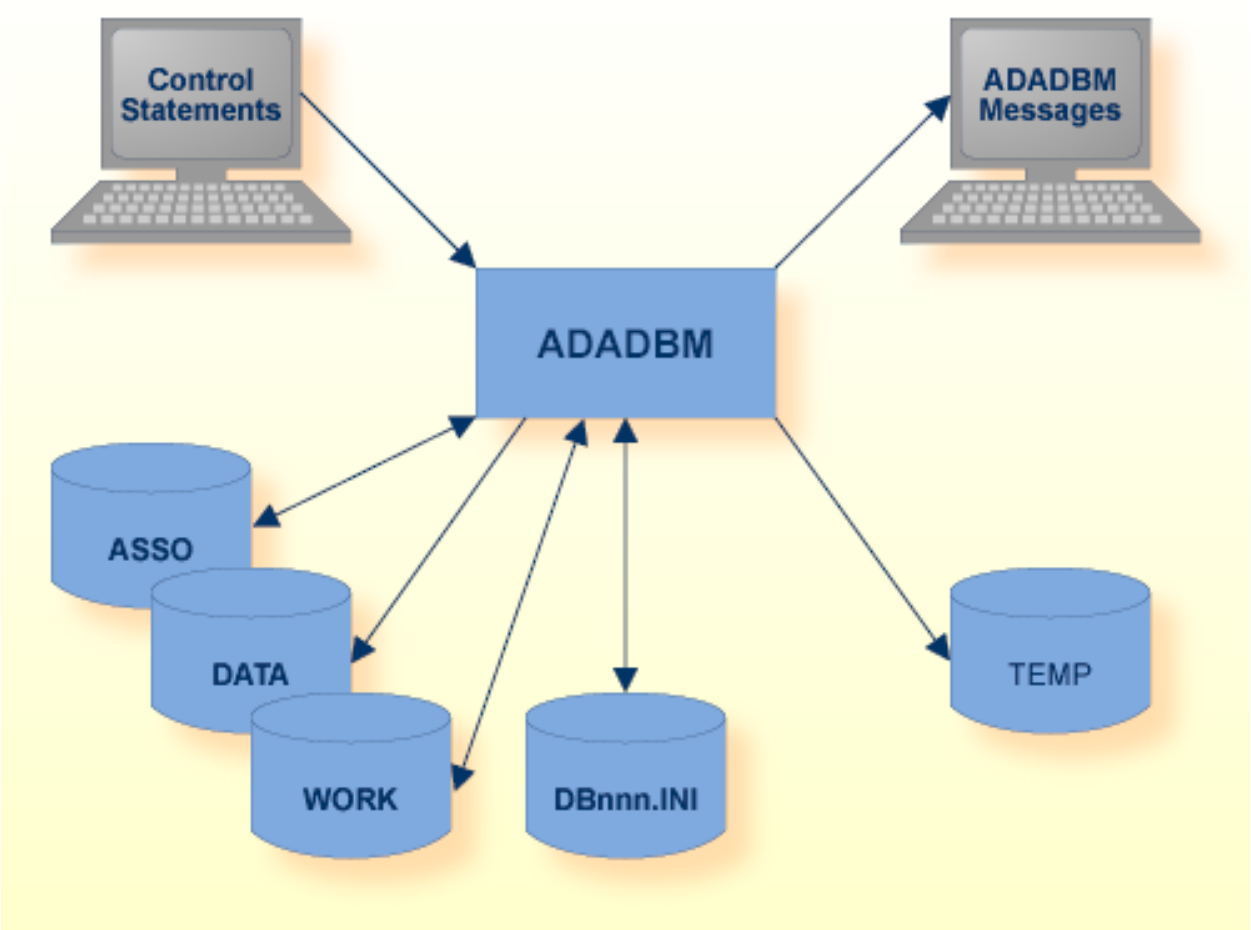
This utility is a multi-function utility. For more information about single- and multi-function utilities, *Adabas Basics*, *Using Utilities* in the Adabas documentation.

Procedure Flow



Online Mode

If the Adabas nucleus is active, ADADBM calls the nucleus to modify the database containers. For some tasks, no checkpoints are written, but the activity is logged in the database log, and in the case of a recovery, the activity is re-executed automatically.



Offline Mode

If the Adabas nucleus is not active, ADADBM itself modifies the database containers.

Data Set	Environment Variable / Logical Name	Storage Medium	Additional Information
Associator	ASSOx	Disk	
Data storage	DATAx	Disk	
DBnnn.INI		Disk	Adabas Extended Operations Manual
Protection Log (online mode only)	NUCPLG	Disk	Utilities Manual: ADANUC, ADAPLP
Temporary storage (offline mode only)	TEMP1	Disk	New WORK data set for the NEWWORK function. After this function is performed, the Work environment variable/logical name must be changed to point to the new Work data set.
Work	WORK1	Disk	

Checkpoints

The following table shows the nucleus requirements for each function and the checkpoints written:

Function	Nucleus must be active	Nucleus must NOT be active	Nucleus is NOT required	Checkpoint written
ADD_FIELDS			X	SYNP (offline) SYNX (online)
BT		X		SYNP
CHANGE	X			-
CHANGE_FIELDS			X	SYNP (offline) SYNX (online)
DELCP	X			SYNP
DELETE			X	SYNP (offline) SYNX (online)
DELETE_DATABASE		X		-
DISPLAY			X	-
DROP_FIELDS			X	SYNP (offline) SYNX (online)
DROP_LOBFILE			X	SYNP
EXTEND_CONTAINER		for WORK	for ASSO or DATA	SYNP
NEW_DBID		X (see note 1)		SYNP
NEWWORK		X (see note 1)		SYNP
PGM_REFRESH			X	SYNP (offline) SYNX (online)
RECORDSPANNING			X	SYNP (offline) SYNX (online)
REDUCE_CONTAINER			for ASSO or DATA	SYNP
REMOVE_CONTAINER		X		SYNP
REMOVE_REPLICATION		X		SYNP (offline)
REPLICATION_FILES			X	SYNP (offline) SYNX (online) (see note 2)
RESET_REPLICATION_TARGET				

Function	Nucleus must be active	Nucleus must NOT be active	Nucleus is NOT required	Checkpoint written
REUSE	X			SYNP
SECURITY		X		SYNP (offline)
SYFMAX			X	SYNP (offline) SYNX (online)

Notes:

1. Function requires exclusive access to the database container files.
2. In addition, ADADBM or ADAFDU checkpoints are generated (also in offline mode) to indicate the system file numbers deleted or generated.

Control Parameters

The following control parameters are available:

ADD_CONTAINER

```
ADD_CONTAINER = keyword
                [,BLOCKSIZE=number[K] ]
                ,SIZE = number [B|M]
```

The ADD_CONTAINER function adds a new container file to an existing Associator or Data Storage dataset in accordance with the keyword used. The keyword can take the values ASSO or DATA .

The new container file may be allocated on the same device as the current container files or it may be allocated on a different device type.

BLOCKSIZE = number[K]

This parameter specifies the block size in bytes (or in kilobytes, if "K" is specified after the number) of the new container file.

The default value for BLOCKSIZE is the block size of the last container file of the dataset in question that is currently present in the database.

Example

ADD_FIELDS

```
ADD_FIELDS = number {field_specification|FDT}... [END_OF_FIELDS]
```

The ADD_FIELDS function adds one or more new fields to the end of the file defined by 'number'. Specifying a LOB file is not permitted. The function is completed by entering END_OF_FIELDS.

If the field definitions are terminated with the END_OF_FIELDS parameter, this parameter must be specified in upper case when the LOWER_CASE_FIELD_NAMES parameter is used. In addition, the FDT parameter must also be specified in upper case when the LOWER_CASE_FIELD_NAMES parameter is used.

Note: It is not possible to add derived descriptors using ADADBM - you should use the utility ADAINV to do this instead.

field_specification

The field specification list is entered in the same way as the FDT input in ADAFDU:

```
level-number, name [,length] [,format] [(,option)...] [;long-name]
```

The first field to be added must be either a level-1 field or extend a trailing non-periodic group that is not inside a periodic group.

The NN option is not permitted. DE is only permitted when the Adabas nucleus is active and together with the NU or NC option. Otherwise use the ADAINV utility to give the new fields descriptor status. UQ is only permitted together with the DE option.

The long name is optional.

The long name can contain up to 32 characters in length. Special characters are permitted. If a long name is omitted, the two-character name is used as the long name.

The long names do not have to be unique within a file.

Note: When you add system-generated fields (fields with the field option SY) to a file, these fields have null values in the records that are already in the database - this is the same behavior as for fields without the SY option.

FDT

This parameter displays the FDT of the file to which the fields are to be added.

Example

```
adadbm: add_fields=12
adadbm: fdt

Field Definition Table:

   Level   | Name | Length | Format |   Options          | Flags    | Long name
-------------------------------------------------------------------------------------------------------------
  1        |  AA  |   15   |    A   | DE,UQ,NU           |          | REG-NUM
  1        |  AB  |    4   |    B   | FI                 |          | CHASSIS-NUM
  1        |  AC  |    8   |    A   | DE,NC              | RF       | PERSONNEL-ID
  1        |  CD  |        |        |                    |          | CAR-DETAILS
   2       |  AD  |   20   |    A   | DE,NU              | SP       | MAKE
   2       |  AE  |   20   |    A   | NU                 |          | MODEL
   2       |  AF  |   10   |    A   | DE,NU              |          | COLOR
  1        |  AG  |    4   |    U   | NU                 | SP       | YEAR
  1        |  AH  |    1   |    A   | DE,FI              |          | CLASS
  1        |  AI  |    1   |    A   | FI                 |          | LEASE-PUR
  1        |  AJ  |    8   |    U   | NU                 |          | DATE-ACQ
  1        |  AK  |        |        |                    |          | CAR-MAINTENANCE
   2       |  AL  |    3   |    A   | NU                 |          | CURR-CODE
   2       |  AM  |    4   |    P   | NU,MU              |          | MAINT-COST
-------------------------------------------------------------------------------------------------------------

   Type    | Name | Length | Format |   Options          | Parent field(s)   Fmt | ↩
Long name
-------------------------------------------------------------------------------------------------------------
 SUPER     |  AO  |   24   |    A   | NU                 | AG (   1 -   4 )   U  | ↩
SUPERDESCRIPTOR
           |      |        |        |                    | AD (   1 -  20 )   A  |
-------------------------------------------------------------------------------------------------------------
                              Referential Integrity
-------------------------------------------------------------------------------------------------------------
   Type    | Name | Refer. | Primary| Foreign | Rules
           |      | file   |  field |  field  |
-------------------------------------------------------------------------------------------------------------
 FOREIGN   |  HO  |     9  |   AA   |   AC    | DELETE_NOACTION  UPDATE_NOACTION
-------------------------------------------------------------------------------------------------------------


adadbm: 01,dd,1,a
adadbm: 01,gr;NEW-GROUP
adadbm: 02,g1,20,a,fi;NEW-FIELD
adadbm: fdt

Field Definition Table:
```

```
   Level   | Name | Length | Format |   Options          | Flags    | Long name
-----------------------------------------------------------------------------------------------
  1        |  AA  |   15   |    A   | DE,UQ,NU           |          | REG-NUM
  1        |  AB  |    4   |    B   | FI                 |          | CHASSIS-NUM
  1        |  AC  |    8   |    A   | DE,NC              | RF       | PERSONNEL-ID
  1        |  CD  |        |        |                    |          | CAR-DETAILS
   2       |  AD  |   20   |    A   | DE,NU              | SP       | MAKE
   2       |  AE  |   20   |    A   | NU                 |          | MODEL
   2       |  AF  |   10   |    A   | DE,NU              |          | COLOR
  1        |  AG  |    4   |    U   | NU                 | SP       | YEAR
  1        |  AH  |    1   |    A   | DE,FI              |          | CLASS
  1        |  AI  |    1   |    A   | FI                 |          | LEASE-PUR
  1        |  AJ  |    8   |    U   | NU                 |          | DATE-ACQ
  1        |  AK  |        |        |                    |          | CAR-MAINTENANCE
   2       |  AL  |    3   |    A   | NU                 |          | CURR-CODE
   2       |  AM  |    4   |    P   | NU,MU              |          | MAINT-COST
  1        |  DD  |    1   |    A   |                    |          | DD
  1        |  GR  |        |        |                    |          | NEW-GROUP
   2       |  G1  |   20   |    A   | FI                 |          | NEW-FIELD
-----------------------------------------------------------------------------------------------

   Type    | Name | Length | Format |   Options          | Parent field(s)   Fmt | ↩
Long name
-----------------------------------------------------------------------------------------------
 SUPER     |  AO  |   24   |    A   | NU                 | AG (   1 -   4 )   U  | ↩
SUPERDESCRIPTOR
           |      |        |        |                    | AD (   1 -  20 )   A  |
-----------------------------------------------------------------------------------------------
                             Referential Integrity
-----------------------------------------------------------------------------------------------
   Type    | Name | Refer. | Primary| Foreign | Rules
           |      | file   |  field |  field  |
-----------------------------------------------------------------------------------------------
 FOREIGN   |  HO  |     9  |   AA   |   AC    | DELETE_NOACTION  UPDATE_NOACTION
-----------------------------------------------------------------------------------------------

adadbm: end_of_fields
%ADADBM-I-FUNC, function ADD_FIELDS executed
```

ALLOCATE

Depending on the keyword specified (AC, DS, NI or UI), the ALLOCATE function increases the Normal Index (NI), Upper Index (UI), Address Converter (AC) or Data Storage (DS) by a given size. Each extent for the required type is checked to see whether it can be extended or not. A new extent is created if none of the current extents can be extended.

This function lets the DBA override the automatic extension method and can be used to preallocate smaller or larger extents. This can be useful when adding a large number of records. Exclusive control of the file is NOT required for this function.

FILE = number

This parameter specifies the file to be extended.

RABN = number

This parameter specifies the allocation start RABN. For NI or UI allocation for a LOB file, the block size of the RABN specified must be less than 16 KB. For DS allocation for a LOB file, the block size of the RABN specified must be 32 KB.

Example

```
adadbm: allocate=ni, file=11, size=100b
%ADADBM-I-ALLOC, 100 NI blocks allocated (611 - 710)


adadbm: allocate=ds, file=11, size=10
%ADADBM-I-DEALLOC, 2560 DS blocks allocated (245 - 2804)
```

BT

This function is used to set or clear the NOBT flag for an existing file. Specifying a LOB file is not permitted. The keyword can take the values YES or NO. It is not permitted to set BT=NO for files that are primary files of referential constraints. This function can only be executed in offline mode.

FILE = number

This parameter specifies the file for which the NOBT flag is to be set/cleared.

Examples

```
adadbm db=12 bt=yes,file=11           ; clear the NOBT flag for file 11
adadbm db=12 bt=no,file=11            ; set the NOBT flag for file 11
```

CHANGE

```
CHANGE = number, FIELD = string, LENGTH = number
```

This function changes the standard length of a field in the file specified by number. Specifying a LOB file is not permitted. The length of fixed storage fields (option FI) and floating point fields (format G) cannot be changed.

Changing the length of a field does not lead to any modifications within the Data Storage, but may affect programs that use the standard length.

Fields defined with the option SY=OPUSER cannot be changed.

FIELD = string

This parameter specifies the field whose standard length is to be changed. The field must be defined in the Field Definition Table for this file.

LENGTH = number

This parameter defines the new standard length of the field.

Example

```
adadbm: change=12, field=ac, len=11
%ADADBM-I-FUNC, function CHANGE executed
```

CHANGE_FIELDS

```
CHANGE_FIELDS = number {field_specification|FDT}... [END_OF_FIELDS]
```

The CHANGE_FIELDS function modifies one or more field specifications of the file defined by 'number'. The function is completed by entering END_OF_FIELDS.

If the field definitions are terminated with the END_OF_FIELDS parameter, this parameter must be specified in upper case when the LOWER_CASE_FIELD_NAMES parameter is used. In addition, the FDT parameter must also be specified in upper case when the LOWER_CASE_FIELD_NAMES parameter is used.

The changes that are allowed depend on the existence of records in the file. The following restrictions apply to all files:

- The field level number must not change;
- A group must remain a group;
- A periodic group must remain a periodic group;
- A field that is not a group or periodic group must not be converted to a group or a periodic group.

The following additional restrictions apply to non-empty files:

- Field length: the new length must be compatible with the new field format and field options. Such a change changes the behavior of adabas commands in which the field length is not specified in the format buffer;
- Field format: A may be changed to W and vice versa. It is the responsibility of the user to ensure that the field contains UTF-8 values if the format is changed from A to W. After changing the format from W to A, the field will contain UTF-8 values. Please note that the format specified in the format buffer of Adabas commands must be identical to the format in the field definition

for A and W fields - therefore it may be necessary to adapt existing programs accordingly. Other changes of the field format except for the change between A and W are not allowed.

- Field options: it is not allowed to add or remove the options DE, FI, HF, MU and UQ.

  The following field option changes are allowed:

Old Field Options	New Field Options	Comments
DT not set	DT set, TZ not set or set	No check is made to see whether the values in the database are compliant with the date/time edit mask specified. TZ may not be set for edit mask names DATE, TIME and NATDATE. **Caution:** Usually, the semantics of the field values defined with the TZ option and the field values defined without the TZ option are different: the field without the TZ value usually contains local time values, whereas the field with the TZ value contains UTC values. The field values are not updated automatically; it is the user's responsibility to ensure that necessary updates are made.
DT set	DT not set	Specifying a date/time edit mask for the field in the format buffer is no longer allowed.
HF set	HF not set	The behavior of cross-platform calls changes. The file must be empty to apply this change.
HF not set	HF set	
LA and LB not set	LA or LB set	The behavior of calls accessing the field with variable length changes.
LA set	LA not set, LB set	
LB set	LB not set, LA set	Only allowed if there is no LOB file defined for the file or if the field is a descriptor of the parent of a derived descriptor. The behavior of calls accessing the field with variable length changes.
NB not set	NB set	
NC and NN set	FI, NC, NU and NN not set	After this change, the field is no longer mandatory in the format buffer for N1/N2 commands; if not specified, the field gets the Adabas null value.
NC and NN set	NC set, NN not set	After this change, the field is no longer mandatory in the format buffer for N1/N2 commands; if not specified, the field gets the SQL null value.
NU set	NC set	Empty values are converted to NULL values. Note that NC set -> NU not set because NU and NC are mutually exclusive.
NV set	NV not set	The behavior of cross-platform calls changes.
NV not set	NV set	
SY not set	SY set	The behavior of A1, N1 and N2 commands changes. The field format must be compatible with the SY option. Note that no check is made to ensure that the existing values are reasonable.
SY set	SY not set	The behavior of A1, N1 and N2 commands changes.

Old Field Options	New Field Options	Comments
TR not set	TR set	
TZ not set DT set	TZ set, DT unchanged	Values in the database will be converted from UTC to local time when you specify a date/time edit mask.
TZ set	TZ not set	Values in the database are no longer converted from UTC to local time when you specify a date/time edit mask.

field_specification

The field specification list is entered in the same way as the FDT input in ADAFDU:

```
level-number, name [,length] [,format] [(,option)...] [;long-name]
```

FDT

This parameter displays the FDT of the file to which the fields are to be added.

Long names

The long name of a field or descriptor can be changed using the LONGNAME() keyword.

```
AA=LONGNAME(‘this is the long name of AA‘)
```

The long name can be changed to empty with LONGNAME(''), or reset to the short name with LONGNAME().

DBID

```
DBID = number
```

This parameter selects the database to be used.

Note: Utility functions which require or allow the nucleus to be shut down need logical assignments for the data sets.

Examples

```
adadbm: dbid=76
%ADADBM-I-DBOFF, database 76 accessed offline


adadbm: dbid=76
%ADADBM-I-DBON, database 76 accessed online
```

DEALLOCATE

```
DEALLOCATE = keyword,  FILE = number [,RABN = number],
             SIZE = numberB
```

DEALLOCATE = AC, DS, NI or UI

Depending on the keyword specified (AC, DS, NI or UI), this function releases a given amount of space from the Address Converter (AC), Data Storage (DS), Normal Index (NI) or Upper Index (UI).

If too much space is allocated to an extent, either automatically or manually, the DBA can release this space and return it to the Free Space Table (FST).

Deallocation is done for only one extent at a time. To release space from multiple extents, DEALLOCATE has to be called several times.

FILE = number

This parameter specifies the file.

RABN = number

This parameter specifies the first RABN to be deallocated. If this parameter is omitted, deallocation starts at the end of the last extent.

SIZE = numberB

This parameter specifies the size of the area to be deallocated, in blocks.

Example

```
adadbm: deallocate=ni, file=11, size=100b
%ADADBM-I-DEALLOC, 100 NI blocks deallocated (611 - 710)


adadbm: deallocate=ni, file=11, size=10b
%ADADBM-I-DEALLOC, 10 NI blocks deallocated (323 - 332)
```

DEFINE_REFINT

```
DEFINE_REFINT = number constraint_specification
```

This function adds a referential constraint to the file 'number', which contains a foreign key. The syntax for the constraint is the same as that used in the FDT file for ADAFDU and is described in *Administration, FDT Record Structure, Referential Constraints*. The constraint is also included in the FDT of the primary file, therefore, the constraint name must not already be defined in the primary file.

Adding a referential constraint is not allowed if the file specified as the primary file is defined with PGM_REFRESH=YES.

If there are violations of the referential integrity, adding of the constraint will fail - no updates are performed on the data of the file in order to establish referential integrity.

DELCP

```
DELCP = { * | ([absolute-date] [,[absolute-date]]) }
```

This function deletes checkpoint records from the checkpoint file.

If an asterisk '*' is entered, all checkpoint records are deleted.

Examples

```
adadbm: delcp=13-NOV-2006:15:09:48
%ADADBM-I-DELCP, 1 record deleted from CHECKPOINT file

adadbm: delcp=(13-NOV-2006:15:09:48,)
%ADADBM-I-DELCP, 81 records deleted from CHECKPOINT file

adadbm: delcp=(,14-NOV-2006:14:37:24)
%ADADBM-I-DELCP, 41 records deleted from CHECKPOINT file

adadbm: delcp=(14-NOV-2006:14:37:25,14-NOV-1996:14:38:15)
%ADADBM-I-DELCP, 42 records deleted from CHECKPOINT file
```

```
adadbm: delcp=*
%ADADBM-I-DELCP, 20 records deleted from CHECKPOINT file
```

DELETE

```
DELETE = (number [-number][,number[-number]]...)
```

The DELETE function deletes one or more files or ranges of files from the database and returns all space which was allocated for this file to the Free Space Table (FST). LOB files specified are ignored, but the LOB files assigned to all base files specified are deleted too. There must not be a referential constraint between a file that is to be deleted and another file, which is not specified. Deletion of system files is not allowed.

Note: If you want to stop using Adabas-to-Adabas replication, and therefore want to delete the replication system files, you must use ADADBM REMOVE_REPLICATION, *not* the DELETE FUNCTION.

ADADBM does not request confirmation of the files to be deleted, i.e. care should be taken when entering the file numbers.

Example

```
adadbm: delete=(4-11,14)
%ADADBM-I-DELETED, file 11 deleted
%ADADBM-I-DELETED, file 14 deleted
```

DELETE_DATABASE

```
DELETE_DATABASE = (keyword [,keyword])
```

The DELETE_DATABASE function deletes a database. Depending on the keyword specified (CONTAINER or FULL), either just the containers are deleted, or the database directory and its contents are deleted.

If you specify the keyword CONTAINER, the container files and the DBID entry in the [DB_LIST] section of the ADABAS.INI file will be deleted. If you specify the keyword FULL, the database directory and all of its contents will be deleted.

If the database you want to delete is encrypted, you can also remove the Key Encryption Key (KEK) for that database by specifying the 'KEK' keyword.

Caution: Deleting the Key Encryption Key for an encrypted database renders all backups of that database useless because you cannot restore from them without the key.

Examples:

```
adadbm: dbid=12 delete_database=container
```

The containers of the database with the DBID 12 will be deleted.

```
adadbm: dbid=12 delete_database=(container, kek)
```

The containers and Key Encryption Key (KEK) of the encrypted database with the DBID 12 will be deleted.

DISPLAY

```
DISPLAY = UCB
```

The DISPLAY function displays the utility communication block. This function can also be executed during a pending AUTORESTART.

Example:

```
adadbm: display=ucb

       Date/Time        Entry Id  Utility  Mode Files
       ---------        --------  -------  ---- -----
14-NOV-2006 14:38:40         233   adaopr  UTO  11
14-NOV-2006 14:38:42         234   adabck  ACC  *
```

The display shows the following items:

- DATE/TIME shows the date and time on which the given files were locked.
- ENTRY ID shows the allocated identification of the entry.
- UTILITY shows the name of the utility.
- MODE shows the mode in which the files are being accessed.
- FILES shows the file numbers of the files that are locked.

DROP_FIELDS

```
DROP_FIELDS = number {field_name|FDT}... [END_OF_FIELDS]
```

The DROP_FIELDS function drops one or more fields from the file defined by 'number' - the specified fields are marked as no longer existing and they cannot be accessed. Specifying a LOB file is not permitted. The function is completed by entering END_OF_FIELDS.

If the field definitions are terminated with the END_OF_FIELDS parameter, this parameter must be specified in upper case when the LOWER_CASE_FIELD_NAMES parameter is used. In addition, the FDT parameter must also be specified in upper case when the LOWER_CASE_FIELD_NAMES parameter is used.

If you specify a group or a periodic group, all of the fields that belong to the group or periodic group are dropped. You must not specify a field that is a descriptor or from which a descriptor is derived - if you want to drop such a field, you must first release all corresponding descriptors with ADAINV.

Once the DROP_FIELDS function has been executed, you can redefine the names of the dropped fields, for example using ADADBM's ADD_FIELDS function.

Notes:

1. The DROP_FIELDS function does not physically remove the fields. You should not drop and then add fields repeatedly, since this can cause the data records or the FDT of the file in question to overflow.
2. ADAMUP is not able to load data into a file that contains the same visible fields but which contains different dropped fields.

FDT

This parameter displays the FDT of the file from which the fields are to be dropped.

DROP_LOBFILE

```
DROP_LOBFILE = number ↩
```

The number must specify the file number of a base file with an empty assigned LOB file to be deleted.

DROP_LOBFILE is not allowed if the assigned LOB file is not empty.

DROP_REFINT

```
DROP_REFINT = number, NAME {=|:} constraint_name
```

The function removes a referential constraint from the file specified by 'number', which contains the foreign key. The constraint is also removed from the FDT of the primary file.

EXTEND_CONTAINER

```
EXTEND_CONTAINER = keyword, SIZE = number [B|M]
```

The EXTEND_CONTAINER function extends the last Associator, Data Storage or WORK container file defined for the database in accordance with the keyword used. The keyword can take the values ASSO, DATA or WORK.

Note: The WORK container can only be extended in the offline mode.

SIZE = number [B|M]

This parameter specifies the size of the expansion area in blocks (B) or megabytes (M). By default, the size is in megabytes.

[NO]LOWER_CASE_FIELD_NAMES

```
[NO]LOWER_CASE_FIELD_NAMES
```

If LOWER_CASE_FIELD_NAMES is specified, Adabas field names are not converted to upper case. If NOLOWER_CASE_FIELD_NAMES is specified, Adabas field names are converted to upper case. The default is NOLOWER_CASE_FIELD_NAMES.

This parameter must be specified before the ADD_FIELDS, CHANGE_FIELDS or DEFINE_REFINT parameters.

NEW_DBID

```
NEW_DBID = number
```

This function is used to change the identifier of the database in use. The new identifier may not already be in use by another active database.

Important: The purpose of the NEW_DBID parameter is to change the DBID stored in the ASSO1 (GCB RABN). Using the NEW_DBID parameter will NOT create a new database.

Possible use case for NEW_DBID usage:

A database backup of database 100 gets restored into an existing (!!!) database 101. I.e. the database 101 needs to be created first either with *adafrm* or *crdemodb*. Let environment variable BCK001 point to the backup file of database 100.

```
Run

   adabck db=101 restore=* NEW_DBID=101
```

Without the NEW_DBID parameter the adabck would abort.

NEWWORK

This function removes the existing WORK1 container file and replaces it with a new WORK1 container file. The new WORK1 container file is allocated and then formatted, if required.

Before a new WORK can be created, the nucleus and all utilities using the database must have been successfully terminated. Since this function requires the current WORK, it must not be deleted before NEWWORK has been executed. TEMP1 must point to the new work file when this function is used.

BLOCKSIZE = number[K]

This parameter specifies the block size in bytes (or in kilobytes, if "K" follows the number) of the new container file.

The minimum block size allowed is 3072 and the maximum block size allowed is 32768.

In addition to these minimum and maximum values, the following size restrictions apply in general to the block sizes for ASSO and WORK:

MAX (ASSOBLS) < WORKBLS

where MAX(ASSOBLS) represents the largest ASSO block size and WORKBLS represents the WORK block size.

The default value for BLOCKSIZE is the block size of the old WORK file.

PGM_REFRESH

```
PGM_REFRESH = keyword, FILE = number
```

This function is used to disable or enable refreshing an Adabas file inside an application program with an E1 command (ISN = 0, CID = BLANK). Specifying a LOB file is not permitted. The keyword can take the values YES or NO. It is not allowed to set PGM_REFRESH=YES for files that are primary files of referential constraints.

FILE = number

This parameter specifies the file for which refreshing is to be enabled/disabled.

RECORDSPANNING

This function is used to disable or enable record spanning for a file. The keyword can take the values YES or NO. The RECORDSPANNING function can only be specified for a base file that has a LOB file assigned.

FILE = number

This parameter specifies the file for which record spanning is to be enabled/disabled.

Examples

```
adadbm db=12 recordspanning=yes,file=9          ; enable record spanning for file 9
adadbm db=12 recordspanning=no,file=9           ; disable record spanning for file 9
```

RECOVER

```
RECOVER
```

This function returns lost space within the Associator and Data Storage to the Free Space Table (FST).

Space can be lost by a non-successful termination of an Adabas utility.

Example

```
adadbm: recover
%ADADBM-I-FUNC, function RECOVER executed
```

REDUCE_CONTAINER

```
REDUCE_CONTAINER = keyword, SIZE = number B
```

The REDUCE_CONTAINER function deallocates free space at the end of the Associator or Data Storage container defined for the database in accordance with the keyword used. The keyword can take the values ASSO or DATA.

The requested number of blocks must not be in use at the end of the container specified. If the complete space of one or more container extents is to be released, the container extents are removed. Note that the message informing you that a container extent is removed is not displayed by ADADBM if ADADBM is executed online - instead, it is included in the nucleus log.

If less blocks than requested are free at the end of the container, all free space at the end of the container is deallocated, and the following warning is displayed:

```
%ADADBM-W-PREDCONT, not all requested blocks removed
```

SIZE = number B

This parameter specifies the size by which the container is to be reduced, in blocks.

REFRESH

```
REFRESH = (number [-number][,number[-number]]...)
```

This function resets the files specified by 'number' to the state of zero records loaded. Only the first extents for Normal Index, Address Converter and Data Storage are kept. The remaining extents are returned to the Free Space Table (FST). The Upper Index is rebuilt and the unused Upper Index extents are then returned to the Free Space Table. LOB files specified are ignored, but the LOB files assigned to all base files specified are refreshed too. The primary file of a referential integrity constraint may be refreshed only if the foreign file of the referential constraint is also refreshed.

ADADBM does not request confirmation of the files to be refreshed, i.e. care should be taken when entering the file numbers.

This function is useful for clearing a test file in a test environment. This method is faster than deleting and reloading the file.

If the REMOVE_DROP function has been specified, dropped fields are removed from the FDT.

Example

```
adadbm: refresh=13
%ADADBM-I-REFRESH, file 13 refreshed
```

REMOVE_CONTAINER

```
REMOVE_CONTAINER = keyword
```

This function removes the last database container file from an existing Associator or Data Storage data set in accordance with the keyword used. The keyword can take the values ASSO or DATA.

The container file to be removed must not be in use when this function is executed, i.e. all of the blocks in the file must be free.

Before a container file can be removed, the nucleus and all of the utilities using the database must have terminated successfully.

Note: If you remove a container, the corresponding entry for this container file in the DBn-nn.INI file is deleted.

Example

```
adadbm: remove_container=data
%ADADBM-I-DMCONREM, container DATA2 removed
```

REMOVE_DROP

```
[NO]REMOVE_DROP
```

If you specify REMOVE_DROP, subsequent REFRESH functions will remove dropped fields from the FDT.

If you specify NOREMOVE_DROP, subsequent REFRESH functions will not remove dropped fields from the FDT.

The default is NOREMOVE_DROP.

Example

```
adadbm: remove_drop
adadbm: refresh=2
%ADADBM-I-REFRESH, file 2 refreshed
adadbm: refresh=3
%ADADBM-I-REFRESH, file 3 refreshed
adadbm: noremove_drop
adadbm: refresh=4
%ADADBM-I-REFRESH, file 4 refreshed
```

File 2 has been refreshed and dropped fields have been removed from the FDT. File 3 has been refreshed and dropped fields have been removed from the FDT. File 4 has been refreshed and dropped fields have not been removed from the FDT.

Note: `REMOVE_DROP` can only be used together with `REFRESH`. `REFRESH` deletes all records in the file. Thus, the file is empty after `REMOVE_DROP` and `REFRESH`. A new ADACMP compress with an FDT without the dropped fields is required to keep the data.

REMOVE_REPLICATION

```
REMOVE_REPLICATION
```

This function stops all replication processing and deletes all replication system files.

Note: This function is only relevant for customers who are using the Adabas Event Replicator with Adabas - Adabas replication.

RENAME

```
RENAME = number, NAME {=|:} string
```

This function changes the name of a file or a database. 'number' is the number of the file whose name is to be changed.

If 'number' is 0, the name of the database is changed. When the ADABAS.INI file is in read-only mode, the execution will be aborted and the name will roll back.

NAME {=|:} string

'string' is the new name of the specified file or database. If you specify an equals sign, the value given for 'string' will be converted to upper case; if you specify a colon, no upper-case conversion is performed.

Example

```
adadbm: rename=11, name=employee-file
%ADADBM-I-FUNC, function RENAME executed
```

RENUMBER

```
RENUMBER = (number, number)
```

This function changes the file number of a loaded Adabas file. If, however, the file's new number already belongs to a loaded file, the numbers of these files are exchanged.

The first 'number' is the file number currently assigned to the file. The second 'number' is the new file number to be assigned to the file.

Example:

```
adadbm: renumber=(12,14)
%ADADBM-I-RENUM, File 12 renumbered to 14
%ADADBM-I-RENUM, File 14 renumbered to 12
```

REPLICATION_FILES

```
REPLICATION_FILES = (file1, file2, file3, file4)
```

This functions performs all of the necessary initialization steps for the Adabas - Adabas replication and creates the replication system files.

file1
The metadata file.

file2
The replication transaction file.

file3
The replication command file.

file4
LOB file for the replication command file.

Notes:

1. This function is only relevant for customers who are using the Adabas Event Replicator with Adabas - Adabas replication.
2. After having initialized the Adabas - Adabas replication, the Adabas nucleus will only work after the Adabas Event Replicator has been installed - in particular, the replication exit is required.
3. The space required for the replication files is about 1 MB of ASSO space (small ASSO blocks with block size < 16 KB) and 5 MB of DATA space (block size 32 KB). If there is a high update load or when a replication is in the status Recording, the replication system files can grow because they store all of the update operations for the replicated files until they have been applied to the target database.

RESET

```
RESET = UCB, IDENT = { (number [,number]...) | * }
```

UCB

This function removes one or more entries from the utility communication block (UCB). This option can also be used during a pending AUTORESTART.

The UCB is used to control access to certain resources (the whole database, one or more files, etc.) within a database. It saves information about the Adabas utilities processing the database and the resources attached to them.

An entry is made in the UCB each time a utility is granted access to a resource. This entry contains information about the utility and the resources it locks. The utility automatically removes the entry when the resource is no longer required. Please refer to the DISPLAY=UCB function of this utility for information about how to display the contents of the UCB.

However, certain special conditions (e.g. an aborted ADAMUP) can cause entries to remain in the UCB and keep allocated resources locked. The RESET function releases these resources by removing one or more entries from the UCB.

IDENT = { (number [,number]...) | * }

This parameter specifies the unique ID of the entry to be removed. '*' removes all entries.

If the RESET UCB function is used offline, only '*' may be specified.

Example

```
adadbm: reset=ucb, ident=233
%ADADBM-I-RESUCB1, 1 entry deleted from UCB


adadbm: reset=ucb, ident=(235,234)
%ADADBM-I-RESUCB, 2 entries deleted from UCB


adadbm: reset=ucb, ident=*
%ADADBM-I-RESUCB1, 1 entry deleted from UCB
```

RESET_REPLICATION_TARGET

```
RESET_REPLICATION_TARGET = number
```

This function resets the replication target flag of Adabas files, after which they are handled as normal files again. If you specify 0, the replication target flag of all replication target files is reset; if you specify a file number, the replication target flag of the file with this file number is reset.

Notes:

1. This function is only relevant for customers who are using the Adabas Event Replicator with Adabas - Adabas replication.
2. After performing this function, a replication to this replication target is no longer possible - if the replication to this replication target is still active, a new update transaction on the replication source will set the replication to status Error. If you want to replicate data to this replication target again, a new initial state processing is required.

REUSE

```
REUSE = (keyword [,keyword]), FILE = number
```

The REUSE function controls the reuse of Data Storage space or ISNs by Adabas.

The File Control Block (FCB) for the specified file is modified to indicate the type of allocation technique to be used when adding new records or moving updated records.

The valid keywords are [NO]DS and [NO]ISN.

If the DS keyword is specified, Adabas scans the Data Storage Space Table (DSST) in order to locate a block with sufficient space. In this case, the first block found with sufficient space is used.

If the NODS keyword is specified, then all newly-added records, together with records that have to be moved to another block (as a result of record expansion caused by updating), are placed in

the last used block in the Data Storage extent allocated to the file. If there is not sufficient space in this block, the next block is used.

DS and NODS are mutually exclusive. The default is REUSE = DS.

If the ISN keyword is specified, Adabas may reuse the ISN of a deleted record.

If the NOISN keyword is specified, Adabas does not reuse the ISN of a deleted record for a new record. Each new record will be assigned the next-highest unused ISN.

ISN and NOISN are mutually exclusive. The default is REUSE = NOISN.

FILE = number

This parameter specifies the file.

Example

```
adadbm: reuse=nods, file=11
%ADADBM-I-FUNC, function REUSE executed


adadbm: reuse=(ds,isn), file=12
%ADADBM-I-FUNC, function REUSE executed
```

SECURITY

```
SECURITY = keyword
```

The SECURITY function sets the security mode of the database. The keyword can either be ACTIVE or WARN.

ACTIVE

The ACTIVE keyword enables the security functionality. ACTIVE implies that only authenticated users are allowed access to the database. Security violations, like authentication or authorization errors, are protocolled as "Error" in the Audit-Trail.

In case of a authentication violation, access to the database is rejected.

Important: Database security cannot be disabled once it has been activated.

WARN

The WARN keyword enables the security functionality. WARN implies that all users are allowed access to the database. Security violations, like authentication or authorization errors, are protocolled as "Warning" in the Audit-Trail. In case of a security violation, access to the operation is not rejected.

This mode is intended for transitioning applications to use a secure database. See also *Nucleus user exit 21*.

Important: The security mode WARN can only be changed to mode ACTIVE.

Default Mode

By default security is not enabled.

SYFMAX

```
SYFMAX = number, FILE = number
```

This parameter specifies the maximum number of values generated for a system generated multiple-value field in the file specified. There is no explicit maximum value, but you should bear in mind, that you can get a record overflow if the value is defined too high; the compressed data record should also fit into one DATA block is SYFMAX values are defined for system generated multiple-value fields. If the SYFMAX value is decreased and a record contains more values for system generated fields than the new value of SYFMAX, the excess values are removed during the next update operation for this record.

FILE = number

This parameter specifies the file.

Restart Considerations

ADADBM has no restart capability. At the end of each function, however, the system reports whether execution was successfully completed or not. If it is not successfully completed, the function has to be re-started.

RBAC Protection

The RBAC protection allows you to secure access to specific functions in this utility. See the list of the secured functions in *Adabas Role-Based Security (ADARBA)* in *Adabas Security Features* in the Adabas documentation.

8 ADADCU (Decompression Of Data)

- Functional Overview 102
- Procedure Flow 103
- Checkpoints 104
- Control Parameters 104
- Input and Output Data 112
- Restart Considerations 113

This chapter describes the utility "ADADCU".

Functional Overview

The decompression utility ADADCU decompresses records produced by the ADACMP, ADAMUP and ADAULD utilities.

The output of the decompression utility ADADCU can be used as input for a program using standard operating system file access methods.

It can also be used as input for the compression utility ADACMP once any required changes have been made to the data structure or once the data definitions of the file have been changed. A warning message is issued if the decompressed output file (DCUOUT file) created by the utility is empty.

ADADCU also decompresses files produced with the SINGLE option of the utilities ADAULD and ADACMP, but no parameter is required since this can be determined by the utility.

With ADADCU, the following functions are available:

- Complete records can be decompressed to the formats and lengths described in the FDT. A one-byte count field precedes each multiple-value field or periodic group in the output record.
- LOB field values can also be stored in separate files; the generated file names are put into the decompressed records.
- Several fields can be decompressed.

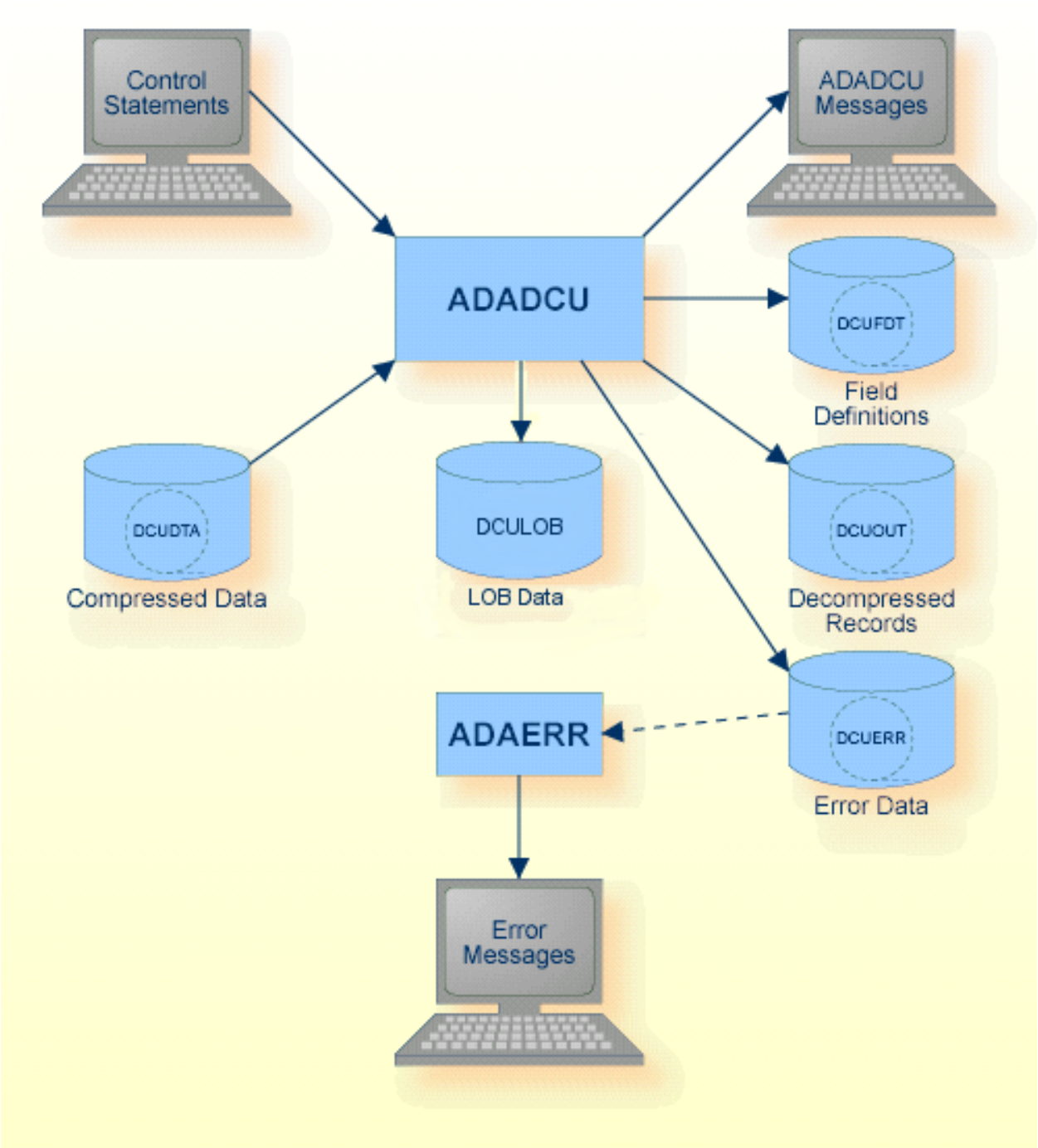
If several fields are decompressed, the fields can be re-arranged within a record, i.e. the record structure may be changed as follows:

- Field lengths can be changed;
- Field formats can be changed;
- Space can be allocated for subsequent addition of new fields using the literal element or blank element.

If the utility writes records to the error file, it will exit with a non-zero status.

This utility is a single-function utility. For more information about single- and multi-function utilities, *Adabas Basics*, *Using Utilities* in the Adabas documentation.

Procedure Flow



DCULOB is a directory where LOB values are stored in separate files. The sequential files DCUDTA and DCUERR can have multiple extents. For detailed information about sequential files with multiple extents, see *Adabas Basics*, *Using Utilities* in the Adabas documentation.

Data Set	Environment Variable / Logical Name	Storage Medium	Additional Information
Compressed data records	DCUDTA	Disk (* see note)	Output of ADACMP or ADAULD
Rejected data	DCUERR	Disk (* see note)	Output of ADADCU
Output data FDT	DCUFDT	Disk (* see note)	Output of ADADCU, Utilities Manual
Decompressed records	DCUOUT	Disk (* see note)	Utilities Manual
LOB data	DCULOB	Disk	Utilities Manual
Control statements	stdin		Utilities Manual
ADADCU messages	stdout		Messages and Codes

Note: (*) A named pipe can be used for this sequential file.

Checkpoints

The utility writes no checkpoints.

Control Parameters

The following control parameters are available:

```
D    [NO]DCUFDT

D    [NO]DST

     FDT

     FIELDS {field_specification | FDT},...[END_OF_FIELDS | .]

D    [NO]LOWER_CASE_FIELD_NAMES

D    MAX_DECOMPRESSED_SIZE = number  [K|M]

D    MUPE_C_L = {1|2|4}

     MUPE_OCCURRENCES
```

```
D    [NO]NULL_VALUE

D    NUMREC = number

D    RECORD_STRUCTURE = keyword

     SKIPREC = number

D    TARGET_ARCHITECTURE = (keyword[,keyword[,keyword]])

D    [NO]TRUNCATION

     TZ {=|:} [timezone]

D    [NO]USERISN

     WCHARSET = char_set
```

[NO]DCUFDT

```
[NO]DCUFDT
```

If this option is set to DCUFDT, the FDT information of the decompressed records is written to the sequential file DCUFDT. The default is NODCUFDT.

If you have used the FIELDS parameter (see below), the fields are written to the sequential file DCUFDT in the order specified in FIELDS. Thus, the fields in DCUFDT might be in a different order to those in the original FDT.

[NO]DST

```
[NO]DST
```

The parameter DST is required if a daylight saving time indicator is to be provided for date/time fields with the option TZ. The daylight saving time indicator will be appended behind the date/time value as a 2-byte integer value (format F) containing the number of seconds to be added to the standard time to get the actual time (usually 0 or 3600).

This parameter is required if there are records containing date/time values with the option TZ in the hour before the time is switched back to standard time, otherwise these values are written to the error file.

The default is NODST.

Notes:

1. The DST parameter is ignored if the FIELDS parameter is specified. In this case, you must specify a D element for fields with the daylight saving time indicator.

2. The DST parameter is not compatible with the RECORD_STRUCTURE = NEWLINE_SEPARATOR parameter because the daylight saving indicator in format F contains non-printable characters.

FDT

```
FDT
```

This parameter displays the FDT of the file containing the compressed records.

FIELDS

```
FIELDS {field_specification | FDT},...[END_OF_FIELDS | . ]
```

This parameter is used to specify a subset of fields given in the FDT and their format and length. This means that the decompressed records created do not have to contain all of the fields given in the FDT, or that fields can be decompressed with a different format or length. The syntax and semantics are the same as for the format buffer, with the exception that you can also specify an R-element (for LOB references) if the decompressed record contains the name of a file containing the LOB value instead of the LOB value itself.

While entering the specification list, the FDT function can be used to display the FDT of the file to be decompressed. The specification list can be terminated or interrupted by entering END_OF_FIELDS or '.'. The '.' option is an implicit END_OF_FIELDS and is compatible with the format buffer syntax. FIELDS or END_OF_FIELDS must always be entered on a line by itself, whereas the '.' may be entered on a line by itself or at the end of the format buffer elements. Processing may be continued after setting any option or parameter by entering FIELDS.

If the field definitions are terminated with the END_OF_FIELDS parameter, this parameter must be specified in upper case when the LOWER_CASE_FIELD_NAMES parameter is used. In addition, the FDT parameter must also be specified in upper case when the LOWER_CASE_FIELD_NAMES parameter is used.

Example

```
adadcu: fields
adadcu: ; This is a comment line
adadcu: AA,AB,6,A,AC,P ;  - inline comment -
adadcu: AD,AF,CBC,CB1-N . ; implicit END_OF_FIELDS
```

Field AA is output with default length and format, field AB with 6 byte alphanumeric and field AC with default length packed. Fields AD and AF are output in default length and format, followed by the one-byte binary multiple field count of field CB and all its occurrences.

[NO]LOWER_CASE_FIELD_NAMES

```
[NO]LOWER_CASE_FIELD_NAMES
```

If LOWER_CASE_FIELD_NAMES is specified, Adabas field names are not converted to upper case. If NOLOWER_CASE_FIELD_NAMES is specified, Adabas field names are converted to upper case. The default is NOLOWER_CASE_FIELD_NAMES.

This parameter must be specified before the FIELDS parameter.

MAX_DECOMPRESSED_SIZE

```
MAX_DECOMPRESSED_SIZE = number  [K|M]
```

This parameter specifies the maximum size of a decompressed record in bytes, kilobytes or megabytes, depending on the specification of "K" or "M" after the number. This parameter is intended to prevent very large decompressed record files from being created unintentionally (if you didn't consider that a file contained LOB data).

The default is 65536. This is also the minimum value.

Note: The exact definition of this parameter is the size of the I/O buffer required for the largest decompressed record. Only multiples of 256 bytes are used for the I/O buffers, which means that you must specify a value greater than or equal to the largest decompressed record (including the preceding length field) rounded up to the next multiple of 256.

MUPE_C_L

```
MUPE_C_L = {1|2|4}
```

If the data contain multiple-value fields or periodic groups, they are preceded by a binary count field with the length of MUPE_C_L bytes in the decompressed data.

The default is 1.

MUPE_OCCURRENCES

```
MUPE_OCCURRENCES
```

This parameter is used to print a list of all multiple fields and periodic groups together with their maximum occurrence. Such information is important because the decompressed data can become very large; if the range specified is too large, it is even possible to exceed the limit for the size of a decompressed record.

Example

The FDT of the file containing the compressed records is as follows:

```
1,AA,4,A,NU
1,PE,PE
2,PA,2,A,NU
2,PB,2,A,NU,MU
1,MM,2,U,NU,MU
1,X1,4,B
```

MUPE_OCCURRENCES might produce something of the form:

```
Name   Max occurrence
---------------------
PE                  4
PB                  8
MM                 12

%ADADCU-I-DCUREC, Number of decompressed records: 5023
%ADADCU-I-DCUIR, Number of incorrect records: 0
```

The file can then be decompressed as follows:

```
adadcu fields "AA,PA1-4,PB1-4(1-8),MM1-12,P,X1" ↩
```

Note: A record is considered to be incorrect if it has too many occurrences of a periodic group containing an MU field, and thus causes an internal overflow. It is not possible to decompress this record including the periodic group.

[NO]NULL_VALUE

```
[NO]NULL_VALUE
```

This parameter can be used to decompress records according to the standard FDT if the record contains NC option fields and their status values (S-elements). It is required if one or more fields have the null value, otherwise these records are put in the error file.

Example

If the FDT entry for field AA is: 1, AA, 2, A, NC, the effect of NULL_VALUE is as follows:

- NULL_VALUE: 1st output record (in hex) 00004141 (AA has a value), 2nd output record (in hex) FFFF2020 (AA has the null value).
- NONULL_VALUE: 1st output record (in hex) 4141 (AA has a value), 2nd output record (in hex) AA is null, therefore the record will be put into the error file.

The default is NONULL_VALUE.

NUMREC

```
NUMREC = number
```

This parameter specifies the number of records to be read from the input file and decompressed. If NUMREC is not specified and SKIPREC is also not specified, all records are processed.

Example

```
adadcu: numrec = 100
```

100 records are read and decompressed.

RECORD_STRUCTURE

```
RECORD_STRUCTURE = keyword
```

This parameter specifies the type of record separation used in the output file with the logical name DCUOUT. The following keywords can be used:

Keyword	Meaning
ELENGTH_PREFIX	The records in the DCUOUT file are separated by a two-byte exclusive length field. There is no separator character and the use of this format is not subject to any restrictions.
E4LENGTH_PREFIX	The records in the decompressed data file are separated by a 4-byte exclusive length field.
ILENGTH_PREFIX	The records in the DCUOUT file are separated by a two-byte inclusive length field. There is no separator character and the use of this format is not subject to any restrictions.
I4LENGTH_PREFIX	The records in the decompressed data file are separated by a 4-byte inclusive length field.
NEWLINE_SEPARATOR	The records in the DCUOUT file are separated by a new-line character. If the DCUOUT file is to be used as input for ADACMP, this keyword can only be specified if the field values of the output do not contain the new-line character

Keyword	Meaning
	(i.e. if there are only unpacked, alphanumeric and Unicode fields, and if the alphanumeric and Unicode fields only contain printable characters). This keyword and the USERISN parameter are mutually exclusive.
RDW	The records in the DCUOUT file are formatted such that they can be transferred to an IBM host using the FTP *site rdw* option.
RDW_HEADER	Like RDW, for decompressed records that can be compressed on a mainframe with HEADER=YES.
VARIABLE_BLOCKED	The records are stored as blocks. Each record begins with an inclusive four-byte length field.

The default is ELENGTH_PREFIX.

SKIPREC

```
SKIPREC = number
```

This parameter specifies the number of records to be skipped before decompression is started.

TARGET_ARCHITECTURE

```
TARGET_ARCHITECTURE = (keyword[,keyword[,keyword]])
```

This parameter specifies the format (character set, floating-point format and byte order) of the output data records. The following keywords can be used:

Keyword Group	Valid Keywords
Character set	ASCII EBCDIC
Floating-point format	IBM_370_FLOATING IEEE_FLOATING VAX_FLOATING
Byte order	HIGH_ORDER_BYTE_FIRST LOW_ORDER_BYTE_FIRST

If no keyword of a keyword group is specified, the default for this keyword group is the keyword that corresponds to the architecture of the machine on which ADADCU is running.

Note: The FDT is always output in ASCII format.

Example

If the output records are to be decompressed into IBM format, the user must specify the following:

```
TARGET_ARCHITECTURE = (EBCDIC, IBM_370_FLOATING, HIGH_ORDER_BYTE_FIRST)
```

[NO]TRUNCATION

```
[NO]TRUNCATION
```

This option enables or disables the truncation of alphanumeric field values.

NOTRUNCATION is the default. In this case, all the records with truncated alphanumeric field values are written to the error file.

Numeric values may not be truncated, and the value must fit into the standard or specified length. If truncated numeric values occur, the records concerned are written to the error file.

TZ

```
TZ  {=|:} [timezone]
```

The specified time zone must be a valid time zone name that is contained in the time zone database known as the Olson database (**https://www.iana.org/time-zones**). If a time zone has been specified, this time zone is used for time zone conversions of date/time fields with the option TZ.

The default is UTC, which is used internally to store date/time fields with option TZ; no conversion is required.

If you specify an empty value, no checks are made to ensure that date/time fields are correct.

Note: The time zone names are file names. Depending on the platform, these file names may or may not be case sensitive. Also, the time zone names, depending on the platform, may or may not be case sensitive.

Examples:

```
tz:Europe/Berlin
```

This is correct on all platforms.

```
TZ=Europe/Berlin
```

With this specification, TZ is converted to upper case EUROPE/BERLIN. This is correct on Windows, because file names are not case sensitive on Windows, but it is not correct on Linux, because Linux file names are case sensitive.

[NO]USERISN

```
[NO]USERISN
```

This parameter indicates whether the ISN is to be output together with each decompressed record or not. The user can specify whether the ISN currently assigned to the record is to be output with the decompressed data or whether it is to be omitted. If the user intends to reload the file with the same ISNs, the USERISN option must be set.

This parameter cannot be specified if RECORD_STRUCTURE=NEWLINE_SEPARATOR is specified.

If this parameter is omitted, the ISN is not output with each record.

NOUSERISN is the default.

Example

```
adadcu: userisn
```

The ISN is output with each record.

WCHARSET

```
WCHARSET = char_set
```

This parameter specifies the default encoding used in the decompressed file based on the encoding names listed at **https://www.iana.org/assignments/character-sets** - most of the character sets listed there are supported by ICU, which is used by Adabas for internationalization support.

Input and Output Data

The input for ADADCU must be a file containing compressed records such as those output by the unload utility ADAULD or by the compression utility ADACMP.

ADADCU decompresses each input record in accordance with the FIELDS specifications and writes the resulting record to the file with the logical name DCUOUT. The records are written in variable-length format. By default, the records are separated by a two-byte exclusive length field (see the parameter RECORD_STRUCTURE in this section for more detailed information).

If USERISN is specified, the data record is preceded by its ISN in the form of a four-byte binary number.

ADADCU Output

The sequential file DCUFDT (field definition information of the decompressed records) can be used as input for the file definition utility ADAFDU or for the compression utility ADACMP.

Rejected Data Records

Any records rejected by ADADCU are written to the ADADCU error file. The contents of this error file should be displayed using the ADAERR utility. Do not print the error file using the standard operating system print utilities since the records contain unprintable characters.

Restart Considerations

ADADCU does not have a restart capability. An interrupted ADADCU run must be re-executed from the beginning.

ADADCU does not update the database, therefore, no considerations regarding the status of the database need to be made before re-executing an interrupted ADADCU execution.

9 ADAERR (Error File Report)

- Functional Overview 116
- Procedure Flow 117
- Checkpoints 117
- Control Parameter 117
- Example 118
- Rejected Data Records 118

This chapter describes the utility "ADAERR".

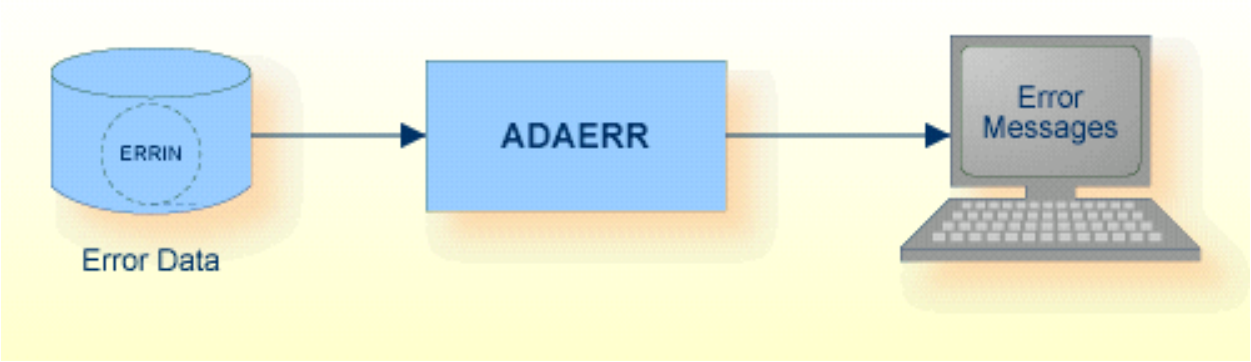
Functional Overview

The ADAERR utility displays the contents of error files generated by the utilities

- ADACMP
- ADADCU
- ADAINV
- ADAMUP
- ADAREC

This utility is a single-function utility. For more information about single- and multi-function utilities, *Adabas Basics*, *Using Utilities* in the Adabas documentation.

Procedure Flow



Data Set	Environment Variable / Logical Name	Storage Medium	Additional Information
Error data	ERRIN	Disk (* see note)	
Error messages	stdout / SYSOUTPUT		

Checkpoints

The utility writes no checkpoints.

Control Parameter

The following control parameter is available:

```
D    [NO]DUMP
```

[NO]DUMP

```
[NO]DUMP
```

If NODUMP is specified, only a description (length of record, ISN of the record etc.) of each error record will be output, but not the actual record content. See the section Rejected Data Records in this section for information on the contents of the error records.

If DUMP is specified, the record content will be dumped in addition to the record description. For ADACMP, the decompressed record will be dumped, whereas for ADADCU the compressed record will be dumped.

The default is NODUMP.

Example

```
$ adaerr
```

```
%ADAERR-I-STARTED,      11-OCT-2006 18:59:20, Version 6.1.1
%ADAERR-I-RECNOTF, Record NOT found for ISN 317 in file 49
%ADAERR-I-PLOGRB,  from record 1 in block 6 on PLOG 1
%ADAERR-I-IOCNT,        1 IO  on dataset ERRIN
%ADAERR-I-TERMINATED,   11-OCT-2006 18:59:20, elapsed time: 00:00:01
```

Rejected Data Records

Any records rejected by the following utilities are written to the error file in variable-length format.

- ADACMP
- ADADCU
- ADAINV
- ADAMUP
- ADAREC

The structure of the error records is contained as a header file `iodesam.h` in the subdirectory "Adabas/inc" of the installation directory on both Windows and Linux.

10 ADAFDU (File Definition)

- Functional Overview 120
- Procedure Flow 121
- Checkpoints 123
- Control Parameters 123
- Examples 134

This chapter describes the utility "ADAFDU".

Functional Overview

The file definition utility ADAFDU defines a new base file and/or a LOB file in a database. It does not require the Adabas nucleus to be active.

The field definitions for a base file, including special descriptor definitions and referential integrity definitions for foreign keys, are read from the sequential file FDUFDT; the field definition of a LOB file is predefined. Additional input for ADAFDU is provided by parameters.

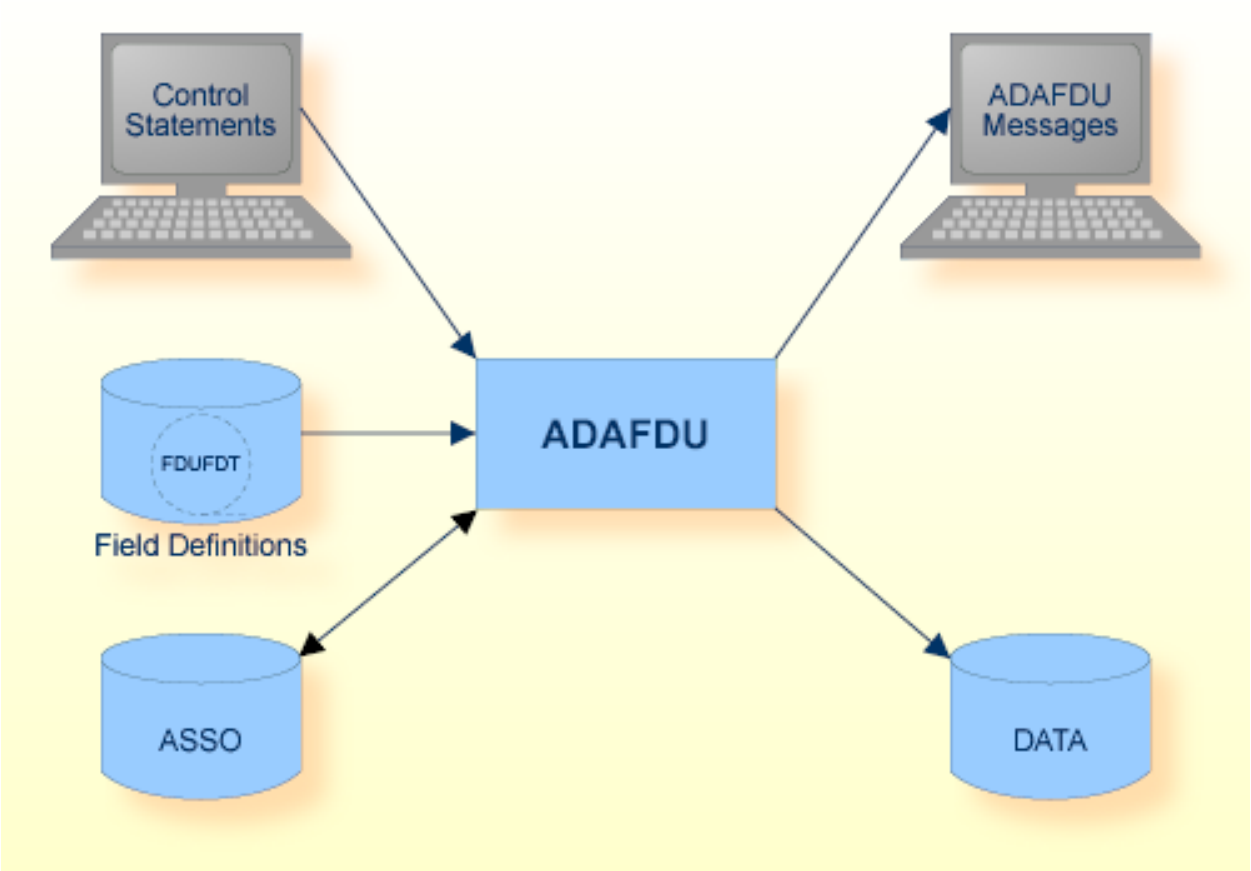
Note: If the new file contains collating descriptors, they are always created with ICU Version 5.4.

See *Administration*, *FDT Record Structure* in the Adabas on Windows and for Linux and Cloud documentation, for information about the syntax and use of the data definitions to define the logical structure of the file in the database.

See *Administration*, *Loading And Unloading Data, File Space Estimation* for information about formulae for calculating the Associator and Data Storage space requirements for a file.

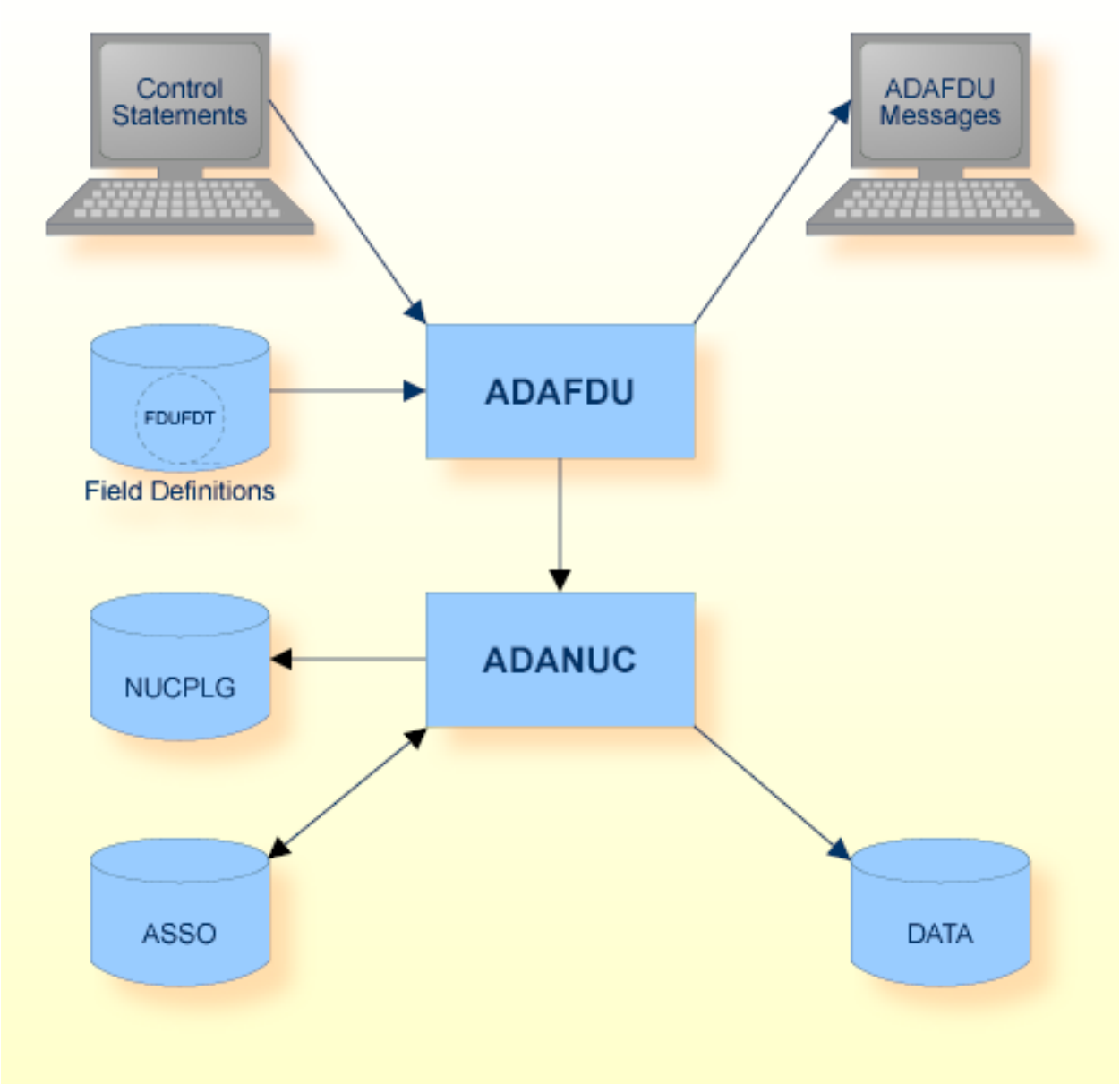
This utility is a single-function utility. For more information about single- and multi-function utilities, please see *Adabas Basics* > *Using Utilities* > *Single- and Multi-function utility* in the Adabas on Windows and for Linux and Cloud documentation.

Procedure Flow



Offline Mode

If the nucleus is not active, ADAFDU itself creates the new file in ASSO and DATA



Online Mode

If the nucleus is active, ADAFDU calls the nucleus to create the new file in ASSO and DATA. In this case, no checkpoint is written, but the file creation is logged in the database log, and in case of a recovery, the file is created automatically.

Data Set	Environment Variable / Logical Name	Storage Medium	Additional Information
Associator	ASSOx	Disk	
Data storage	DATAx	Disk	
FDT information	FDUFDT	Disk (* see note)	
Protection Log	NUCPLG	Disk	Utilities Manual, ADAPLP

Checkpoints

The utility writes a SYNP checkpoint if it is performed offline. If the utility is performed online, the file definition is written to the PLOG, a SYNX checkpoint is written.

Control Parameters

The following control parameters are available:

ACBLOCKSIZE

```
ACBLOCKSIZE = numberK
```

This parameter allows you to specify a block size for the allocation of the address converter extent.

Example:

```
acblocksize = 6k
```

The address converter will be allocated with a block size of 6 kilobytes.

If the database does not contain enough space with this block size, ADAFDU aborts.

ACRABN

```
ACRABN = number
```

This parameter specifies the RABN at which the space allocation for the Address Converter is to start.

This parameter can be used to allocate the Address Converter to a given container file extent.

If this parameter is omitted, ADAFDU assigns the starting RABN.

ADAM_KEY

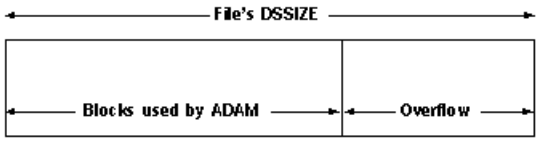
```
ADAM_KEY = key
```

If this parameter is specified, the file is defined as an ADAM file. The key can be either a descriptor name or the keyword 'ISN'. If an ADAM key is used, it must be defined with the UQ option in the FDT. It must not be a sub-, super-, phonetic or hyperdescriptor. It must not be a multiple-value field or a field within a periodic group. It must not have the NU/NC option.

ADAM_OVERFLOW

```
ADAM_OVERFLOW = number
```

This parameter specifies the number of DATA overflow blocks for the file. Overflow blocks are required in case ADAM-calculated blocks get full. The overflow blocks are taken from the end of the file's DATA blocks.



At least one overflow block must be allocated.

The maximum is DSSIZE - 1.

Note: When checking the maximum value, and DSSIZE is specified in megabytes, it is assumed that the Data Storage block size is 32 - independent of the actual value. If you want to specify a larger value for ADAM_OVERFLOW, which is possible with a smaller Data Storage block size, DSSIZE must be specified in blocks.

The default is 1.

ADAM_PARAMETER

```
ADAM_PARAMETER = number
```

This parameter specifies the number of consecutive ISNs to be stored in one block if the keyword 'ISN' is specified for the ADAM_KEY parameter.

If the ADAM key is a descriptor with fixed-point format, the parameter specifies the number of consecutive values for one block. For other key formats, it specifies an offset into the values. See Administration for more information.

A value may be specified in the range 1 to 10000.

The default value is 8.

ADD_LOBFILE

```
ADD_LOBFILE = (number, number)
```

The parameter ADD_LOBFILE is used to create a LOB file and assign it to an existing base file that is specified by the first number, the base file must not yet have an assigned LOB file. A LOB file, with the file number specified by the second number, is generated and assigned to the base file, and the base file is enabled for LOB processing. A file with the specified file number must not yet exist. The maximum number that can be specified is 32000. You can specify the parameters describing the data storage, the address converter, the normal and upper index of the LOB file, but the following should be taken into consideration:

- The block size for LOB file data blocks must be 32 KB.
- The block size for LOB NI blocks must be < 16 KB.

It is not possible to specify FILE if you specify ADD_LOBFILE, and vice versa.

Because there are some predefined requirements for a LOB file, not all the other ADAFDU parameters make sense in connection with ADD_LOBFILE, for example the ADAM_* parameters. These parameters are ignored by ADAFDU when the LOB file is added.

ASSOPFAC

```
ASSOPFAC = number
```

This parameter specifies the padding factor to be used for the file's index. The number specified is the percentage of each index block which is not to be used by a subsequent run of the mass update utility ADAMUP. This padding area is reserved for future use if additional entries have to be added to the block by the Adabas nucleus. This avoids the necessity of having to relocate overflow entries to another block.

A value may be specified in the range 0 to 95.

A small padding factor (0 to 10) should be specified if little or no descriptor updating is expected. A larger padding factor (10 to 50) should be specified if there is a large amount of descriptor updating in which new descriptor values are created.

You can change the padding factor at a later time using the utility ADAORD.

The default padding factor is 5.

[NO]BT

```
[NO]BT
```

If you specify NOBT, this file is a no-BT file, which means that modifications to this file are not made within normal transaction logic, and all modifications are kept in the database even if a transaction is backed out.

The default value is BT.

If Adabas is running in a cluster environment, the NOBT flag is ignored.

Note: The following points should be considered if the nucleus crashes:

- All database modifications for a no-BT file issued before the last ET command are applied to the database.
- It is not defined whether database modifications for a no-BT file issued after the last ET command are applied to the database or not.

[NO]CIPHER

```
[NO]CIPHER
```

This option can be used to enable or disable data record ciphering.

The default is NOCIPHER.

CONTIGUOUS

```
CONTIGUOUS = ( [AC] [,DS] [,NI] [,UI] )
```

This parameter is used to control ADAFDU's space allocations. If specified, ADAFDU ensures that only the first logical extent of the types specified is used.

By default, ADAFDU makes contiguous-best-try allocations.

DATAPFAC

```
DATAPFAC = number
```

This parameter specifies the padding factor to be used for the file's Data Storage. The number specified is the percentage of each data block which is not to be used when subsequently adding new records to the file with the mass update utility ADAMUP or with the Adabas nucleus. This padding area is reserved for future use if any record in the block requires additional space as a result of record updating by the Adabas nucleus. This avoids the necessity of having to relocate the record to another block.

A value in the range 0 to 95 may be specified.

A small padding factor (0 to 10) should be specified if there is little or no record expansion. A larger padding factor (10 to 50) should be specified if there is a large amount of record updating which will cause expansion.

You can change the padding factor at a later time using the utility ADAORD.

The default padding factor is 5.

DBID

```
DBID = number
```

This parameter selects the database to be used.

DSBLOCKSIZE

```
DSBLOCKSIZE = numberK
```

This parameter allows you to specify a block size for the allocation of the data storage extent.

Example:

```
dsblocksize = 6k
```

Data storage will be allocated with a block size of 6 kilobytes.

If the database does not contain enough space with this block size, ADAFDU aborts.

DSRABN

```
DSRABN = number
```

This parameter specifies the RABN at which the space allocation for Data Storage is to start.

This parameter can be used to allocate Data Storage to a given container file extent.

If this parameter is omitted, ADAFDU assigns the starting RABN.

DSSIZE

A contiguous-best-try allocation is made unless CONTIGUOUS=DS has been specified.

For non-ADAM files, this parameter can be omitted; in this case Adabas calculates a reasonable number of blocks to be used for Data Storage. If the size that is actually required is larger, the file is automatically increased.

FDT

```
FDT
```

If this parameter is specified, the FDT contained in the sequential file FDUFDT is displayed.

FILE

```
FILE = number
```

This parameter is required when a base file is to be created; it specifies the file number to be assigned to the file.

The 'number' specified must not be currently assigned to another file in the database and must not exceed the maximum file number defined for the database. The maximum number that can be specified is 32000.

File numbers can be assigned in any sequence.

It is not possible to specify FILE if you specify ADD_LOBFILE, and vice versa.

[NO]FORMAT

```
[NO]FORMAT
```

This option is used to control whether the RABNs allocated for the file's index and Data Storage are to be formatted or not. The RABNs of the file's Address Converter are always formatted.

The default is NOFORMAT.

LOBFILE

```
LOBFILE = number [, LOBSIZE = number[B|M]]
```

If LOBFILE is specified, a LOB file with the specified number is generated and assigned to the base file to be created, and the base file is enabled for LOB processing. A LOB file with the specified file number must not already exist. The maximum number that can be specified is 32000. You should take the following into consideration:

- The block size for LOB file data blocks will be 32 KB.
- The block size for LOB NI and UI blocks will be < 16 KB.
- LOBSIZE specifies the size in Data storage of the LOB file, analogously to the parameter DSSIZE.
- Adabas calculates reasonable sizes for the Address converter, the normal and upper index of the LOB file. If you want to specify these values yourself, you should create the base file first without specifying LOBFILE, and then you should call ADAFDU again and add the LOB file with the ADD_LOBFILE parameter.

[NO]LOWER_CASE_FIELD_NAMES

```
[NO]LOWER_CASE_FIELD_NAMES
```

If LOWER_CASE_FIELD_NAMES is specified, Adabas field names are not converted to upper case. If NOLOWER_CASE_FIELD_NAMES is specified, Adabas field names are converted to upper case. The default is NOLOWER_CASE_FIELD_NAMES.

MAXISN

```
MAXISN = number
```

This parameter specifies the highest ISN expected in the file. The file definition utility ADAFDU uses this parameter to determine the amount of space to be allocated for the file's Address Converter (AC). The default value for MAXISN is 5000.

A contiguous-best-try allocation is made unless CONTIGUOUS=AC has been specified.

If the an ISN reaches the MAXISN, Adabas automatically allocates more AC blocks to that particular file. The maximum ISN in an Adabas file can be equal to a 4 byte unsigned integer (4,294,967,294).

Note: The value is rounded up to the number of ISNs that fit into the Address converter blocks required to store MAXISN ISNs in the Address converter, the exact value used as MAXISN for the file is: (MAXISN specified / (Address converter block size / 4) + 1) * (Address converter block size / 4) -1. For example, using an Address converter with a block size of 4KB, the default value of 5000 is increased to (5000 / (4096 / 4) + 1) * (4096 / 4) -1 = 5119.

NAME

```
NAME {=|:} string
```

This parameter specifies the name to be assigned to the file. This name will appear together with data about this file in the database status report produced by the report utility ADAREP. A maximum of 16 characters are permitted. If you specify an equals sign, the value given for 'string' will be converted to upper case; if you specify a colon, no upper-case conversion is performed

NIBLOCKSIZE

```
NIBLOCKSIZE = numberK|(numberK,numberK)
```

This parameter allows you to specify a block size for the allocation of the Normal Index. Note that the Normal Index requires a block size >= 16 KB for large index values > 253 bytes, while a smaller block is allocated for descriptors with smaller descriptor values. The following must be taken into consideration:

- If you specify one block size, the file is created with all normal index blocks having this size.

- If you specify two block sizes, one value should be < 16K, and one value should be >= 16K. You should also specify two values for NISIZE; the first value for NIBLOCKSIZE corresponds to the first value of NISIZE, and the second value for NIBLOCKSIZE corresponds to the second value of NISIZE.

Examples:

```
niblocksize = 6k
```

The normal index will be allocated with a block size of 6 kilobytes.

```
niblocksize = (8k,32k)
nisize = (1000b,10m)
```

The normal index will be allocated with 1000 blocks of block size 8 KB and 10 MB of block size 32 KB.

If the database does not contain enough space with this block size, ADAFDU aborts.

NIRABN

```
NIRABN = number|(number,number)
```

This parameter specifies the RABN at which the space allocation for the Normal Index is to start. This parameter can be used to allocate the Normal Index to a given container file extent.

If two RABNs have been specified, one should have a block size < 16KB, and the other should have a block size of >= 16KB.

If this parameter is omitted, ADAFDU assigns the starting RABNs.

If both NIBLOCKSIZE and NIRABN are specified, the block sizes of the RABNs specified as NIRABN must be equal to the values specified as NIBLOCKSIZE.

NISIZE

If the block size cannot be derived from the NIBLOCKSIZE or the NIRABN parameter, the first value for NISIZE is used for blocks < 16KB, and the second value is used for blocks >= 16KB.

A contiguous-best-try allocation is made unless CONTIGUOUS=NI has been specified.

Examples:

```
adafdu: nisize = 100b
```

If the block size cannot be derived from the NIBLOCKSIZE or NIRABN parameter, 100 blocks with block size < 16KB are allocated for the Normal Index.

```
adafdu: nisize = (10m,1000b)
```

If the block size cannot be derived from the NIBLOCKSIZE or NIRABN parameter, 10 MB of blocks with block size < 16KB and 1000 blocks of block size >= 16KB are allocated for the Normal Index.

[NO]PGM_REFRESH

```
[NO]PGM_REFRESH
```

If PGM_REFRESH is specified, when the file loads, it can be refreshed by an E1 command, which resets to a state of no loaded records.

The default is NOPGM_REFRESH.

Note: Referential constraint is not allowed if the file specified as the primary file is defined with PGM_REFRESH=YES. Therefore, Response code 195 from ADABAS will come up when PGM_REFRESH is enabled for primary file.

REUSE

```
REUSE = ( keyword [,keyword] )
```

The REUSE parameter controls the reuse of Data Storage space or ISNs by Adabas.

REUSE = [NO]DS

NODS causes all newly-added records, together with records that have to be moved to another block (as a result of record expansion caused by updating) to be placed in the last used block in the Data Storage extent allocated to the file. If there is not sufficient space in this block, the next block is used.

If the DS keyword is specified, Adabas will scan the Data Storage Space Table (DSST) in order to locate a block with sufficient space. In this case, the first block found with sufficient space will be used.

The file control block for the specified file is modified to indicate the type of allocation to be used when adding new records or moving updated records.

The default value is DS.

REUSE = [NO]ISN

If REUSE is set to NOISN, Adabas does not reuse the ISN of a deleted record for a new record. Each new record will be assigned the next-highest unused ISN.

If REUSE is set to ISN, Adabas may reuse ISNs of deleted records. ISN reusage is done as follows: when a new record is stored in the database, an Address Converter (AC) block is read and checked for a free ISN. In order to keep the overhead for ISN reusage small, only one AC block is read - if no free ISN is found in the AC block, handling is the same as when ISN reusage is switched off.

Note: Setting REUSE to ISN, does not necessarily mean that ISN reusage is actually done. Because unsuccessful ISN reusage means an overhead for reading an additional AC block, Adabas sets ISN reusage to inactive if the probability of finding a reusable ISN is small. After deleting enough records, ISN reusage is then set to Active again.

The default value is NOISN.

Examples

```
adafdu: reuse = (isn, ds)
```

ISNs of deleted records can be reassigned to new records. The DSST is scanned for free space when a record is added to the database or when an updated record is moved in the database.

```
adafdu: reuse = isn
```

Reuse of data storage and ISNs is allowed.

```
adafdu: reuse = <cr>
```

Reuse of data storage and no reuse of ISNs is specified. This is the default setting.

SYFMAX

```
SYFMAX = number
```

This parameter specifies the maximum number of values generated for a system generated multiple-value field. There is no explicit maximum value, but you should bear in mind, that you can get a record overflow if the value is defined too high; the compressed data record should also fit into one DATA block is SYFMAX values are defined for system generated multiple-value fields.

The default value is 1.

UIBLOCKSIZE

```
UIBLOCKSIZE = numberK|(numberK,numberK)
```

This parameter allows you to specify a block size for the allocation of the Upper Index. Note that the Upper Index requires a block size >= 16 KB for large index values > 253 bytes, while a smaller block is allocated for descriptors with smaller descriptor values. The following must be taken into consideration:

- If you specify one block size, the file is created with all normal index blocks having this size.
- If you specify two block sizes, one value should be < 16K, and one value should be >= 16K. You should also specify two values for UISIZE; the first value for UIBLOCKSIZE corresponds to the first value of UISIZE, and the second value for UIBLOCKSIZE corresponds to the second value of UISIZE.

Examples:

```
uiblocksize = 6k
```

The upper index will be allocated with a block size of 6 kilobytes.

```
uiblocksize = (8k,32k)
uisize = (1000b,10m)
```

The upper index will be allocated with 1000 blocks of block size 8 KB and 10 MB of block size 32 KB.

If the database does not contain enough space with this block size, ADAFDU aborts.

UIRABN

```
UIRABN = number|(number,number)
```

This parameter specifies the RABN at which the space allocation for the Upper Index is to start. This parameter can be used to allocate the Upper Index to a given container file extent.

If two RABNs have been specified, one should have a block size < 16KB, and the other should have a block size of >= 16KB.

If both UIBLOCKSIZE and UIRABN are specified, the block sizes of the RABNs specified as UIRABN must be equal to the values specified as UIBLOCKSIZE.

If this parameter is omitted, ADAFDU assigns the starting RABN.

UISIZE

If the block size cannot be derived from the UIBLOCKSIZE or the UIRABN parameter, the first value for UISIZE is used for blocks < 16KB, and the second value is used for blocks >= 16KB.

A contiguous-best-try allocation is made unless CONTIGUOUS=UI has been specified.

Examples

Example:

```
adafdu: dbid = 1, file = 9, maxisn = 55000, dssize = 2000b, dsrabn = 30629,
adafdu: uisize = 50b, nisize = 300b,
adafdu: assopfac = 20, datapfac = 10
```

File 9 is to be loaded. The maximum number of expected records preset for the file is 55000. 2000 blocks are allocated for Data Storage. The Data Storage allocation will start at RABN 30629. 50 blocks are allocated for the Upper Index. 300 blocks are allocated for the Normal Index. The padding factor for the Associator is 20 percent. The padding factor for Data Storage is 10 percent.

11 ADAFIN (File Information Report)

- Functional Overview 136
- Procedure Flow 137
- Checkpoints 138
- Control Parameters 138

This chapter describes the utility "ADAFIN".

Functional Overview

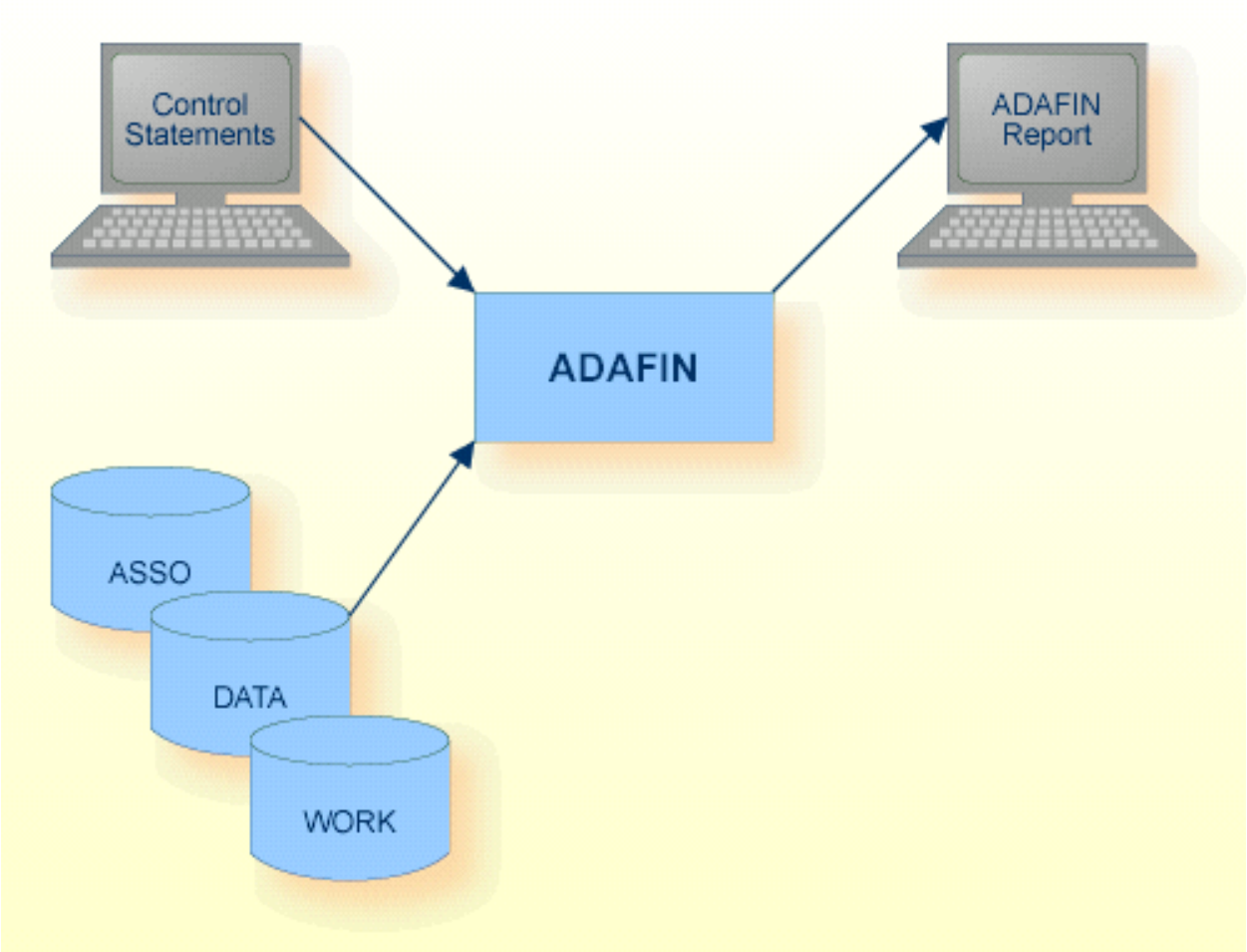
The file information utility ADAFIN displays

- the FDT,
- descriptor information, and
- the number of blocks in the Data Storage, Normal Index or Upper Index and their usage

of one or more selected files.

This utility is a multi-function utility. For more information about single- and multi-function utilities, see *Adabas Basics*, *Using Utilities* in the Adabas documentation.

Procedure Flow



Data Set	Environment Variable / Logical Name	Storage Medium	Additional Information
Associator	ASSOx	Disk	
Data storage	DATAx	Disk	
Control statements	stdin		Utilities Manual
ADAFIN messages	stdout		Messages and Codes
Work	WORK1	Disk	

Checkpoints

The utility writes no checkpoints.

Control Parameters

The following control parameters are available:

```
     ADAM_DS = keyword

M    DBID = number

     DESCRIPTOR = { = | : }{ * | (string [,string]...)  }

     FDT

M    FILE = { * | (number [-number] [,number [-number]]...)  }

D    [NO]HISTOGRAM

     USAGE = (keyword [,keyword [,keyword]])
```

ADAM_DS

```
ADAM_DS = keyword
```

This parameter can be used in conjunction with USAGE=DS for ADAM files. It selects the data section of the ADAM file for which information is to be displayed. The following keywords can be used:

Keyword	Meaning
FULL	All of the DS space is selected
ADAM	Only the ADAM area is selected
OVERFLOW	Only the ADAM overflow area is selected

DBID

```
DBID = number
```

This parameter selects the database to be used.

DESCRIPTOR

```
DESCRIPTOR = { = | : }{ * | (string [,string]...) }
```

This function defines the list of descriptors for which information is to be displayed. If more than one file is selected, information may only be requested for all descriptors (DESCRIPTOR = *).

The DESCRIPTOR function can only be executed if the selected files are not opened for update with the nucleus running. This function can only be selected in conjunction with the FILE parameter.

The DESCRIPTOR function does not synchronize against parallel updates (for example ADAINV REINVERT).

Examples

```
adafin: file=13, descriptor=ca

Database 76, File    13  (MISCELLANEOUS   )          27-OCT-2006 08:08:17


Descriptor CA , Format: A , Options: NU

                             min         max          ave
-----------------------------------------------------------
Length                         1         233        20.59
ISNs per value                 1           2         1.08

Values:       different:         86     total:         93
ASSO-Blocks:         NI:          2        UI:          1
```

```
adafin: file=(11,12), descriptor=*
Database 1, File    11  (EMPLOYEES-NAT   )          27-OCT-2006 08:09:39


Descriptor AA , Format: A , Options: UQ

                                  min              max                ave
-------------------------------------------------------------------------
Length                              8                8               8.00
ISNs per value                      1                1               1.00

Values:       different:            1,107     total:              1,107
ASSO-Blocks:         NI:                5        UI:                  1
```

```
Descriptor AE , Format: A , Options: None

                                  min              max                ave
-------------------------------------------------------------------------
Length                              3               17               6.78
ISNs per value                      1               19               1.37

Values:       different:                804     total:              1,107
ASSO-Blocks:         NI:                  4        UI:                  1


Descriptor AH , Format: P , Options: NC

                                  min              max                ave
-------------------------------------------------------------------------
Length                              4                4               4.00
ISNs per value                      1               43               1.20

Values:       different:                921     total:              1,107
ASSO-Blocks:         NI:                  4        UI:                  1


Descriptor AJ , Format: A , Options: NU

                                  min              max                ave
-------------------------------------------------------------------------
Length                              3               20               8.52
ISNs per value                      1              141               3.60

Values:       different:                307     total:              1,107
ASSO-Blocks:         NI:                  3        UI:                  1


Descriptor AO , Format: A , Options: None

                                  min              max                ave
-------------------------------------------------------------------------
Length                              6                6               6.00
ISNs per value                      1               99               6.62

Values:       different:                167     total:              1,107
ASSO-Blocks:         NI:                  2        UI:                  1


Descriptor AP , Format: A , Options: NU

                                  min              max                ave
-------------------------------------------------------------------------
Length                              2               25              12.56
ISNs per value                      1               75               4.67
```

```
Values:       different:                237     total:              1,107
ASSO-Blocks:         NI:                  3        UI:                  1


Descriptor AZ , Format: A , Options: NU,MU

                                   min              max                ave
---------------------------------------------------------------------------
Length                               3                3               3.00
ISNs per value                       1              843              86.28

Values:       different:                 21     total:              1,812
ASSO-Blocks:         NI:                  2        UI:                  1


Super-Descriptor H1 , Format: B , Options: NU
Parent field(s):            AU  (  1 -   2) U
                            AV  (  1 -   2) U

                                   min              max                ave
---------------------------------------------------------------------------
Length                               4                4               4.00
ISNs per value                       1               93               4.17

Values:       different:                259     total:              1,081
ASSO-Blocks:         NI:                  2        UI:                  1


Phonetic-Descriptor PH , Format: A , Options: None
Parent field(s):            AE              A

                                   min              max                ave
---------------------------------------------------------------------------
Length                               3                3               3.00
ISNs per value                       1               33               1.82

Values:       different:                608     total:              1,107
ASSO-Blocks:         NI:                  3        UI:                  1


Sub-Descriptor S1 , Format: A , Options: None
Parent field(s):            AO  (  1 -   4) A

                                   min              max                ave
---------------------------------------------------------------------------
Length                               4                4               4.00
ISNs per value                       1              208              85.15

Values:       different:                 13     total:              1,107
ASSO-Blocks:         NI:                  2        UI:                  1
```

```
Super-Descriptor S2 , Format: A , Options: None
Parent field(s):            AO  (  1 -   6) A
                            AE  (  1 -  20) A

                                  min              max                ave
-----------------------------------------------------------------------
Length                              9               23              12.78
ISNs per value                      1                5               1.05

Values:       different:              1,052     total:              1,107
ASSO-Blocks:         NI:                  6        UI:                  1


Super-Descriptor S3 , Format: A , Options: NU,PE
Parent field(s):            AR  (  1 -   3) A
                            AS  (  1 -   9) P

                                  min              max                ave
-----------------------------------------------------------------------
Length                             12               12              12.00
ISNs per value                      1               25               2.15

Values:       different:              1,567     total:              3,383
ASSO-Blocks:         NI:                 10        UI:                  1

Highest PE-occurrence:          5



Database 1, File    12  (VEHICLES        )          10-OCT-2006 14:30:39


Descriptor AA , Format: A , Options: UQ,NU

                                  min              max                ave
-----------------------------------------------------------------------
Length                              6               10               7.91
ISNs per value                      1                1               1.00

Values:       different:                772     total:                772
ASSO-Blocks:         NI:                  4        UI:                  1


Descriptor AC , Format: A , Options: None

                                  min              max                ave
-----------------------------------------------------------------------
Length                              1                8               7.74
ISNs per value                      1               24               1.16

Values:       different:                662     total:                773
```

```
ASSO-Blocks:         NI:                  3        UI:                  1


Descriptor AD , Format: A , Options: NU

                                    min              max                ave
-------------------------------------------------------------------------
Length                                2               14               6.63
ISNs per value                        1              179              17.17

Values:       different:                 45     total:                773
ASSO-Blocks:         NI:                  1        UI:                  1


Descriptor AF , Format: A , Options: NU

                                    min              max                ave
-------------------------------------------------------------------------
Length                                3               10               4.95
ISNs per value                        1              135              11.36

Values:       different:                 68     total:                773
ASSO-Blocks:         NI:                  1        UI:                  1


Descriptor AH , Format: A , Options: FI

                                    min              max                ave
-------------------------------------------------------------------------
Length                                1                1               1.00
ISNs per value                      169              329             257.66

Values:       different:                  3     total:                773
ASSO-Blocks:         NI:                  1        UI:                  1


Super-Descriptor AO , Format: A , Options: NU
Parent field(s):            AG  (  1 -   2) U
                            AD  (  1 -  20) A

                                    min              max                ave
-------------------------------------------------------------------------
Length                                4               16               8.63
ISNs per value                        1               45               4.29

Values:       different:                180     total:                773
ASSO-Blocks:         NI:                  2        UI:                  1

Total of 18 descriptors
```

Information about all descriptors in the specified files is displayed.

FDT

```
FDT
```

This parameter displays the Field Definition Tables (FDTs) of the files selected with the FILE parameter. This function can only be selected in conjunction with the FILE parameter.

Example

```
adafin: file=9, fdt
Database 189, File     9  (EMPLOYEES       )           8-MAY-2026 13:28:18


Field Definition Table:

   Level  | Name | Length | Format |   Options          | Flags   | Long name
-----------------------------------------------------------------------------------------------------------
  1       |  AO  |        |        |                    |         | PERSONNEL-DATA
   2      |  AA  |    8   |    A   | DE,UQ,NC,NN        | RP      | PERSONNEL-ID
   2      |  AB  |        |        |                    |         | ID-DATA
    3     |  AC  |    4   |    F   | DE                 |         | PERSONNEL-NO
    3     |  AD  |    8   |    B   | NU,HF              |         | ID-CARD
    3     |  AE  |    0   |    A   | NU,NV,NB,LA        |         | SIGNATURE
  1       |  BO  |        |        |                    |         | FULL-NAME
   2      |  BA  |   40   |    W   | NU                 |         | FIRST-NAME
   2      |  BB  |   40   |    W   | NU                 |         | MIDDLE-NAME
   2      |  BC  |   50   |    W   | DE,NU              | SP      | NAME
  1       |  CA  |    1   |    A   | FI                 |         | MAR-STAT
  1       |  DA  |    1   |    A   | FI                 |         | SEX
  1       |  EA  |    4   |    P   | DE,NC              |         | BIRTH
  1       |  FO  |        |        | PE                 |         | PRIVATE-ADDRESS
   2      |  FA  |   60   |    W   | NU,MU              |         | ADDRESS-LINE
   2      |  FB  |   40   |    W   | DE,NU              |         | CITY
   2      |  FC  |   10   |    A   | NU                 |         | POST-CODE
   2      |  FD  |    3   |    A   | NU                 |         | COUNTRY
   2      |  F1  |        |        |                    |         | PHONE-EMAIL
    3     |  FE  |    6   |    A   | NU                 |         | AREA-CODE
    3     |  FF  |   15   |    A   | NU                 |         | PRIVATE-PHONE
    3     |  FG  |   15   |    A   | NU                 |         | PRIVATE-FAX
    3     |  FH  |   15   |    A   | NU                 |         | PRIVATE-MOBILE
    3     |  FI  |   80   |    A   | DE,NU,MU           |         | PRIVATE-EMAIL
  1       |  IO  |        |        | PE                 |         | BUSINESS-ADDRESS
   2      |  IA  |   40   |    W   | NU,MU              |         | ADDRESS-LINE
   2      |  IB  |   40   |    W   | DE,NU              |         | CITY
   2      |  IC  |   10   |    A   | NU                 |         | POST-CODE
   2      |  ID  |    3   |    A   | NU                 |         | COUNTRY
   2      |  IE  |    5   |    A   | NU                 |         | ROOM NUMBER
   2      |  I1  |        |        |                    |         | PHONE-EMAIL
    3     |  IF  |    6   |    A   | NU                 |         | AREA-CODE
    3     |  IG  |   15   |    A   | NU                 |         | BUSINESS-PHONE
```

```
    3      |  IH  |   15   |    A   | NU                |          | BUSINESS-FAX
    3      |  II  |   15   |    A   | NU                |          | BUSINESS-MOBILE
    3      |  IJ  |   80   |    A   | DE,NU,MU          |          | BUSINESS-EMAIL
  1        |  JA  |    6   |    A   | DE                | SB,SP    | DEPT
  1        |  KA  |   66   |    W   | DE,NU             |          | JOB-TITLE
  1        |  LO  |        |        | PE                |          | INCOME
   2       |  LA  |    3   |    A   | NU                | SP       | CURR-CODE
   2       |  LB  |    6   |    P   | NU                | SP       | SALARY
   2       |  LC  |    6   |    P   | DE,NU,MU          |          | BONUS
  1        |  MA  |    4   |    G   | NU                |          | TOTAL INCOME(EUR)
  1        |  NO  |        |        |                   |          | LEAVE-DATE
   2       |  NA  |    2   |    U   |                   | SP       | LEAVE-DUE
   2       |  NB  |    3   |    U   | NU                | SP       | LEAVE-TAKEN
  1        |  OO  |        |        | PE                |          | LEAVE-BOOKED
   2       |  OA  |    8   |    U   | NU                |          | LEAVE-START
           |      |        |        | DT(DATE)          |          |
   2       |  OB  |    8   |    U   | NU                |          | LEAVE-END
           |      |        |        | DT(DATE)          |          |
  1        |  PA  |    3   |    A   | DE,NU,MU          |          | LANGUAGE
  1        |  QA  |    7   |    P   |                   |          | LAST UPDATE
  1        |  RA  |    0   |    A   | NU,NV,NB,LB       |          | PICTURE
  1        |  SO  |        |        | PE                |          | DOCUMENTS
   2       |  SA  |   80   |    W   | NU                |          | ↩
DOCUMENT-DESCRIPTION
   2       |  SB  |    3   |    A   | NU                |          | DOCUMENT-TYPE
   2       |  SC  |    0   |    A   | NU,MU,NV,NB,LB    |          | DOCUMENT
  1        |  TC  |   20   |    U   | DT(TIMESTAMP)     |          | CREATION TIME
           |      |        |        | SY=TIME,CR        |          |
  1        |  TU  |   20   |    U   | MU                |          | LAST UPDATES
           |      |        |        | DT(TIMESTAMP)     |          |
           |      |        |        | SY=TIME           |          |
-------------------------------------------------------------------------------------------------------------------------

   Type    | Name | Length | Format |   Options         | Parent field(s)   Fmt | ↩
Long name
-------------------------------------------------------------------------------------------------------------------------
 COLL      |  CN  |1,176   |        | NU,HE             | BC  de@collation=phone| ↩
PHONEBOOK
          |      |        |        |                   |     book              |  ↩
                                                                                 ↩

          |      |        |        |                   |     PRIMARY           |  ↩
                                                                                 ↩

 SUPER     |  H1  |    5   |    B   | NU                | NA (   1 -   2 )   U  | ↩
SUPERDESCRIPTOR
          |      |        |        |                   | NB (   1 -   3 )   U  |
 SUB       |  S1  |    2   |    A   |                   | JA (   1 -   2 )   A  | ↩
SUBDESCRIPTOR 1
 SUPER     |  S2  |   46   |    A   | NU                | JA (   1 -   6 )   A  | ↩
SUBDESCRIPTOR 2
          |      |        |        |                   | BC (   1 -  40 )   W  |
```

```
 SUPER    |  S3  |    9   |    A   | NU,PE             | LA (   1 -   3 )   A  | ↩
SUBDESCRIPTOR 3
          |      |        |        |                   | LB (   1 -   6 )   P  |
-----------------------------------------------------------------------------------
                            Referential Integrity
-----------------------------------------------------------------------------------
   Type   | Name | Refer. | Primary| Foreign | Rules
          |      | file   |  field |  field  |
-----------------------------------------------------------------------------------
 PRIMARY  |  HO  |    12  |   AA   |   AC    | DELETE_NOACTION  UPDATE_NOACTION
-----------------------------------------------------------------------------------
```

FILE

```
FILE = { * | (number [-number] [,number [-number]]...) }
```

This parameter selects one or more files from a database and displays information about these files in accordance with the following parameter. Specifying FILE = * selects all files.

[NO]HISTOGRAM

```
[NO]HISTOGRAM
```

If the HISTOGRAM option is selected, a graphical overview of the descriptor-value length distributions will be provided in all the information that is subsequently displayed by the DESCRIPTOR function.

If HISTOGRAM is used, it must be specified before the DESCRIPTOR parameter.

Using the HISTOGRAM option does not lead to additional I/Os on the data sets.

The default is NOHISTOGRAM.

Example (with HISTOGRAM)

```
adafin: file=9, histogram, descriptor=ap
Database 1, File     9  (EMPLOYEES       )          27-OCT-2006 08:12:44


Descriptor AP , Format: W , Options: NU

                                  min              max                ave
-------------------------------------------------------------------------
Length                              2               26              12.71
ISNs per value                      1               75               4.61

Values:       different:                240     total:              1,107
ASSO-Blocks:         NI:                  3        UI:                  1
```

```
Histogram of descriptor value length for descriptor AP


Length |              25%            50%            75%          100%| Frequency
-------+--------------+--------------+--------------+---------------+----------
   2   |*                                                            |         1
   3   |*                                                            |        22
   5   |*                                                            |         7
   6   |*                                                            |        26
   7   |******                                                       |       124
   8   |****                                                         |        83
   9   |******                                                       |       117
  10   |******                                                       |       119
  11   |***                                                          |        67
  12   |****                                                         |        83
  13   |*                                                            |        23
  14   |**                                                           |        47
  15   |**                                                           |        46
  16   |**                                                           |        46
  17   |*                                                            |        29
  18   |*****                                                        |       101
  19   |*                                                            |        27
  20   |*                                                            |        29
  21   |*                                                            |        33
  22   |*                                                            |        17
  23   |*                                                            |        20
  24   |*                                                            |        21
  25   |*                                                            |         5
  26   |*                                                            |        14
-------------------------------------------------------------------------------

adafin:
```

The information that is displayed has the following meaning:

Keyword	Meaning
Length	Each value n shown in this column indicates that there is a descriptor value with a length of n bytes in the file. The range of values in this column lies between the minimum (column "min") and maximum (column "max") values shown in the table before the histogram.
Frequency	The value shown in this column indicates the number of descriptor values for the given descriptor length. The sum of the values in the frequency column is equal to the total number of values for the descriptor in question.

If all of the descriptor values are of the same length, the histogram will be of an unusual type, e.g.:

```
adafin: file=9, histogram, descriptor=aa
Database 1, File     9  (EMPLOYEES       )          27-OCT-2006 08:15:16


Descriptor AA , Format: A , Options: UQ

                                  min              max                ave
-----------------------------------------------------------------------
Length                              8                8               8.00
ISNs per value                      1                1               1.00

Values:       different:              1,107     total:              1,107
ASSO-Blocks:         NI:                  5        UI:                  1



Histogram of descriptor value length for descriptor AA


Length |             25%            50%            75%          100%| Frequency
-------+-------------+-------------+-------------+-------------+----------
   8   |*************************************************************|     1,107
-----------------------------------------------------------------------
```

This histogram shows that the file only contains descriptor values that have a length of 8 bytes. The file contains a total of 1107 values for the descriptor AA.

Example (with NOHISTOGRAM)

```
adafin: file=9, histogram, descriptor=ap
Database 1, File     9  (EMPLOYEES       )          27-OCT-2006 08:14:24


Descriptor AP , Format: W , Options: NU

                                  min              max                ave
-----------------------------------------------------------------------
Length                              2               26              12.71
ISNs per value                      1               75               4.61

Values:       different:                240     total:              1,107
ASSO-Blocks:         NI:                  3        UI:                  1
```

USAGE

```
USAGE = (keyword [,keyword [,keyword]])
```

Depending on the keyword specified, this parameter displays the percentage of used blocks in the file.

Keyword	Meaning
DS	Displays statistics of used blocks in the Data Storage;
NI	Displays statistics of used blocks in the Normal Index;
UI	Displays statistics of used blocks in the Main/Upper Index.

Example

```
adafin: file=13, usage=ds

Database 76, File    13  (MISCELLANEOUS   )          27-OCT-2006 08:16:18


DS - Blocks allocated =         50 , used =         49 , unused =          1

Records:  Number      =        179
          Length: max =      1,991 , min  =        260 , avg    =     997.47

  0%:                                                                 0 blocks
  5%:                                                                 0 blocks
 10%:                                                                 0 blocks
 15%:                                                                 0 blocks
 20%:                                                                 0 blocks
 25%:                                                                 0 blocks
 30%:                                                                 0 blocks
 35%:                                                                 0 blocks
 40%:                                                                 0 blocks
 45%:                                                                 0 blocks
 50%:                                                                 0 blocks
 55%:                                                                 0 blocks
 60%:                                                                 0 blocks
 65%:                                                                 0 blocks
 70%:****                                                             2 blocks
 75%:                                                                 0 blocks
 80%:**                                                               1 block
 85%:*************                                                    6 blocks
 90%:******************************                                  13 blocks
 95%:*************************************************************** 27 blocks
100%:                                                                 0 blocks
```

Information about the used data blocks of file 13 in database 76 is displayed. 50 DS blocks are allocated, of which 49 are in use and 1 is unused. The total number of records is 179, with the record

length varying between a maximum of 1991 and a minimum of 260. The average record length is 997.47. The following lines give an overview of the number of blocks that are used up to a given level. The majority of the blocks (27) is used up to between 90% and 95%.

Example (for ADAM file)

```
adafin: file = 8
adafin: adam_ds = full
adafin: usage = ds

Database  30, File   8 (ADAM_FILE       )      11-OCT-2006 12:08:57


ADAM key = FF     ADAM parameter =     5               ADAM_DS = FULL

DS - Blocks used for ADAM =          94
Total overflow blocks     =           1,  used = 1

Records:  Number      =    3863
          In ADAM area=    3860 , ovfl =        3
          Length: max =       9 , min  =        9 , avg    =      9.00

  0%: ********                                                10 blocks
  5%: ***                                                      4 blocks
 10%:                                                          0 blocks
 15%:                                                          0 blocks
 20%:                                                          0 blocks
 25%:                                                          0 blocks
 30%: *                                                        1 block
 35%:                                                          0 blocks
 40%:                                                          0 blocks
 45%:                                                          0 blocks
 50%: *                                                        1 block
 55%:                                                          0 blocks
 60%:                                                          0 blocks
 65%: ************************************************     74 blocks
 70%:                                                          0 blocks
 75%:                                                          0 blocks
 80%:                                                          0 blocks
 85%:                                                          0 blocks
 90%:                                                          0 blocks
 95%: ***                                                      3 blocks
100%: *                                                        2 blocks
```

Information about all data blocks of file 8, which is an ADAM file, is displayed. The ADAM parameter is set to 5. 94 blocks are used for the ADAM area, with 1 block reserved for overflow. The ADAM area contains 3860 records, and 3 records are in the overflow area.

12 ADAFRM (Format And Create A New Database)

- Functional Overview 152
- Procedure Flow 154
- Checkpoints 155
- Control Parameters 155
- Restart Considerations 159
- Control Statement Examples 160

This chapter describes the utility "ADAFRM".

Functional Overview

The utility ADAFRM creates the container files (ASSO, DATA, WORK) assigned to the database and establishes the database including the database system files. It can also be used to format the TEMP and SORT files.

The database is included in the ADABAS.INI file.

If the file DBnnn.INI does not yet exist, ADAFRM creates the DBnnn.INI file, including the default parameters derived from ADABAS.INI, and stores it in the appropriate database directory (please refer to *Adabas Extended Operation* in the Adabas documentation for further information about the DBnnn.INI files).

The following rules apply for determining the locations for the container extents to be created:

1. If an environment variable for the container extent exists, use the environment variable.
2. If the DBnnn.INI file already exists before ADAFRM is started, and it contains an entry for the container extent, use the entry in the DBnnn.INI file.
3. Otherwise create the container extent in the database directory ($ADADATADIR/dbnnn on Linux, %ADADATADIR%\dbnnn on Windows) with name *XXXXx.nnn*, where *XXXX* is the container type, *x* the container extent and *nnn* the database number.

Exceptions are the SORT and TEMP containers if they are created without creating a database at the same time; here the following rules apply:

1. If an environment variable for the container extent exists, use the environment variable.
2. Otherwise create the container in the current directory with the name *XXXXx*, where *XXXX* is the container type, and *x* the container extent.
3. If a database ID is specified, the corresponding encryption information stored in ASSO1 is read. If the database has been created with encryption, TEMP containers are also encrypted.

If you create a database, but not if you only create the SORT or TEMP containers, the created container extents are stored in theDBnnn.INI file. If the DBnnn.INI file already was created before ADAFRM was started, all other values in the file are not changed; if the file didn't exist, it is created with default values.

If you create only the SORT or TEMP containers without creating a database, no updates are performed in a DBnnn.INI file. If a dbid is supplied, the corresponding entries are added in the DBnnn.INI file in case they are missing.

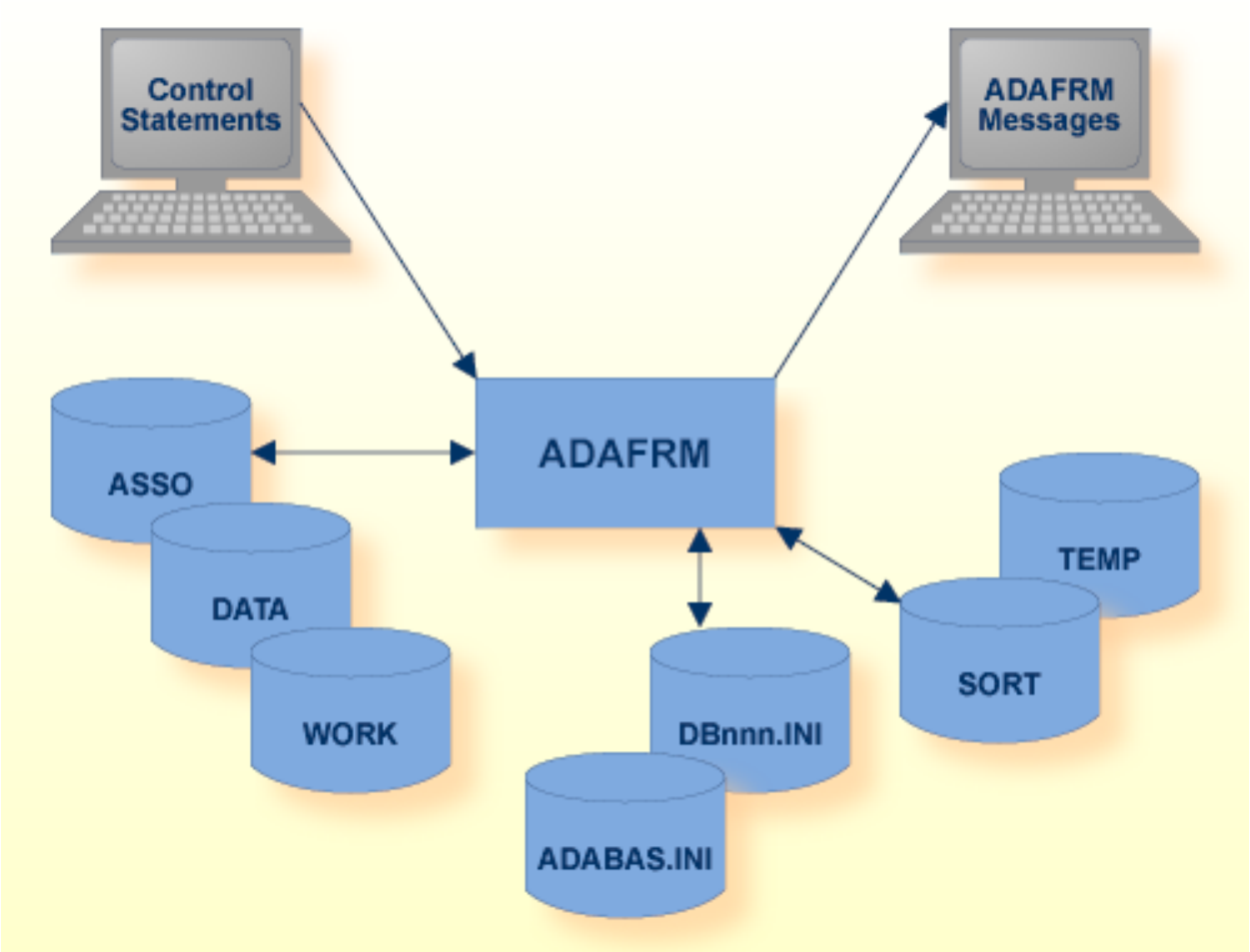
If you try to reformat a container file, the utility terminates with an error message. This ensures that the database is not accidentally overwritten.

If you create an encrypted database, the encryption algorithm and the key management system have to be specified. The ASSO and DATA container files are encrypted with the specified encryption algorithm. The key management system manages the corresponding encryption keys.

Note: In order to use the encryption functionality, the Adabas Encryption for Linux (AEL) license is required.

This utility is a single function utility.

Procedure Flow



If a database is to be formatted:

Data Set	Environment Variable / Logical Name	Storage Medium	Additional Information
ADABAS.INI		Disk	Adabas Extended Operations Manual
Associator	ASSOx	Disk	
Data storage	DATAx	Disk	
DBnnn.INI		Disk	Adabas Extended Operations Manual
Control statements	stdin	Utilities Manual	
ADAFRM messages	stdout		Messages and Codes
Work	WORK1	Disk	

If a TEMP or SORT is to be formatted:

Data Set	Environment Variable / Logical Name	Storage Medium	Additional Information
Sort storage	SORTx	Disk	
Control statements	stdin		Utilities Manual
ADAFRM messages	stdout		Messages and Codes
Temporary storage	TEMPx	Disk	

Checkpoints

The utility writes no checkpoints.

Control Parameters

The following control parameters are only used when establishing a new database:

```
D    ASSOBLOCKSIZE = (number[K] [,number[K]] ... )

M    ASSOSIZE = (number[B|M] [,number[B|M]]...)

D    DATABLOCKSIZE = (number[K] [,number[K]] ... )

M    DATASIZE = (number[B|M] [,number[B|M]]...)

     DBID = number

     ENCRYPTION = keyword

     KMSTARGET = string

D    NAME {=|:} string

M    SORTSIZE = (number[M] [,number[M]] ... )

D    SYSFILES = (number, number, number)

M    TEMPSIZE = (number[M] [,number[M]] ... )

D    WORKBLOCKSIZE = number[K]

M    WORKSIZE = number[M | B]
```

ASSOBLOCKSIZE

```
ASSOBLOCKSIZE = (number[K] [,number[K]] ... )
```

This parameter specifies the block sizes that are to be used for the Associator container file(s). The first block size corresponds to ASSO1, the second to ASSO2 etc.

If block sizes are not specified, the default of 4K will be used.

For ASSO1, only blocks sizes from 2K to 8K can be specified. For ASSO2 to ASSOn, block sizes between 1K and 32K are permitted.

Note: The ASSOBLOCKSIZE parameter should be specified once for each ASSOSIZE that is specified, i.e. these parameters should be specified in pairs. If ASSOSIZE is specified more frequently than ASSOBLOCKSIZE, then the last specified block size will be used for the containers that do not have a block size specified. The default value will be used if ASSOBLOCKSIZE is not specified at all.

ASSOSIZE

```
ASSOSIZE = (number[M|B] [,number[M|B]]...) ↩
```

This parameter specifies the number of blocks or megabytes to be assigned to the Associator.

If the Associator is to be contained in more than one physical file, the size of each file must be specified.

DATABLOCKSIZE

```
DATABLOCKSIZE = (number[K] [,number[K]] ... )
```

This parameter specifies the block sizes that are to be used for the Data Storage container file(s). The first block size corresponds to DATA1, the second to DATA2 etc.

If block sizes are not specified, the default of 32K will be used.

Note: The DATABLOCKSIZE parameter should be specified once for each DATASIZE that is specified, i.e. these parameters should be specified in pairs. If DATASIZE is specified more frequently than DATABLOCKSIZE, then the last specified block size will be used for the containers that do not have a block size specified. The default value will be used if DATABLOCKSIZE is not specified at all.

DATASIZE

```
DATASIZE = (number[B|M] [,number[B|M]]...) ↩
```

This parameter specifies the number of blocks or megabytes to be assigned to the Data Storage.

If the Data Storage is to be contained in more than one file, the size of each file must be specified.

If a 'B' is appended to the number, the size is in blocks, otherwise it is in megabytes.

DBID

```
DBID = number
```

This parameter selects the database to be created.

The minimum value is 1 and the maximum value is 255.

Note: This parameter only needs to be set when formatting ASSO, DATA and WORK. When formatting TEMP and/or SORT, for an encrypted database, a dbid is mandatory. The encryption information stored in ASSO1 is read, and the TEMP and/or SORT container are encrypted accordingly. For the unencrypted case, a dbid may be given, but is not mandatory.

ENCRYPTION

```
ENCRYPTION = keyword
```

This parameter specifies that the database to be created is encrypted, and assigns the encryption algorithm. The keyword can take the values AES_256_XTS, AES_128_XTS and NO. Depending on the keyword specified, the ASSO and DATA container files are encrypted using XTS Advanced Encryption Standard with a key length of 256 bits (AES_256_XTS), a key length of 128 bits (AES_128_XTS), or they are not encrypted (NO).

The default value is NO.

Note: Database encryption cannot be disabled.

KMSTARGET

```
KMSTARGET = string
```

This parameter specifies the key management system to be used if the database to be created is encrypted. See *Adabas Security Features* > *Adabas Encryption for Linux* > *Key Management System* in the Adabas documentation for all supported key management systems.

The default value is FILE.

NAME

```
NAME {=|:} string
```

This parameter specifies the name to be assigned to the database. This name will appear in the title of the database status report produced by the report utility ADAREP. If you specify an equals sign, the value given for 'string' will be converted to upper case; if you specify a colon, no upper-case conversion is performed.

A maximum of 16 characters may be specified.

If this parameter is omitted, a default value of 'GENERAL-DATABASE' is assigned.

SORTSIZE

```
SORTSIZE = (number[M] [,number[M]] ... )
```

This parameter specifies the number of megabytes to be assigned to the SORT dataset.

If the SORT dataset consists more than one extent, the size of each extent must be specified. Up to 50 extents can be specified. The SORT dataset can be formatted independently.

If a SORT container is to be used with an encrypted database, a dbid has to be specified. The encryption information stored in ASSO1 is read and the SORT container is encrypted accordingly. For the unencrypted case, a dbid may be give, but it is not necessary.

SYSFILES

```
SYSFILES = (number, number, number)
```

This parameter specifies the file numbers to be reserved for the Adabas system files. These file numbers must not be used subsequently for user files.

The first value specifies the file number of the checkpoint file.

The second value specifies the file number of the security file.

The third value specifies the file number of the user data file.

The default setting is SYSFILES=(1, 2, 3).

TEMPSIZE

```
TEMPSIZE = (number [M] [,number[M]] ... )
```

This parameter defines the number of megabytes to be assigned to TEMPx.

If the TEMP dataset is to be contained in more than one physical file, the size of each file must be specified.

This component may be formatted independently.

If a TEMP container is to be used with an encrypted database, you must specify a dbid. The encryption information stored in ASSO1 is read, and the TEMP container is encrypted accordingly. For the unencrypted case, a dbid may be given, but is not mandatory.

WORKBLOCKSIZE

```
WORKBLOCKSIZE = number[K]
```

This parameter specifies the block size that is to be used for the WORK file.

If no block size is specified, the default of 16K will be used.

WORKSIZE

```
WORKSIZE = number [B|M]
```

This parameter defines the number of blocks or megabytes to be assigned to WORK1.

If a 'B' is appended to the number, the size is in blocks, otherwise it is in megabytes.

Restart Considerations

ADAFRM does not have a restart capability. An interrupted ADAFRM run must be restarted from the beginning. Associator, Data Storage and WORK must be formatted together.

Control Statement Examples

Example: Formatting a database

```
adafrm: dbid = 1, name = DATABASE_1
adafrm: assosize = (200M, 100M), assoblocksize = (2K, 4K)
adafrm: datasize = (500M, 500M, 2M), datablocksize = (4K, 16k)
adafrm: worksize = 50M, workblocksize = 16K
```

A new database is created with the DBID 1 and the name "DATABASE_1". Two ASSO container files are created: ASSO1 has a size of 200 megabytes and a blocksize of 2 kilobytes, and ASSO2 has a size of 100 megabytes and a blocksize of 4 kilobytes. There are three DATA containers. DATA1 and DATA3 have a blocksize of 4 kilobytes, DATA2 has a blocksize of 16 kilobytes. There is a single WORK container file with a block size of 16 kilobytes. The file numbers 1, 2 and 3 will be used for the 3 system files.

Example: Formatting an encrypted database

```
adafrm: dbid = 2, name = DATABASE_2
adafrm: assosize = (200M, 100M), assoblocksize = (2K, 4K)
adafrm: datasize = (500M, 500M, 2M), datablocksize = (4K, 16k)
adafrm: worksize = 50M, workblocksize = 16K
adafrm: encryption = aes_256_xts
```

A new database is created with DBID 2, and the name "DATABASE_2", and with the same layout as in the example above. In this database, the ASSO and DATA container files are encrypted with algorithm AES_256_XTS. The encryption keys are created and managed by the Adabas file-based key management system (default: KMSTARGET = FILE).

Example: Formatting SORT and TEMP

```
adafrm: sortsize = (10M,10M)
adafrm: tempsize = 10M
```

Explanation: Two container files, each 10 megabytes in length, are to be formatted as SORT1 and SORT2. A container file, 10 megabytes in length, is to be formatted as TEMP1.

13 ADAINV (Creating, Removing And Verifying Inverted Lists)

- Functional Overview 162
- Procedure Flow 164
- Checkpoints 165
- Control Parameters 166
- Restart Considerations 176
- Examples 177

This chapter describes the utility "ADAINV".

Functional Overview

The inverted list utility ADAINV creates, removes, and verifies inverted lists for loaded files in a database. It does not require the Adabas nucleus to be active. The nucleus may, however, be active or shut down while ADAINV is running. The following functions are available:

- The INVERT function establishes new descriptors;
- The REINVERT function performs an implicit RELEASE and INVERT;
- The RELEASE function removes existing descriptors;
- The RESET_UQ function removes the unique status from descriptors;
- The SET_UQ function establishes a unique status for existing descriptors;
- The VERIFY function checks the integrity of inverted lists.

A LOB file can only be specified for the functions REINVERT, SUMMARY, and VERIFY.

These functions are mutually exclusive and only one of them may be executed each time this utility is run.

Notes:

1. When you perform ADAINV INVERT or REINVERT for a collation descriptor, the collation descriptor is always created using the highest ICU version supported (for Adabas Version 6.5: ICU 5.4, for older Adabas versions: ICU 3.2).
2. If you reinvert a collation descriptor created with ICU version 3.2 in order to upgrade to ICU version 5.4, there may be differences in the syntax or semantics of the collation specification. For example, with ICU version 3.2, the locale ‘fr’ implies the FRENCH option, while with ICU version 5.4, the FRENCH option must be specified explicitly. In such a case, instead of performing ADAINV REINVERT, first perform ADAINV RELEASE for the collation descriptor and then ADAINV INVERT with a new specification, which is equivalent to the old ICU version 3.2 specification.

When ADAINV is performed in online mode, Adabas users accessing the file may be active when ADAINV is running:

Function	ACC Users	UPD Users
INVERT	Yes	No
REINVERT	No	No
RELEASE	No	No
RESET_UQ	Yes	No
SET_UQ	Yes	No
SUMMARY	Yes	Yes
VERIFY	Yes	No

An Adabas user is an ACC user if he has only performed read or search operations for the file, or if he has executed an appropriate Open command. He is an UPD user if he has also performed insert, update or delete operations for the file or placed an ISN of the file in exclusive hold status, or if he has executed an appropriate open command.

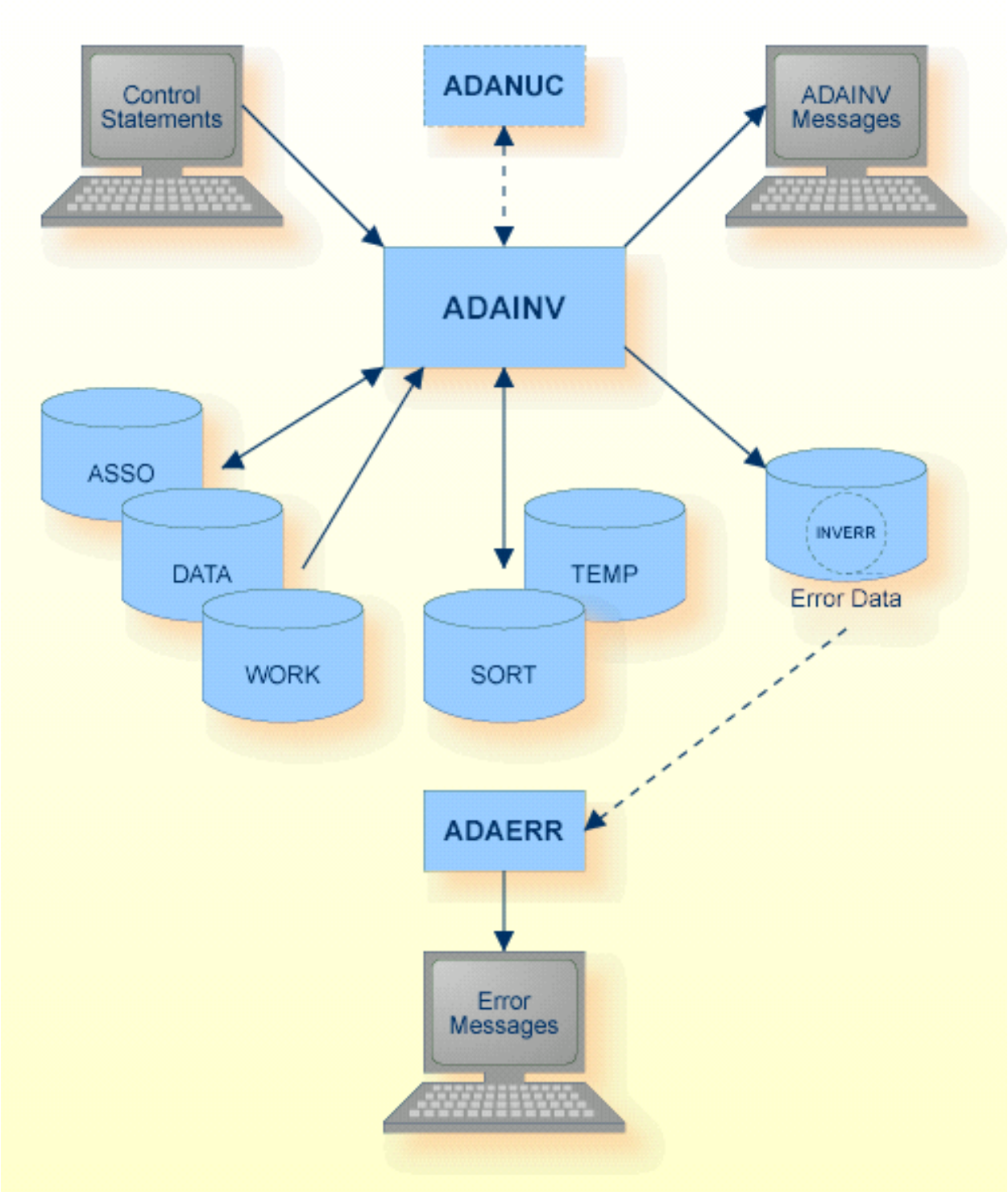
If there are users of the file that are not permitted, as defined in the table shown above, ADAINV fails with an error ADA048.

Notes:

1. An UPD user remains an UPD user after the end of the current transaction until the end of the user session.
2. You can determine which users are currently accessing a file with the ADAOPR DISPLAY=UQ_FILES command.

This utility is a single-function utility. For more information about single- and multi-function utilities, see *Adabas Basics*, *Using Utilities* in the Adabas documentation.

Procedure Flow



The sequential file INVERR can have multiple extents. For detailed information about sequential files with multiple extents, see *Adabas Basics*, *Using Utilities* in the Adabas documentation.

Data Set	Environment Variable / Logical Name	Storage Medium	Additional Information
Associator	ASSOx	Disk	
Data storage	DATAx	Disk	
Rejected data	INVERR	Disk (* see note)	output of ADAINV
Sort storage	SORTx TEMPLOCx	Disk	Administration Manual, temporary working space
Control statements	stdin		Utilities Manual
ADAINV messages	stdout		Messages and Codes
Temporary storage	TEMPx	Disk	
Work storage	WORK1	Disk	

Note: (*) A named pipe can be used for this sequential file.

In cases without an active nucleus and no pending AUTORESTART, the WORK may be used as TEMP by setting the environment variable/logical name TEMP1 to the path name of a WORK container.

Checkpoints

ADAINV handles checkpoints differently based on the status of the nucleus.

When the ADABAS nucleus is active, INVERT, RELEASE, SET_UQ and RESET_UQ are logged to the PLOG and SYNX checkpoints are set.

When the ADABAS nucleus is not active, all parameters except SUMMARY will write checkpoints.

The following table shows the nucleus requirements for each function and the checkpoints written:

Function	Nucleus must be active	Nucleus must NOT be active	Nucleus is NOT required	Checkpoint written	Nucleus operations allowed
INVERT			X	SYNP (offline) SYNX (online)	R
REINVERT		X(* see note)	X	SYNP	
RELEASE			X	SYNP (offline) SYNX (online)	R

Function	Nucleus must be active	Nucleus must NOT be active	Nucleus is NOT required	Checkpoint written	Nucleus operations allowed
RESET_UQ			X	SYNP (offline) SYNX (online)	R
SET_UQ			X	SYNP (offline) SYNX (online)	R
SUMMARY			X		W
VERIFY		X(* see note)	X	SYNX	R

Note: (*) When processing an Adabas system file.

R: read operations allowed for the processed file.

W: read and write operations allowed for the processed file.

Control Parameters

The following control parameters are available:

```
M   DBID = number

    INVERT = number,
            FIELDS {field_name [,UQ] [,TR] | derived_descriptor_definition | FDT},
             ... [END_OF_FIELDS]
             [,FDT]
D            [,LWP = number[K|M]]
D            [,UQ_CONFLICT = keyword]

D   [NO]LOWER_CASE_FIELD_NAMES

    REINVERT = number,
               {ALL_FIELDS | FIELDS {descriptor_name | FDT}, ...  [END_OF_FIELDS]}
               [,FDT]
D              [,[NO]FORMAT]
D              [,LWP = number[K|M]]
D              [,UQ_CONFLICT = keyword]

    RELEASE = number,
              {ALL_FIELDS | FIELDS {descriptor_name | FDT}, ...  [END_OF_FIELDS]}
              [,FDT]
D             [,[NO]FORMAT]

    RESET_UQ = number,
               {ALL_FIELDS | FIELDS {descriptor_name | FDT}, ...  [END_OF_FIELDS]}
```

```
                [,FDT]

    SET_UQ = number,
             {ALL_FIELDS | FIELDS {descriptor_name | FDT}, ...  [END_OF_FIELDS]}
             [,FDT]
D            [,UQ_CONFLICT = keyword]

    SUMMARY = number,
              {ALL_FIELDS | FIELDS
              {descriptor_name | derived_descriptor_definition | FDT},
              ...  [END_OF_FIELDS]}
              [,FDT]
D             [,FULL]

    VERIFY = number,
             {ALL_FIELDS | FIELDS {descriptor_name | FDT}, ...  [END_OF_FIELDS]}
D            [,ERRORS = number]
             [,FDT]
D            [,LWP = number[K|M]]
```

DBID

```
DBID = number
```

This parameter selects the database to be used.

INVERT

```
INVERT = number,
         FIELDS {field_name [,UQ] [,TR] | derived_descriptor_definition | FDT},
          ... [END_OF_FIELDS]
         [,FDT]
         [,LWP = number[K|M]]
         [,UQ_CONFLICT = keyword]
```

This function establishes new elementary, sub-, super-, hyper-, phonetic and collation descriptors at any time after a file has been initially loaded. 'number' specifies the file containing the fields to be inverted. You are not allowed to specify the number of a LOB file.

FDT

This parameter displays the FDT of the selected file. This option may be specified before or within the field specification list.

FIELDS {field_name [,UQ] [,TR] | derived_descriptor_definition | FDT}, ... [END_OF_FIELDS]

This parameter specifies fields to be inverted. It can contain one or more

- field name,
- phonetic descriptor or
- sub-, super-, hyper- or collation descriptor

specifications, each starting on a separate line. See *Adabas Basics*, *FDT Record Structure* in the Adabas documentation, for valid specifications of field names, phonetic, sub-, super-, hyper- or collation descriptors.

The options UQ and TR are used to specify whether the field in question is a unique descriptor or whether index truncation will be performed. See *Adabas Basics*, *Definition Options* for further information about the UQ and TR options.

Note: Only fields for which the values are stored in the base file can be used as descriptors or parent fields of derived descriptors. For this reason, an invert function will be aborted if a field to be inverted or a parent field of a derived descriptor to be created has the LA or LB option and values are stored in the LOB file. LA and LB fields can be used as descriptors or parent fields of derived descriptors, but then all values are limited to 16 KB – 3, and the base record including these LA or LB field values must fit into one data block.

If the field definitions are terminated with the END_OF_FIELDS parameter, this parameter must be specified in upper case when the LOWER_CASE_FIELD_NAMES parameter is used. In addition, the FDT parameter must also be specified in upper case when the LOWER_CASE_FIELD_NAMES parameter is used.

LWP = number[K|M]

For the sort of descriptor values, ADAINV uses a work pool in memory. The default size of the work pool in most cases results in an optimal performance for ADAINV. The LWP parameter allows you to increase the work pool; it defines the additional space added to the default work pool size in bytes, kilobytes (K) or megabytes (M).

Increasing the work pool size may be useful in the following cases:

- If you notice that in your environment the performance is better with a large work pool.
- If the SORT container is too small for sorting the descriptor values; an adequate LWP parameter can decrease the required size of the SORT container.

You can use the SUMMARY function to determine the required value for this parameter.

UQ_CONFLICT = keyword

This parameter determines which action is to be taken when duplicate values are found for a unique descriptor. 'keyword' may take the values ABORT or RESET. If ABORT is specified, ADAINV terminates execution and returns an error status if duplicate UQ descriptor values are found. If RESET is specified, the UQ status of the descriptors in question is removed and processing continues.

The default is UQ_CONFLICT = ABORT.

[NO]LOWER_CASE_FIELD_NAMES

```
[NO]LOWER_CASE_FIELD_NAMES
```

If LOWER_CASE_FIELD_NAMES is specified, Adabas field names are not converted to upper case. If NOLOWER_CASE_FIELD_NAMES is specified, Adabas field names are converted to upper case. The default is NOLOWER_CASE_FIELD_NAMES.

This parameter must be specified before the FIELDS parameter.

REINVERT

```
REINVERT = number,
           {ALL_FIELDS | FIELDS {descriptor_name | FDT}, ...  [END_OF_FIELDS]}
           [,FDT]
           [,[NO]FORMAT]
           [,LWP = number[K|M]]
           [,UQ_CONFLICT = keyword]
```

This function performs an implicit RELEASE and INVERT. This reduces the probability of a typing error, especially for sub- and superdescriptors.

Note: The purpose of ADAINV REINVERT is to recreate a descriptor if the index tree becomes unbalanced as a result of a large number of updates, or if an index error occurred. Descriptors are always recreated with the same definition as before; if you want to change the definition of a descriptor, for example a superdescriptor, you must perform ADAINV RELEASE followed by ADAINV INVERT with the new descriptor definition.

ALL_FIELDS

This parameter specifies that all descriptors of the selected file are to be released/inverted.

FDT

This parameter displays the FDT of the selected file. This option may be specified before or within the fields specification list.

FIELDS {descriptor_name | FDT}, ... [END_OF_FIELDS]

This parameter specifies the descriptors to be released/reinverted. It can be followed by one or more field names, each starting on a separate line. See *FDT Record Structure* in the Adabas documentation for a description of valid field name specifications.

If the field definitions are terminated with the END_OF_FIELDS parameter, this parameter must be specified in upper case when the LOWER_CASE_FIELD_NAMES parameter is used. In addition, the FDT parameter must also be specified in upper case when the LOWER_CASE_FIELD_NAMES parameter is used.

[NO]FORMAT

If a descriptor is released or reinverted, the new index created is generally smaller than the old index and requires less disk space. The FORMAT option can be used to format the blocks that are no longer used by the index but which are still allocated to the file.

The default is NOFORMAT.

LWP = number[K|M]

For the sort of descriptor values, ADAINV uses a work pool in memory. The default size of the work pool in most cases results in an optimal performance for ADAINV. The LWP parameter allows you to increase the work pool; it defines the additional space added to the default work pool size in bytes, kilobytes (K) or megabytes (M).

Increasing the work pool size may be useful in the following cases:

- If you notice that in your environment the performance is better with a large work pool.
- If the SORT container is too small for sorting the descriptor values; an adequate LWP parameter can decrease the required size of the SORT container.

UQ_CONFLICT = keyword

This parameter determines which action is to be taken when duplicate values are found for a unique descriptor. 'keyword' may take the values ABORT or RESET. If ABORT is specified, ADAINV terminates execution and returns an error status if duplicate UQ descriptor values are found. If RESET is specified, the UQ status of the descriptors in question is removed and processing continues.

The default is UQ_CONFLICT = ABORT.

RELEASE

```
RELEASE = number,
          {ALL_FIELDS | FIELDS {descriptor_name | FDT}, ...  [END_OF_FIELDS]}
          [,FDT]
          [,[NO]FORMAT]
```

This function removes elementary, sub-, super-, hyper-, phonetic and collation descriptors from the file specified by 'number'. You are not allowed to specify the number of a LOB file.

ALL_FIELDS

This parameter specifies that all descriptors of the selected file are to be released.

FDT

This parameter displays the FDT of the selected file. This option may be specified before or within the fields specification list.

FIELDS {descriptor_name | FDT}, ... [END_OF_FIELDS]

This parameter specifies the descriptors to be released. It can be followed by one or more field names, each starting on a separate line. See *FDT Record Structure* in the Adabas documentation for a description of valid field name specifications.

If the field definitions are terminated with the END_OF_FIELDS parameter, this parameter must be specified in upper case when the LOWER_CASE_FIELD_NAMES parameter is used. In addition, the FDT parameter must also be specified in upper case when the LOWER_CASE_FIELD_NAMES parameter is used.

[NO]FORMAT

If a descriptor is released or reinverted, the new index created is generally smaller than the old index and requires less disk space. The FORMAT option can be used to format the blocks that are no longer used by the index but which are still allocated to the file.

The default is NOFORMAT.

RESET_UQ

```
RESET_UQ = number,
           {ALL_FIELDS | FIELDS {descriptor_name | FDT}, ...  [END_OF_FIELDS]}
           [,FDT]
```

This function removes the unique status from elementary, sub-, hyper-, super- and collation descriptors defined in the file specified by 'number'. You are not allowed to specify the number of a LOB file.

ALL_FIELDS

This parameter specifies that the unique status is to be removed from all unique descriptors in the specified file.

FDT

This parameter displays the Field Definition Table (FDT) of the selected file. This option may be specified before or within the fields specification list.

FIELDS {descriptor_name | FDT}, ... [END_OF_FIELDS]

This parameter specifies the descriptors that are to have unique status removed. It can be followed by one or more field names, each starting on a separate line. See *FDT Record Structure* in the Adabas documentation for a description of valid field name specifications.

If the field definitions are terminated with the END_OF_FIELDS parameter, this parameter must be specified in upper case when the LOWER_CASE_FIELD_NAMES parameter is used. In addition, the FDT parameter must also be specified in upper case when the LOWER_CASE_FIELD_NAMES parameter is used.

SET_UQ

```
SET_UQ = number,
         {ALL_FIELDS | FIELDS {descriptor_name | FDT}, ...  [END_OF_FIELDS]}
         [,FDT]
         [,UQ_CONFLICT = keyword]
```

This function establishes the unique status for elementary, sub-, hyper-, super- and collation descriptors defined in the file specified by 'number'. You are not allowed to specify the number of a LOB file.

ALL_FIELDS

This parameter specifies that the unique status is to be established for all elementary, sub-, hyper-, super- and collation descriptors defined in the specified file.

FDT

This parameter displays the FDT of the selected file. This option may be specified before or within the fields specification list.

FIELDS {descriptor_name | FDT}, ... [END_OF_FIELDS]

This parameter specifies the descriptors for which the unique status is to be established. It can be followed by one or more field names, each starting on a separate line. See *FDT Record Structure* in the Adabas documentation for a description of valid field name specifications.

If the field definitions are terminated with the END_OF_FIELDS parameter, this parameter must be specified in upper case when the LOWER_CASE_FIELD_NAMES parameter is used. In addition, the FDT parameter must also be specified in upper case when the LOWER_CASE_FIELD_NAMES parameter is used.

UQ_CONFLICT = keyword

This parameter determines which action is to be taken when duplicate values are found for a unique descriptor. 'keyword' may take the values ABORT or RESET. If ABORT is specified, ADAINV terminates execution and returns an error status if duplicate descriptor values are found. If RESET is specified, the UQ status of the descriptors in question is not established and processing continues.

The default is UQ_CONFLICT = ABORT

SUMMARY

```
SUMMARY = number,
          {ALL_FIELDS | FIELDS
          {descriptor_name | derived_descriptor_definition | FDT},
           ...  [END_OF_FIELDS]}
          [,FDT]
          [,FULL]
```

This function displays the descriptor space summary (DSS) for the specified descriptors and the required sizes to process the descriptors.

Note: Processing the exact size would be too complicated. It may be that sizes a little smaller than those displayed are sufficient. If the file is updated during or after the SUMMARY function, the displayed values might also be too small.

See *Adabas Basics*, *Optimization of ADAMUP and ADAINV Processing* in the Adabas documentation for further information about ADAINV SUMMARY processing.

ALL_FIELDS

This parameter specifies that all descriptors of the selected files are to be checked.

FDT

This parameter displays the FDT of the selected file. This option may be specified before or within the fields specification list.

FIELDS {descriptor_name | derived_descriptor_definition | FDT}, ... [END_OF_FIELDS]

This parameter specifies the descriptors for which the unique status is to be established. It can be followed by one or more field names, phonetic descriptors, subdescriptors, superdescriptors, hyperdescriptors or collation descriptors, each starting on a separate line. You can specify fields that are descriptors or fields that are not descriptors. See *FDT Record Structure* in the Adabas documentation for a description of valid field name specifications.

If the field definitions are terminated with the END_OF_FIELDS parameter, this parameter must be specified in upper case when the LOWER_CASE_FIELD_NAMES parameter is used. In addition, the FDT parameter must also be specified in upper case when the LOWER_CASE_FIELD_NAMES parameter is used.

FULL

If this is specified, each descriptor is displayed along with the sizes that are required for the descriptor. This can be helpful if not all of the specified fields are to be processed.

VERIFY

```
VERIFY = number,
         {ALL_FIELDS | FIELDS {descriptor_name | FDT}, ...  [END_OF_FIELDS]}
         [,ERRORS = number]
         [,FDT]
         [,LWP = number[K|M]]
```

This function checks the integrity of inverted lists of the file specified by 'number'.

ALL_FIELDS

This parameter specifies that all descriptors of the selected file are to be checked.

ERRORS = number

This parameter specifies the number of errors that have to be reported in order to terminate the verification of a descriptor.

The default is 20.

FDT

This parameter displays the FDT of the selected file. This option may be specified before or within the fields specification list.

FIELDS {descriptor_name | FDT}, ... [END_OF_FIELDS]

This parameter specifies the descriptor fields to be verified. It can be followed by one or more field names, each starting on a separate line. See *FDT Record Structure* in the Adabas documentation for a description of valid field name specifications.

If the field definitions are terminated with the END_OF_FIELDS parameter, this parameter must be specified in upper case when the LOWER_CASE_FIELD_NAMES parameter is used. In addition, the FDT parameter must also be specified in upper case when the LOWER_CASE_FIELD_NAMES parameter is used.

LWP = number[K|M]

For the sort of descriptor values, ADAINV uses a work pool in memory. The default size of the work pool in most cases results in an optimal performance for ADAINV. The LWP parameter allows you to increase the work pool; it defines the additional space added to the default work pool size in bytes, kilobytes (K) or megabytes (M).

Increasing the work pool size may be useful in the following cases:

- If you notice that in your environment the performance is better with a large work pool.
- If the SORT container is too small for sorting the descriptor values; an adequate LWP parameter can decrease the required size of the SORT container.

You can use the SUMMARY function to determine the required value for this parameter.

Restart Considerations

ADAINV has no restart capability. However, it may or may not be possible to re-start an abnormally terminated ADAINV from the beginning.

If ADAINV terminates abnormally, it can usually be restarted from the beginning. However, if ADAINV has modified the index, the following points have to be considered:

- The function REINVERT ... FIELDS is the same as the function RELEASE ... FIELDS followed by the function INVERT ... FIELDS. So if ADAINV has aborted in the INVERT phase, perform the function INVERT ... FIELDS to restart the operation.
- If ADAINV is performed offline, there is a very small amount of time where a few records that together form a logical unit are written to disk. If ADAINV terminates after the first of these records has been written and before the last has been written, ADAINV cannot be restarted. In this case, the function REINVERT ... ALL_FIELDS is required. This cannot happen if ADAINV is performed online.
- If ADAINV terminates abnormally, it can happen that some index blocks are lost. These index blocks can only be recovered by the function REINVERT ... ALL_FIELDS or by using the utility ADAORD or by using the utilities ADAULD and ADAMUP.

Examples

Example 1

```
adainv: dbid=1
adainv: invert=10, fields
adainv: HO
```

The elementary field HO in file 10 of database 1 is inverted.

Example 2

```
adainv: dbid=1
adainv: invert=10
adainv: lwp=600k
adainv: fields
adainv: ph=phon(na)
adainv: sp=na(1,3),yy(1,2),uq
adainv: bb,uq
```

Three new descriptors are established for file 10 in database 1. PH is a phonetic descriptor based on the field NA. SP is a unique superdescriptor derived from bytes 1 to 3 of field NA and bytes 1 to 2 of field YY. The elementary field BB is changed to descriptor status and the unique flag is set. The size of the work pool to be used for the sort is increased to 600 K.

Example 3

```
adainv: dbid=1
adainv: release=10
adainv: fields
adainv: ho
adainv: ph
```

The two descriptors HO and PH from the examples above are released.

Example 4

```
adainv: dbid = 1, verify = 10
adainv: errors = 5
adainv: fields
adainv: sp
adainv: na
adainv: end_of_fields
```

The descriptors SP and NA are verified. The descriptor value table entries generated for descriptor NA are checked against the decompressed values of this field. Verification is terminated if more than five errors are reported for each descriptor.

Example 5

```
adainv: dbid = 1, reinvert = 10
adainv: fields
adainv: na
```

The descriptor NA in file 10 of database 1 is to be reinverted (this may be necessary if errors are reported in example 4).

Example 6

```
adainv: db=12
adainv: reinvert=9
adainv: all_fields
```

The complete index is recreated for file 9 in database 12.

The following output is produced:

```
%ADAINV-I-FILE, file 9, EMPLOYEES

%ADAINV-I-UIUPD, upper index being modified

%ADAINV-I-SORTDESC, sorting descriptor KA
%ADAINV-I-LOADDESC, loading descriptor KA

%ADAINV-I-SORTDESC, sorting descriptor S3
%ADAINV-I-LOADDESC, loading descriptor S3

%ADAINV-I-SORTDESC, sorting descriptor S2
%ADAINV-I-LOADDESC, loading descriptor S2

%ADAINV-I-SORTDESC, sorting descriptor PA
%ADAINV-I-LOADDESC, loading descriptor PA
```

```
%ADAINV-I-SORTDESC, sorting descriptor FB
%ADAINV-I-LOADDESC, loading descriptor FB

%ADAINV-I-SORTDESC, sorting descriptor AA
%ADAINV-I-LOADDESC, loading descriptor AA

%ADAINV-I-SORTDESC, sorting descriptor BC
%ADAINV-I-LOADDESC, loading descriptor BC

%ADAINV-I-SORTDESC, sorting descriptor CN
%ADAINV-I-LOADDESC, loading descriptor CN

%ADAINV-I-SORTDESC, sorting descriptor JA
%ADAINV-I-LOADDESC, loading descriptor JA

%ADAINV-I-SORTDESC, sorting descriptor H1
%ADAINV-I-LOADDESC, loading descriptor H1

%ADAINV-I-SORTDESC, sorting descriptor EA
%ADAINV-I-LOADDESC, loading descriptor EA

%ADAINV-I-SORTDESC, sorting descriptor LC
%ADAINV-I-LOADDESC, loading descriptor LC

%ADAINV-I-SORTDESC, sorting descriptor S1
%ADAINV-I-LOADDESC, loading descriptor S1

%ADAINV-I-SORTDESC, sorting descriptor AC
%ADAINV-I-LOADDESC, loading descriptor AC

%ADAINV-I-NULLDESC, no values for descriptor IJ
%ADAINV-I-LOADDESC, loading descriptor IJ

%ADAINV-I-NULLDESC, no values for descriptor IB
%ADAINV-I-LOADDESC, loading descriptor IB

%ADAINV-I-NULLDESC, no values for descriptor FI
%ADAINV-I-LOADDESC, loading descriptor FI

%ADAINV-I-UIUPD, upper index being modified
%ADAINV-I-DSPASSES, data storage passes : 17
%ADAINV-I-REMOVED, dataset SORT1, file C:\Program Files\Software AG\Adabas/db012
\SORT01_3664.012 removed
%ADAINV-I-IOCNT,          1 IOs on dataset SORT
%ADAINV-I-IOCNT,         85 IOs on dataset DATA
%ADAINV-I-IOCNT,         49 IOs on dataset ASS0
```

Notes:

1. The message NULLDESC indicates that no descriptor values exist for this descriptor. This may happen for fields defined with option NU or NC if the field contains the null value/SQL null values for all records.
2. The message DSPASSES shows how often the data records of the file were read. In this case the number of data storage passes is 17, i.e. the data records were reread for each descriptor, because no TEMP container was defined where descriptor values can be saved. The number of data storage passes can be reduced by defining a TEMP container. This is recommended in particular for large files, because it reduces the number of required I/O operations significantly. The ADAINV parameter SUMMARY can be used to find out which size is useful for the TEMP container.
3. The message REMOVED shows that a temporary SORT container created by ADAINV was deleted. You can also use a persistent SORT container, which is not created and deleted by ADAINV (see **ADAFRM** for further details).

Example 7

```
adainv: dbid = 1, set_uq=10
adainv: fields
adainv: na
adainv: end_of_fields
adainv: uq_conflict=reset
```

The unique status is to be established for the descriptor NA in file 10 of database 1. If there is more than one ISN per descriptor value, the conflicting ISNs are written to the error log and the unique status is removed.

Example 8

```
adainv: dbid = 1, reset_uq=10
adainv: fields
adainv: sp
```

The unique status is to be removed from the descriptor SP in file 10 of database 1.

Example 9

```
adainv: db=33
adainv: summary=112
adainv: fields
adainv: ab
adainv: ae
adainv: s1=ap(1,1),aq(1,1),ar(1,1)
adainv: s2=ac(1,3),ad(1,8),ae(1,9)
adainv: s3=ao(2,3)
```

This produces the following output:

```
Descriptor summary:
===================

Descriptor AB :           1,194,469 bytes,             581,209 occ
Descriptor AE :           3,605,545 bytes,             538,769 occ
Descriptor S1 :           1,566,501 bytes,             581,209 occ
Descriptor S2 :           1,520,169 bytes,              72,389 occ
Descriptor S3 :           1,340,949 bytes,             446,983 occ


Required sizes to process these descriptors:
============================================
           - SORTSIZE (LWP=          0 KB)        =          8 MB
           - LWP for incore sort                  =     13,230 KB
           - TEMPSIZE (1 pass)                    =         24 MB
           - TEMPSIZE (2 passes)                  =         13 MB
           - TEMPSIZE (recommended minimum size)  =          5 MB

%ADAINV-I-IOCNT,     1710 IOs on dataset DATA
%ADAINV-I-IOCNT,        3 IOs on dataset ASSO
%ADAINV-I-TERMINATED,   24-NOV-2006 14:15:06, elapsed time: 00:04:03
```

14 ADAKMS (Key Management Service)

- Functional Overview 184
- Procedure Flow 185
- Checkpoints 185
- Control Parameters 186

This chapter describes the "ADAKMS" utility.

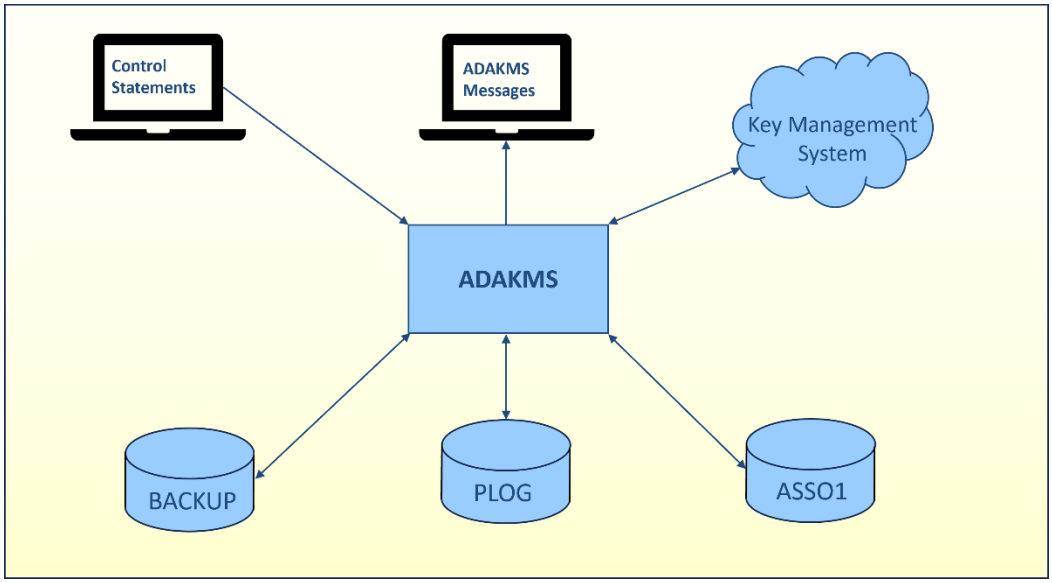
Functional Overview

The ADAKMS administration utility helps to manage keys for encrypted Adabas assets. It can display, create, delete and undelete keys, and perform key rotation on encrypted databases as well as on backup save sets and protection log files (PLOG).

The database must be offline for key rotation.

This utility is a multi-function utility. For more information about single- and multi-function utilities, see *Adabas Basics > Using Utilities*.

Procedure Flow



Data Set	Environment Variable / Logical Name	Storage Medium	Additional Information
Database configuration		DBnnn.INI	
Associator	ASSO1	Disk	
Control statements	stdin		See the *Control Parameters* section
ADAKMS messages	stdout		Messages and Codes
PLOG		Disk	
Backup		Disk	

Checkpoints

The utility writes no checkpoints.

Control Parameters

The following control parameters are available:

```
     CREATE = keyword, DESCRIPTION = string, ALIAS = string

     DELETE = (string[,string]...)

     DISPLAY = keyword

     ROTATE= keyword, {DBID=number | FILENAME=(string[,string]...), KEYNAME=string}

M    KMSTARGET = keyword

     UNDELETE = (string[,string]...)
          ↩
```

CREATE

```
CREATE=keyword, description=string, alias=string
```

This function creates a key encryption key (KEK) in the desired key management system. The keyword can take the KEY value .

DESCRIPTION=string
A string describing the key. The string can be up to 256 characters.

ALIAS=string
A smaller string up to 32 characters used as an alternative for the key name which might be difficult to read.

If the utility is being used non-interactively and strings contain spaces, the string must be surrounded by quotes. For interactive usage where strings contain spaces, each command must be entered in a separate line, and the strings must not be surrounded by quotes. For more information, see *Using Utilities*.

Example

Non-interactive:

```
$> adakms kmstarget=aws create=key description=”My new key” alias=KEY_1
```

Interactive:

```
$> adakms
adakms: kmstarget=aws
adakms: create=key
adakms: description=My new key
adakms: alias=KEY_1
```

DELETE

```
DELETE=(string[,string]...)
```

This function removes a key from the key management system. The key will not be deleted immediately. It is marked as deleted and will only be deleted after a key management system dependent period.

A maximum of 5 keys can be deleted in one command.

Example

```
adakms: display=keys
            NO STATE KEY NAME                                                ↩
    ALIAS                DESCRIPTION
           0 Act   1773647932CC6F9763449248C321E00EC360EB04F924EEEEAB62CBA55BB8C10F ↩
KEK_DB200            KEK created at: 16-MAR-2026 08:58:52
           1 Act   177364865014F6F88735D694F9BFF4A0BEB55BABE87ECC1A181AAFD965D07FE5 ↩
KEK_DB210            KEK created at: 16-MAR-2026 09:10:50
           2 Act   177364869273FB5449AF532517D57B314A31348E5FF6FF0664F00B228790B157 ↩
KEK_DB202            KEK created at: 16-MAR-2026 09:11:32
adakms: delete=177364865014F6F88735D694F9BFF4A0BEB55BABE87ECC1A181AAFD965D07FE5
%ADAKMS-I-KEYDEL, Key ↩
177364865014F6F88735D694F9BFF4A0BEB55BABE87ECC1A181AAFD965D07FE5 deleted
adakms: display=keys
            NO STATE KEY NAME                                                ↩
    ALIAS                DESCRIPTION
           0 Act   1773647932CC6F9763449248C321E00EC360EB04F924EEEEAB62CBA55BB8C10F ↩
KEK_DB200            KEK created at: 16-MAR-2026 08:58:52
           2 Act   177364869273FB5449AF532517D57B314A31348E5FF6FF0664F00B228790B157 ↩
KEK_DB202            KEK created at: 16-MAR-2026 09:11:32
adakms:
```

DISPLAY

```
DISPLAY=keyword
```

This function lists all keys from the key management system. The keyword can take the KEYS value for displaying all active keys, and KEYS_FULL value for displaying all keys including the deleted keys. Active keys are shown with the 'Act' state, while deleted keys are shown with the 'Del' state.

Example 1

Display only active keys.

```
adakms: display=keys

     NO STATE KEY NAME                                                  ALIAS ↩
               DESCRIPTION
      0 Act   1773647932CC6F9763449248C321E00EC360EB04F924EEEEAB62CBA55BB8C10F ↩
KEK_DB200            KEK created at: 16-MAR-2026 08:58:52
      2 Act   177364869273FB5449AF532517D57B314A31348E5FF6FF0664F00B228790B157 ↩
KEK_DB202            KEK created at: 16-MAR-2026 09:11:32
```

Example 2

Display all keys including deleted keys.

```
adakms: display=keys_full

     NO STATE KEY NAME                                                  ALIAS ↩
               DESCRIPTION
      0 Act   1773647932CC6F9763449248C321E00EC360EB04F924EEEEAB62CBA55BB8C10F ↩
KEK_DB200            KEK created at: 16-MAR-2026 08:58:52
      1 Del   177364865014F6F88735D694F9BFF4A0BEB55BABE87ECC1A181AAFD965D07FE5 ↩
KEK_DB210            KEK created at: 16-MAR-2026 09:10:50
      2 Act   177364869273FB5449AF532517D57B314A31348E5FF6FF0664F00B228790B157 ↩
KEK_DB202            KEK created at: 16-MAR-2026 09:11:32
adakms:
```

ROTATE

```
ROTATE=keyword, {DBID=number | FILENAME=(string[,string]...), KEYNAME=string}
```

This function will encrypt the data encryption key (DEK) with a new key encryption key (KEK) from the key management system. It first decrypts the DEK with the former KEK, encrypts the DEK with a new KEK, and then writes it back to either the database, backup save set, or protection log file.

The keyword can take the DATABASE value for rotating the key in an existing encrypted database, or the FILE value for rotating in sequential files. Only backup save sets and protection log files are supported; all other files will be rejected.

Key rotation does not mean to encrypt the database or sequential file with a new key, only the data encryption key will be encrypted with the new key. See also *Adabas Security Features* > *Adabas Encryption for Linux* > *Key Management System*.

The ADAKMS utility uses a temporary file for key rotation in sequential files in the same directory as the original file. The file name is constructed from the original file name with a suffix like '.XXXXXX', where 'X' can be any character. A correspondingly large amount of space is needed in this directory.

The key rotation can take the new key encryption key from a different key management system. It will first decrypt the data encryption key using the key management system stored in the database or file, then encrypt it with the key specified from the key management system specified by the `KMSTARGET` command.

DBID=number
Selects the database for the key rotation. The database must be offline.

FILENAME=(string[,string]…)
Selects the sequential files for key rotation. Up to 100 files can be specified for key rotation in one command and the files can be a mixture of backup save sets and protection log files.

Note: Wildcards are not supported.

Example 1

Rotate the key in a database.

```
$> adarep dbid=200 sum
…
ENCRYPTION                        AES_256_XTS
KMSTARGET                         FILE
KEKNAME                           ↩
1773647932CC6F9763449248C321E00EC360EB04F924EEEEAB62CBA55BB8C10F
…

adakms: rotate=database, dbid=200, ↩
keyname=177364865014F6F88735D694F9BFF4A0BEB55BABE87ECC1A181AAFD965D07FE5
%ADAKMS-I-DBOFF, database 200 accessed offline
%ADAKMS-I-KEYROTSUCC, Successfully exchanged encryption with new key: ↩
177364865014F6F88735D694F9BFF4A0BEB55BABE87ECC1A181AAFD965D07FE5
%ADAKMS-I-IOCNT, 4 IOs on dataset ASS0

$> adarep dbid=200 sum
…
ENCRYPTION                        AES_256_XTS
```

```
KMSTARGET                        FILE
KEKNAME                          ↩
177364865014F6F88735D694F9BFF4A0BEB55BABE87ECC1A181AAFD965D07FE5
…
```

Example 2

Rotate the key in a sequential file.

```
$> export BCK001=BCK200.ENC
$> adabck sum
…
ENCRYPTION                       AES_256_XTS
KMSTARGET                        FILE
KEKNAME                          ↩
1773647932CC6F9763449248C321E00EC360EB04F924EEEEAB62CBA55BB8C10F
…

adakms: rotate=file, filename=BCK200.ENC, ↩
keyname=177364865014F6F88735D694F9BFF4A0BEB55BABE87ECC1A181AAFD965D07FE5
%ADAKMS-I-KEYROTSUCC, Successfully exchanged encryption with new key: ↩
177364865014F6F88735D694F9BFF4A0BEB55BABE87ECC1A181AAFD965D07FE5
%ADAKMS-I-IOCNT, 178 IOs on dataset BCK200.ENC

$> adabck sum
…
ENCRYPTION                       AES_256_XTS
KMSTARGET                        FILE
KEKNAME                          ↩
177364865014F6F88735D694F9BFF4A0BEB55BABE87ECC1A181AAFD965D07FE5
…
```

Example 3

Rotate the key in a database with the new key from a different key management system.

```
$> adarep dbid=200 sum
…
ENCRYPTION                       AES_256_XTS
KMSTARGET                        FILE
KEKNAME                          ↩
1773647932CC6F9763449248C321E00EC360EB04F924EEEEAB62CBA55BB8C10F
…

adakms: kmstarget=aws
adakms: rotate=database
adakms: dbid=200
adakms: keyname=ad6e4b57-5bd7-4613-9879-310d89e42487
%ADAKMS-I-DBOFF, database 200 accessed offline
%ADAKMS-I-KEYROTSUCC, Successfully exchanged encryption with new key: ↩
ad6e4b57-5bd7-4613-9879-310d89e42487
```

```
%ADAKMS-I-IOCNT, 4 IOs on dataset ASSO

$> adarep dbid=200 sum
…
ENCRYPTION                        AES_256_XTS
KMSTARGET                         AWS
KEKNAME                           ad6e4b57-5bd7-4613-9879-310d89e42487
…
```

KMSTARGET

```
KMSTARGET=keyword
```

This function loads the plugin library for the desired key management system and it is mandatory to be entered as the first command. The keyword can take the values for the key management system:

- AWS for the Amazon Web Services.
- AZURE for the Azure Cloud.
- GCP for the Google Cloud Platform.
- FILE for the Adabas internal file system based key management.

For interactive usage of the utility, the `KMSTARGET` command does not need to be repeated for subsequent commands. The same target will be used unless it is changed by a new `KMSTARGET` command.

UNDELETE

```
UNDELETE=(string[,string]...)
```

This function reverts a deletion of a key from the key management system. This is only possible for keys that are in the deleted state but not already removed.

A maximum of 5 keys can be undeleted in one command.

Example

```
adakms: display=keys_full
     NO STATE KEY NAME                                                  ALIAS ↩
               DESCRIPTION
      0 Act   1773647932CC6F9763449248C321E00EC360EB04F924EEEEAB62CBA55BB8C10F ↩
KEK_DB200            KEK created at: 16-MAR-2026 08:58:52
      1 Del   177364865014F6F88735D694F9BFF4A0BEB55BABE87ECC1A181AAFD965D07FE5 ↩
KEK_DB210            KEK created at: 16-MAR-2026 09:10:50
      2 Act   177364869273FB5449AF532517D57B314A31348E5FF6FF0664F00B228790B157 ↩
KEK_DB202            KEK created at: 16-MAR-2026 09:11:32
adakms: undelete=177364865014F6F88735D694F9BFF4A0BEB55BABE87ECC1A181AAFD965D07FE5
%ADAKMS-I-KEYUNDEL, Key ↩
```

```
177364865014F6F88735D694F9BFF4A0BEB55BABE87ECC1A181AAFD965D07FE5 undeleted
adakms: display=keys
     NO STATE KEY NAME                                                  ALIAS ↩
               DESCRIPTION
      0 Act   1773647932CC6F9763449248C321E00EC360EB04F924EEEEAB62CBA55BB8C10F ↩
KEK_DB200            KEK created at: 16-MAR-2026 08:58:52
      1 Act   177364865014F6F88735D694F9BFF4A0BEB55BABE87ECC1A181AAFD965D07FE5 ↩
KEK_DB210            KEK created at: 16-MAR-2026 09:10:50
      2 Act   177364869273FB5449AF532517D57B314A31348E5FF6FF0664F00B228790B157 ↩
KEK_DB202            KEK created at: 16-MAR-2026 09:11:32
```

15 ADAMUP (Mass Add And Delete)

- Functional Overview 194
- Procedure Flow 195
- Checkpoints 198
- Control Parameters 199
- Restart Considerations 205
- SORT Data Set Placement 206
- TEMP Data Set Placement 206
- Examples 206

This chapter describes the utility "ADAMUP".

Functional Overview

The mass update utility ADAMUP adds records to, or deletes records from a file in a database. It does not require the Adabas nucleus to be active.

The output files produced by the compression utility ADACMP or the unload utility ADAULD may be used as input for a mass add.

Note: The ADAMUP ADD function can process MUPDTA/MUPDVT files created with earlier Adabas versions, but not MUPDTA/MUPDVT files created with later Adabas versions.

Input files produced by ADACMP or ADAULD with the SINGLE_FILE option or from a previous run of ADAMUP using the DELETE function with the LOG option can also be used.

Input files produced without descriptor value tables (SHORT option in ADAULD or LOG=SHORT option in ADAMUP) can be processed if the database file to be processed does not contain any descriptors.

The input for the DELETE function is provided in an input file. Each record contains one or more ISNs or ISN ranges.

Records may be both added to and deleted from a database file during a single run of ADAMUP.

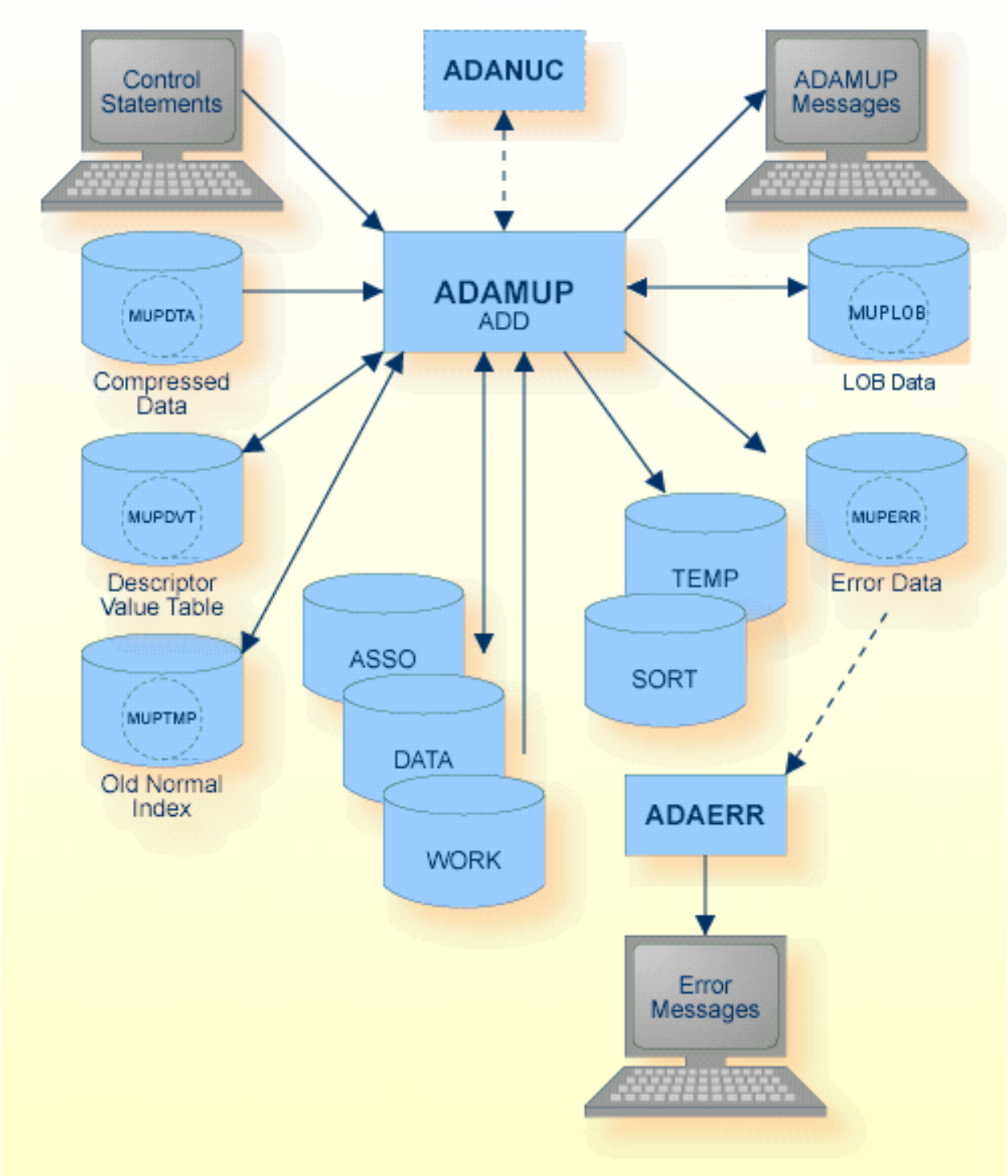
The fcb_file_update_time field in the file information is updated during ADD or DELETE operations performed by ADAMUP.

If the utility writes records to the error file, it will exit with a non-zero status.

Note: FDTs that contain the same fields, but collation descriptors that belong to different ICU versions are considered to be different. This means you can only load the data into a file with a different ICU version if the file is empty, and if you use the NEW_FDT parameter.

This utility is a single-function utility. For more information about single- and multi-function utilities, see *Adabas Basics*, *Using Utilities* in the Adabas documentation.

Procedure Flow

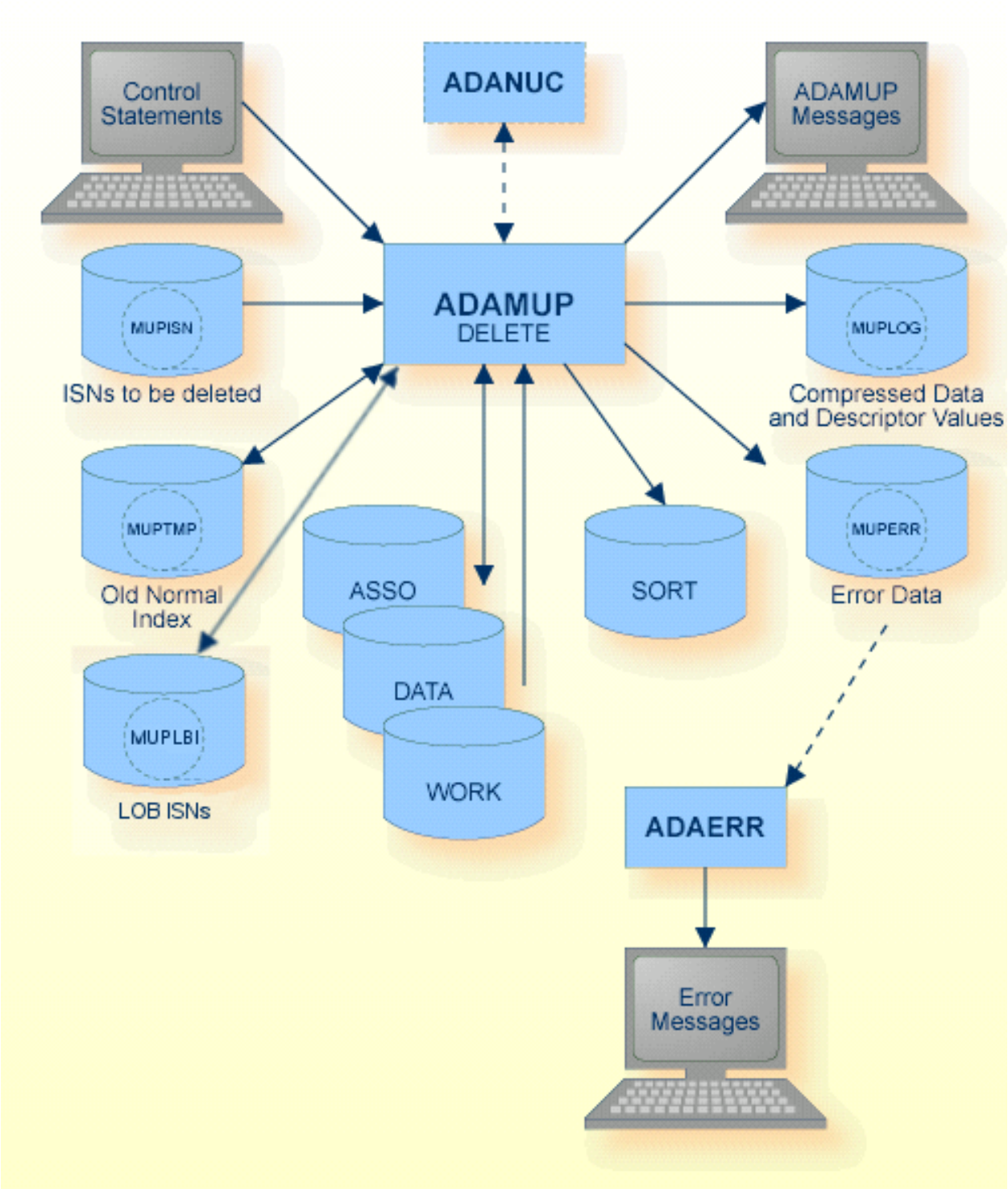


The sequential files MUPDTA, MUPDVT, MUPTMP, MUPLOB and MUPERR can have multiple extents. For detailed information about sequential files with multiple extents, see *Adabas Basics, Using Utilities* in the Adabas documentation.

Data Set	Environment Variable / Logical Name	Storage Medium	Additional Information
Associator	ASSOx	Disk	
Data storage	DATAx	Disk	
Compressed input data	MUPDTA	Disk	
Descriptor values	MUPDVT	Disk	
Rejected data	MUPERR	Disk (* see note)	
LOB data	MUPLOB	Disk	Temporary working space, will be deleted again when ADAMUP terminates
Normal index	MUPTMP	Disk	Temporary working space, will be deleted again when ADAMUP terminates
Sort storage	SORTx TEMPLOCx	Disk	Administration Manual, temporary working space
Control statements	stdin		Utilities Manual
ADAMUP messages	stdout		Messages and Codes
Temporary storage	TEMPx	Disk	
Work	WORK1	Disk	

Note: (*) A named pipe can be used for this sequential file.

In cases without an active nucleus and no pending AUTORESTART, the WORK may be used as TEMP by setting the environment variable/logical name TEMP1 to the same value as WORK1.



The sequential files MUPTMP, MUPLBI, MUPLOG and MUPERR can have multiple extents.

Data Set	Environment Variable / Logical Name	Storage Medium	Additional Information
Associator	ASSOx	Disk	
Data storage	DATAx	Disk	
Rejected data	MUPERR	Disk (* see note)	
ISNs to be deleted	MUPISN	Disk	
LOB ISNs	MUPLBI	Disk	Temporary working space, will be deleted again when ADAMUP terminates
Compressed data	MUPLOG	Disk	
Normal index	MUPTMP	Disk	Temporary working space, will be deleted again when ADAMUP terminates
Sort storage	SORTx TEMPLOCx	Disk	Administration Manual, temporary working space
Control statements	stdin		Utilities Manual
ADAMUP messages	stdout		Messages and Codes
Work	WORK1	Disk	

Note: (*) A named pipe can be used for this sequential file.

Checkpoints

The following table shows the nucleus requirements for each function and the checkpoints written:

Function	Nucleus must be active	Nucleus must NOT be active	Nucleus is NOT required	Checkpoint written
FDT			X	-
UPDATE	X(* see note 1)	X(* see note 2)	X(* see note 3)	SYNP
SUMMARY			X	-

Notes:

1. When deleting records in a file with LOB data.
2. When updating an Adabas system file.
3. Except when deleting records in a file with LOB data.

Control Parameters

The following control parameters are available:

```
M    DBID = number

     FDT

     SUMMARY

     UPDATE = number [,FDT]
              [ADD [,add_keywords]]
              [DELETE [,delete_keywords]]
 D            [,[NO]FORMAT]
 D            [LWP = number[K|M]]
```

DBID

```
DBID = number
```

This parameter selects the database to be used.

FDT

```
FDT
```

This parameter displays the Field Definition Table (FDT) of the selected file in the database. If records are to be added to a file, the FDT of the sequential input file containing these records can also be displayed. This parameter may also be used in an ADD/DELETE specification.

Depending on the context in which the FDT parameter is used, the Field Definition Table contained in the sequential input file MUPDTA and/or the Field Definition Table contained in the selected database file is displayed.

Examples

```
adamup db=2 fdt update=11 add ↩
```

The FDT stored in the MUPDTA file is displayed.

```
adamup db=2 update=11,fdt add ↩
```

The FDT of file 11 in database 2 is displayed.

```
adamup db=2 fdt update=11,fdt add ↩
```

The FDT stored in the MUPDTA file is displayed first; then the FDT of file 11 in database 2 is displayed.

SUMMARY

```
SUMMARY
```

This parameter displays the Descriptor Space Summary (DSS) on the sequential input file that contains the compressed records. This display is identical to the one at the end of the ADACMP, ADAULD or ADAMUP run which generated this input file, and can be used to estimate the space required in the index.

Additionally, the following information is displayed:

- required SORT size (for default LWP)
- recommended TEMP size (the size required to do the index update in one pass)
- current size of SORT (if present)
- LWP needed for memory-resident sort
- Recommended size of LWP and SORT (if LWP is large enough to allow a smaller SORT size to be used).

Note: If the default LWP is large enough to do a memory-resident sort, SORT sizes are not displayed.

UPDATE

```
UPDATE  = number [,FDT]
              [ADD [,add_keywords]]
              [DELETE [,delete_keywords]]
              [,[NO]FORMAT]
              [LWP = number[K|M]]
```

This function specifies the file to which records are to be added/deleted. Since ADAMUP requires exclusive control of the file, it cannot be used for an Adabas system file while the nucleus is active. You are not allowed to specify a LOB file.

ADD

```
ADD
   [,DE_MATCH = keyword]
   [,FDT]
   [,[NO]NEW_FDT]
   [,NUMREC = number]
   [,SKIPREC = number]
   [,UQ_CONFLICT = keyword]
   [,RI_CONFLICT = keyword]
   [,[NO]USERISN]
```

This parameter indicates that records are to be added to the file specified by the UPDATE parameter.

The input for mass add is produced by the compression utility ADACMP, the unload utility ADAULD or by a previous run of the mass update utility ADAMUP using the DELETE function with the LOG option set.

ADAMUP compares the FDT in the sequential input file that contains the compressed records with the FDT of the database file specified. The FDTs must have identical layouts and must use the same field names, formats, lengths and options.

Descriptors in the database file can be a subset of the descriptors defined in the FDT in the sequential input file, but the input file must contain descriptor value table (DVT) entries for all descriptors defined in the database file. Therefore, input files produced without descriptor value tables (SHORT option) can only be processed if there are no descriptors currently defined in the database file to be updated.

If the input for mass update contains LOB data, the Adabas file must have an assigned LOB file.

DE_MATCH = keyword

This parameter is used to indicate which action is to be taken if a descriptor provided with the input data is not a descriptor in the actual FDT of the file. If keyword = IDENTICAL, ADAMUP terminates processing and returns an error message. If keyword = SUBSET, ADAMUP ignores a descriptor which is in the input file, but which has been removed from the database file.

The default is DE_MATCH=IDENTICAL.

[NO]NEW_FDT

If NEW_FDT is specified, the FDT of the file is replaced by the FDT of the MUPDTA file. NEW_FDT can only be specified if the file is empty when ADAMUP is started.

NEW_FDT must be specified if the FDT of the file in the database and the FDT of the MUPDTA file are different - a mass update is not possible if the FDTs are different and the file is not empty.

The default is NONEW_FDT.

NUMREC = number

This parameter specifies the number of records to be added. If NUMREC is specified, ADAMUP terminates after adding the predefined number of records, unless an end-of-file condition on the input file causes ADAMUP processing to end. If NUMREC is omitted and SKIPREC is not specified, all records in the input file are added.

SKIPREC = number

This parameter specifies the number of records in the input file to be skipped before starting to add records.

UQ_CONFLICT = keyword

This parameter is used to indicate which action is to be taken if duplicate values are found for a unique descriptor. 'keyword' may take the values ABORT or RESET. If ABORT is specified, ADAMUP terminates execution and returns an error status if duplicate UQ descriptor values are found. If RESET is specified, conflicting ISNs are written to the error log, the UQ status of the descriptors in question is removed and processing continues.

The default is UQ_CONFLICT=ABORT.

RI_CONFLICT

This parameter is used to indicate which action is to be taken if referential integrity is violated. 'keyword' may take the values ABORT or RESET. If ABORT is specified, ADAMUP terminates execution and returns an error status. The index is marked as not accessible. If RESET is specified, the violated constraint is removed. In both cases the violating ISNs are stored in the error log.

The default is RI_CONFLICT=ABORT.

[NO]USERISN

This option indicates whether the ISN to be assigned to each record is to be taken from the input file or not.

This option should be set to USERISN if the user wants to control ISN assignment for each record added to the database file. Each ISN provided must be:

- a four-byte binary number immediately preceding each data record;
- within the current limit (MAXISN) for the file - the file's Address Converter is not automatically extended;
- unique within the specified file.

Otherwise ADAMUP terminates execution and returns an error message.

Note that problems could arise if this option is set to USERISN for an input file created by an unload that is based on a descriptor which is a multiple-value field. This is because the same record may have been unloaded more than once. Please refer to the ADAULD utility, SORTSEQ parameter for more information.

If this option is set to NOUSERISN, the ISN for each record is assigned by ADAMUP. However, the ISN of a DVT record that has been previously re-vectored by a hyperexit will not be changed by ADAMUP.

The default is NOUSERISN.

DELETE

```
DELETE
      [,DATA_FORMAT = keyword]
      [,FDT]
      [,ISN_NOT_PRESENT = keyword]
      [,LOG = keyword | ,NOLOG]
```

This parameter indicates that records are to be deleted from the file specified by the UPDATE parameter. The ISNs of the records to be deleted are given in an input file.

DATA_FORMAT = keyword

This parameter defines the data type of the records in the input file containing the ISNs to be deleted. Each record contains one or more ISNs or ISN ranges.

Valid ISNs are within the range 1...MAXISN.

In accordance with the formats supported, 'keyword' may take the following values:

Keyword	Meaning
BINARY	A single ISN is contained in a 4 byte binary value, an ISN range is contained in two consecutive binary values, with the high-order bit set in the second value. Blocks in this file start with 2 byte exclusive length field. **Note:** ISNs >= 2**31 (2147483648) cannot be deleted with DATA_FORMAT=BINARY.
DECIMAL	Each record has the following layout: `[number[-number] [,number[-number]]...] [;comment]` where 'number' is decimal number with 1 to 10 digits.

ADAMUP validates all input records in a first step. ADAMUP displays the line number and the offset for each error that is detected. If an error is detected, ADAMUP terminates execution once the input file has been completely parsed.

The default is DATA_FORMAT = BINARY.

ISN_NOT_PRESENT = keyword

This parameter indicates the action to be taken when an ISN given in the input file of records to be deleted is:

- not within the current limit (MAXISN) for the file;
- not in the file's Address Converter.

'keyword' may take the following values:

Keyword	Meaning
ABORT	ADAMUP aborts execution and returns an error message if a conflicting ISN is detected.
IGNORE	ADAMUP writes the conflicting ISNs to the error log and continues processing.

The default is ISN_NOT_PRESENT=IGNORE

LOG = keyword NOLOG

LOG=keyword indicates that the deleted records are logged in a sequential file. The records are written in compressed format and are identical to those produced by the compression utility ADACMP and the unload utility ADAULD. Because each data record is preceded by its ISN, these ISNs can be used as user ISNs when reloading or mass-adding this file (see the USERISN option described above).

'keyword' may take the following values:

Keyword	Meaning
FULL	The descriptor values which are required to build the index, are included in the output file.
SHORT	The descriptor values which are required to build the index, are omitted from the output file.

ADAMUP writes both the compressed data records and the descriptor values generated to a single file.

The default is NOLOG.

[NO]FORMAT

This option may be used to format blocks at the end of the file's Normal Index (NI) and Upper Index (UI) extents if the new index (after the modifications have been made) requires less space than the old index did. This may be the result of deletions within the index, recovery of lost index blocks or re-establishing the padding factor.

Because these blocks are returned to the file's unused blocks, there are no side-effects if the data stored in these blocks is not deleted. If this option is set to FORMAT, ADAMUP overwrites these blocks with binary zeros.

The default is NOFORMAT.

LWP = number[K|M]

For the sort of descriptor values, ADAMUP uses a work pool in memory. The default size of the work pool in most cases results in an optimal performance for ADAMUP. The LWP parameter allows you to increase the work pool; it defines the additional space added to the default work pool size in bytes, kilobytes (K) or megabytes (M).

Increasing the work pool size may be useful in the following cases:

- If you notice that in your environment the performance is better with a large work pool.
- If the SORT container is too small for sorting the descriptor values; an adequate LWP parameter can decrease the required size of the SORT container.

You can use the SUMMARY function to determine the required value for this parameter.

Restart Considerations

ADAMUP has no restart capability. An abnormally terminated ADAMUP must be rerun from the beginning.

If the Data Storage space becomes exhausted, ADAMUP will not abort, but will attempt to build the index for the records that have already been loaded; this means that the file is consistent, and the remaining records can then be loaded with the SKIPREC option after additional Data Storage space has been allocated.

SORT Data Set Placement

It is recommended that the SORT data set does not reside on the same volume as the Associator and the input file that contains the Descriptor Value Tables.

The SORT data set may be omitted when adding only small amounts of data. ADAMUP then performs an in-core sort.

Use the SUMMARY function to get information about the required SORT and LWP sizes.

TEMP Data Set Placement

It is recommended that the TEMP data set does not reside on the same volume as the input file that contains the Descriptor Value Tables and the SORT. Although the size of TEMP is closely related to the performance when loading the Normal/Main Index, successful execution does not depend on a given size or the presence of a TEMP.

Use the SUMMARY function to display the recommended TEMP size.

Examples

Example 1:

```
adamup: dbid=1
adamup: update=10
adamup: add, userisn
```

File 10 of database 1 is updated by adding new records. The ISN given with each input record is used.

Example 2:

```
adamup: dbid=1
adamup: update=10
adamup: delete
```

The records identified by the ISNs provided on the input file are to be deleted from file 10 of database 1. The ISNs to be deleted are in binary format.

Example 3:

```
adamup: dbid=1
adamup: update=10
adamup: add, skiprec=1000
adamup: delete, data_format=decimal, log=full
```

New records are to be added while old ones are deleted from file 10 of database 1. The first thousand records found on the input file are not added. The ISN for each record added is assigned by ADAMUP. The ISNs of the records to be deleted are supplied in decimal format on the input file. All records which have been deleted are logged on an output file. The values required to re-create the inverted list when reloading are included in the log.

16 ADAOPR (Operator Utility)

- Functional Overview 210
- Procedure Flow 211
- Checkpoints 211
- Control Parameters 212
- RBAC Protection 261

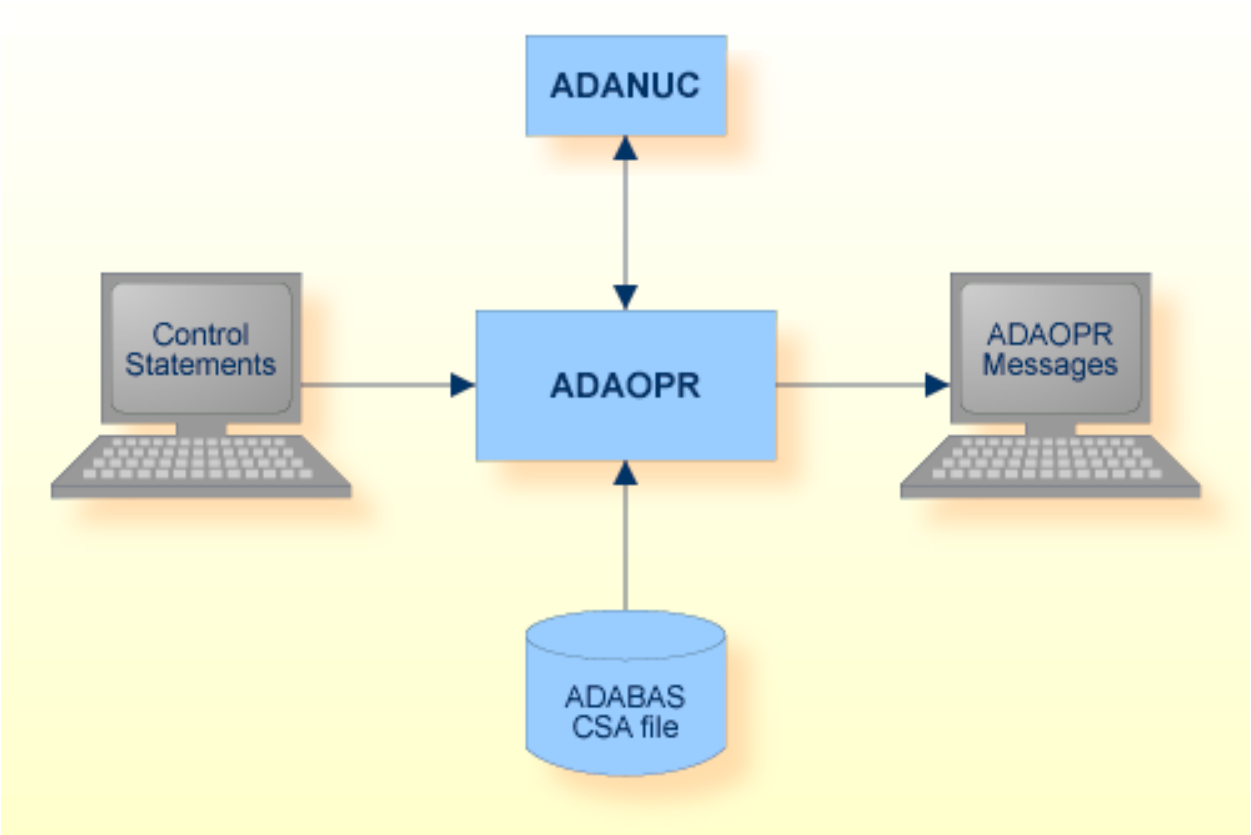
This chapter describes the utility "ADAOPR".

Functional Overview

The DBA uses this utility to operate the Adabas nucleus.

This utility is a multi-function utility. For more information about single- and multi-function utilities, see *Adabas Basics*, *Using Utilities* in the Adabas documentation.

Procedure Flow



Data Set	Environment Variable / Logical Name	Storage Medium	Additional Information
Control statements	stdin		Utilities Manual
ADAOPR messages	stdout		Messages and Codes

Checkpoints

The following table shows the nucleus requirements for each function and the checkpoint written:

Function	Nucleus must be active	Nucleus must NOT be active	Nucleus is NOT required	Checkpoint written
FEOF=PLOG	x			SYNC (see note 1)
EXT_BACKUP=PREPARE	x			SYNX (EXT_BACKUP STARTED) (see note 2)
EXT_BACKUP=CONTINUE	x			SYNX (EXT_BACKUP) SYNC (FEOF=PLOG) (see note 1)

Notes:

1. After the FEOF=PLOG checkpoint, ADANUC writes a SYNC checkpoint for the start of the new PLOG session.
2. Writing the checkpoint for EXT_BACKUP=PREPARE was introduced with Adabas Version 6.3 SP 4, and for Adabas Version 6.4 SP2.

Control Parameters

The following control parameters are available:

```
ABORT

ADD_REPLICATION [= number]
,FILE = number
,TARGET_DBID = number
,TARGET_FILE = number

BFIO_PARALLEL_LIMIT = number

CANCEL

CHANGE_REPLICATION keyword
,REPLICATION_ID = (number [ - number] [ , number [- number]] ...

CLEAR_FILE_STATS = (number [- number] [, number [- number] ] ... )

CLUSTER_LOG_LEVEL = [ FATAL | ERROR | WARN | INFO | DEBUG ]

CONNECTION_STRING = string

CSA = string

DBID = number

DELETE_REPLICATION = (number [ - number] [ , number [- number]] ...
```

```
     DELUI = number

     DISPLAY = (keyword [,keyword]...)

     ES_ID = number

D    [NO]ET_SYNC

     [NO]EVENTING

     EXT_BACKUP = [PREPARE | CONTINUE | ABORT]

     FEOF = (keyword [,keyword])


     FILE = number


     FREE_CLQ

     ID = number

D    [NO]IO_TIME

     ISN = ( number [- number] [,number [- number] ] ... )

     [UN]LOCK = (number [,number]...)

     LOGGING = (keyword [,keyword]...)

     LOGIN_ID = string

     NISNHQ = number

     NODE_ID = string

     OPTIONS = (keyword [,keyword]...)

     RESET = keyword

D    [NO]RESPONSE_ABORT

     RESPONSE_CHECK = (number[-number][,number[-number]]...)

     SET_FILE_STATS = (number[-number][,number[-number]]...)

     SHUTDOWN

     STATUS = (keyword [,keyword]...)

     STOP = (number[-number][,number[-number]]...)
```

```
 STOPI = number

TCP[ENABLE | DISABLE]

 THREAD = number

 TNAA = number

 TNAE = number

 TNAX = number

 TT = number

 USER_ID = string

 WCHARSET = <ICU encoding>

 WRITE_LIMIT = [number]
```

ABORT

```
ABORT
```

This function terminates the Adabas session immediately. All command processing is immediately stopped. The session is terminated abnormally with a pending AUTORESTART.

ABORT causes the following files to be written to the databases's default directory:

- The CSA dump file, which contains status information from the adabas nucleus. The name of the file is ADABAS.*xxx*.*hh:mm:ss* (Linux) or ADABAS.*xxx*.*hh-mm-ss* (Windows),where *xxx* is the database ID and *hh:mm:ss* (or *hh-mm-ss*) is the time at which the file was created. ADAOPR can also display the same information that you can get for a running nucleus for a CSA dump file if you specify the CSA parameter.
- The SMP dump file, which contains some diagnostic information. The name of the file is SAGSMP.*xxx*.*hh:mm:ss* (Linux), SAGSMP.*xxx*.*hh-mm-ss* (Windows) , where *xxx* is the database ID and *hh:mm:ss* (or *hh-mm-ss*) is the time at which the file was created.

This parameter setting is not supported when CONNECTION_STRING is used.

ADD_REPLICATION

```
ADD_REPLICATION [= number]
                ,FILE = number
                ,TARGET_DBID = number
                ,TARGET_FILE = number
```

This parameter is only relevant for customers who are using the Adabas Event Replicator with Adabas - Adabas Replication.

A new Adabas - Adabas replication is defined with status Inactive. It is optional to specify a non-zero number as the replication ID.

Note: A value may be specified in the range 1 to 524280.

This number must not be the replication ID of an existing replication. If no number is specified, a replication ID is created by Adabas. You must specify the source file for the replication, the target database, and the target file number.

This parameter setting is not supported when CONNECTION_STRING is used.

BFIO_PARALLEL_LIMIT

```
BFIO_PARALLEL_LIMIT = number
```

This function sets the number of parallel I/O requests by a buffer flush, allowing earlier processing of concurrent I/Os from other threads. A large buffer flush, for example, can cause the I/O queue to be very busy, and other I/Os (such as buffer pool read I/Os and WORK I/Os) can be enqueued for a long time, slowing down command throughput and possibly causing applications to stall if a buffer flush is active.

If BFIO_PARALLEL_LIMIT is specified, the buffer flush sets up the specified number of I/Os and waits until these have been processed before issuing the next packet. The maximum value for ´number´ is defined by the Adabas system, If a value of 0 is specified, the number of buffer flush I/Os is unlimited.

This parameter setting is not supported when CONNECTION_STRING is used.

CANCEL

```
CANCEL
```

This function terminates the Adabas session immediately. A BT command is issued for each active ET user and the session is terminated.

The communication link to the database is cut but the shared memory is still held. In this case, display functions are still possible with ADAOPR but parameter modification commands are no longer permitted.

This parameter setting is not supported when CONNECTION_STRING is used.

CHANGE_REPLICATION

```
CHANGE_REPLICATION = keyword
                   , REPLICATION_ID = (number [ - number] [ , number [- number]] ...
```

This parameter is only relevant for customers who are using the Adabas Event Replicator with Adabas - Adabas Replication.

CHANGE_REPLICATION can be used to change the status of one or more replications. A replication can have one of the following status values:

Status	Meaning
Inactive	Currently no data are replicated to the target file, and no activities have been performed to initiate the replication.
Prepare	This indicates that it is planned to perform the Initial State processing for the replication. This status is the prerequisite for creating a backup of the files to be replicated using ADABCK with the parameter REPLICATION.
Initialization	This indicates that ADABCK with the parameter REPLICATION is running and creating a backup that contains the initial state of files to be replicated.
Recording	Adabas is recording the update transactions within the replication command file and the replication transaction file, but currently is not replicating the update operations to the target database.
Active	The replication is active; all modifications of the source file are replicated to the target file.
Error	An unexpected error occurred during replication. In order to continue replication, a new initial state processing is required.

The following options can be specified to change the replication status:

Keyword	Meaning
INACTIVE	Currently no data is replicated to the target file, and it is not currently planned to start the replication. Transactions not yet replicated to the target database are deleted.
INITIALIZATION	Prepare the initial state processing: the status is set to Prepare. Then normally ADABCK DUMP/EXUDUMP must be called with the parameter REPLICATION for the files to be replicated; ADABCK first sets the status to Initializing, and then during ET synchronization, when the current state in the database is the same as on the backup file, ADABCK sets the status to Recording. Alternatively you can perform your own initial state processing and then perform ADAOPR CHANGE_REPLICATION=RECORDING.
RECORDING	One of the following: ■ The data of the file to be replicated have been saved without using ADABCK with the parameter REPLICATION. The replication status is set to Recording; this means that new database modifications are recorded to be replicated to the target file as soon as the copying of the data to the target file has been completed and you set the status to Active. **Note:** Performing ADAOPR CHANGE_REPLICATION = RECORDING is not required if you use ADABCK with the parameter REPLICATION to save the data; in this case ADABCK sets the status to Recording. ■ The replication is to be stopped, and new database modifications are only recorded; they will be replicated to the target database as soon as the replication is activated again.
ACTIVE	The replication is active; all modifications of the source file are replicated to the target file.

You must specify the replication IDs for which the status change is to be performed.

Notes:

1. If the list of replications to be modified contains a file with a referential integrity constraint, you must also specify replications for the related files and the same target database.
2. The following matrix shows the allowed options depending on the current replication status and the resulting status changes:

Status/Keyword	INACTIVE	INITIALIZATION	RECORDING	ACTIVE
Inactive	Inactive	Prepare	-	-
Prepare	Inactive	Prepare	Recording (see note 1)	-
Initialization	Inactive	Init	-	-
Recording	Inactive	Recording	Recording	Active
Active	Inactive	Prepare	Recording (see note 2)	Active

Status/Keyword	INACTIVE	INITIALIZATION	RECORDING	ACTIVE
Error	Inactive	Prepare	-	-

Notes:

1. The recommended way for an initial state processing is to use ADABCK with the parameter REPLICATION. ADABCK then sets the status first to Initialization, and later to Recording, when the backup is finished. You must only set the status to Recording if you don't use ADABCK for initial state processing, for example because you want to start the replication with an empty file.
2. It is not necessary to set the replication to Recording if the target database is shutdown, for example, for maintenance reasons. Then the database remains at status Active and the target database is polled until it is available again. Setting the status to Recording may be useful, for example, if you want to replicate the updates at night, which were done during the day, and if your target database should contain the database state of the previous day.

This parameter setting is not supported when CONNECTION_STRING is used.

CLEAR_FILE_STATS

```
CLEAR_FILE_STATS = (number [- number] [, number [- number] ] ... )
```

This function disables the collection of I/O statistics enabled by SET_FILE_STATS for the specified file(s).

CLUSTER_LOG_LEVEL

```
CLUSTER_LOG_LEVEL = [ FATAL | ERROR | WARN | INFO | DEBUG ]
```

This function changes the logging level of the cluster dynamically. The default logging level is `'INFO'`.

You can set the logging level to `'WARN'` to reduce the number of cluster based log entries in the nucleus log file.

Use the **DISPLAY=CLUSTER** command to display the current log level.

CONNECTION_STRING

```
CONNECTION_STRING = string
```

This parameter selects the (remote) database to which all subsequent ADAOPR commands apply. ‘String’ is string format to establish the (remote) database connection. This parameter connects the adaopr utility to a (remote) database via TCP/IP (ADATCP/ADATCPS). [protocol], [hostname] and [port number] are mandatory fields to establish connection to the remote database via AD-ATCP/ADATCPS. [hostname] address and [port number] are defined with the (remote) database's host name or IP address and port number. (Remote) data-base is accessible if ADATCP feature is enabled for the database.

The CONNECTION_STRING or DBID parameter must be the first ADAOPR parameter specified.

The parameters DBID, CSA and CONNECTION_STRING are mutually exclusive.

String format:

CONNECTION_STRING=[protocol]://[hostname]:[port number]

It must be succeeding the defined syntax, otherwise it will not establish the connection to remote database.

Example:

CONNECTION_STRING = adatcp://localhost: 50001

Note: The parameters (CLEAR_FILE_STATS, DISPLAY, ES_ID, FILE, ID, IO_TIME, ISN, LOGIN_ID, NODE_ID, RESET, SET_FILE_STATS, STATUS, USER_ID) settings are supported only when CONNECTION_STRING is used. All other adaopr control parameters are not allowed when CONNECTION_STRING is used but can be used in combination with DBID (local IPC communication).

CSA

```
CSA = string
```

'string' is a file specification of a file containing status information from an Adabas nucleus, a so-called CSA dump file. This file may be created by an ADAOPR ABORT function, by an abnormal termination of Adabas, or by response check trapping (refer to the RESPONSE_CHECK function for further information).

The following naming conventions are used for the file:

Linux

```
ADABAS.xxx.hh:mm:ss
ADABAS.xxx.RSPyyy.hh:mm:ss
```

Windows

```
ADABAS.xxx.hh-mm-ss
ADABAS.xxx.RSPyyy.hh-mm-ss
```

(with the NORESPONSE_ABORT option set), where

- 'xxx' is the three digit database ID;
- 'yyy' is the trapped three digit response code;
- 'hh:mm:ss' is the time the file was created (Linux),
- 'hh-mm-ss' is the time the file was created (Windows)

For example, if the database ID is 5, and the file creation was initiated by a trapped response code 113, the file name will start with ADABAS.005.RSP113, and then the time of creating will be appended, e.g. ADABAS.005.RSP113.12:16:50 (Linux) or ADABAS.005.RSP113.12-16-50 (Windows).

The file will be created in the directory that is pointed to by the environment variable/logical name ADA_CSA_DUMP. The default is the directory from which the nucleus was started. If a file with the same name already exists in this directory, it will be overwritten.

The DBID, CSA and CONNECTION_STRING parameters are mutually exclusive.

DBID

```
DBID = number
```

This parameter selects the database to which all subsequent ADAOPR commands apply. Multiple DBIDs are supported within one session.

The DBID, CSA and CONNECTION_STRING parameters are mutually exclusive.

Example:

```
adaopr: dbid=1
adaopr: shutdown
adaopr: dbid=2
adaopr: shutdown
adaopr: dbid=3
adaopr: shutdown
adaopr: quit
```

DELETE_REPLICATION

```
DELETE_REPLICATION = (number [ - number] [ , number [- number]] ...
```

This parameter is relevant only for customers who are using the Adabas Event Replicator with Adabas - Adabas Replication.

The replications with the specified replication IDs are stopped, if they are active, and deleted, including the commands and transactions that have not yet been replicated to the target files.

This parameter setting is not supported when CONNECTION_STRING is used.

DELUI

```
DELUI = number
```

Use the `DELUI` command to delete all users who have not executed a command during the specified number of seconds. Any open transactions of the deleted users are backed out.

This command does not delete EXF or UTI users.

Caution: If Adabas is running NOT with `options = OPEN_REQUIRED` (specifying that users are not required to issue an OP as the first command of the session), run the `DELUI` command only if you are certain that the users to be deleted are no longer active. If a user with an open transaction is deleted, but then returns (by sending a command), no indication is given about the transaction backout. If the user continues the transaction, logical inconsistencies in the database could occur.

DISPLAY

```
DISPLAY = (keyword [,keyword]...)
```

This parameter displays various information during an Adabas session.

Note: Certain `DISPLAY` options show a table with the Node ID, which represents the hostname of the machine where the client is running. If the hostname exceeds eight characters, it will be truncated. By default, only the first eight characters of the hostname are displayed in the Node ID column. However, the environment variable `ACLHOSTNAME_STARTINDEX` can be used to define a starting position within the hostname, from which the next eight characters will be displayed. For example, if the hostname is `my_hostname_machine7` and the variable is `ACLHOSTNAME_STARTINDEX=12`, the first 12 characters are skipped and `machine7` is displayed in the Node ID column. Since `ACLHOSTNAME_STARTINDEX` is used by the Adabas Client (ACL), it must be set on the machine where the client application is running.

The following keywords can be used:

Keyword	Meaning
ACTIVITY	Database activities display.
BF_STATISTICS	Buffer flush statistics display.
BP_STATISTICS	Buffer pool statistics display.
CLUSTER	Database cluster status display. For more information about this keyword and a sample output, see *Adabas Cluster -> Monitoring -> Minimal Requirements: Health Check -> Health Check for Primary and Secondary Nodes*.
COMMANDS	Command table display.
CQ	Command queue display.
DYNAMIC_PARAMETERS	Dynamic nucleus parameters display.
FILE_IO	File I/O display.
FP_STATISTICS	Format pool statistics display.
HIGH_WATER	High water marks display.
HQ	Hold queue display.
ICQ	Internal command queue display.
IO_TIMES	Container I/O times display.
PLOG_STATISTICS	Protection log statistics.
REPLICATIONS	Adabas - Adabas replications.
RPL_STATS	Internally-collected replication statistics.
STATIC_PARAMETERS	Static nucleus parameters display.
TCPCONNECTIONS	ADATCP connections display.
TT	Thread table display.
UCB	Utility communication block.
UQ	User queue display.
UQ_FILES	User file list display.
UQ_FULL	Full information about user queue element.
UQ_TIME_LIMITS	User time limits display.

The following examples show the information produced by the various keywords, together with explanations of the information that is displayed.

Some of the following displays include percentages. The corresponding values are always truncated. An undefined value (divided by 0) is specified with " *%" and an overflow with "***%".

Example: DISPLAY=ACTIVITY

```
adaopr: display=activity

                       ADANUC Version <version number>
        Database 76           Activity          on 22-JAN-2014 13:19:30


I/O Activity                      Total   Throwbacks                        Total
------------                      -----   ----------                        -----
Buffer Pool                       5,440   Waiting for UQ context               87
WORK Read                           728   Waiting for ISN                      53
WORK Write                          647   ET Sync                               0
PLOG Write                          194   DWP Overflow                          0
NUCTMP                            1,600
NUCSRT                              531

Pool Hit Rate                     Total   Interrupts       Current         Total
-------------                     -----   ----------       -------         -----
Buffer Pool                         99.6% WP Space Wait          0             0
Format pool                         98%
```

The information has the following meaning:

- I/O ACTIVITY shows the total numbers of:
  - physical buffer pool I/Os (physical read I/Os + physical write I/Os);
  - read and write I/Os for WORK and PLOG.
  - I/Os for NUCTMP and NUCSRT
- INTERRUPTS shows the current and total number of workpool space waits;
- POOL HIT RATE shows:
  - the buffer pool hit rate. This is the relationship between the logical read I/Os and the physical read I/Os. The buffer pool hit rate is calculated using the following formula:

    ```
    hit rate (in %) =  ((logical read I/Os - physical read I/Os) * 100)
        / logical read I/Os
    ```

  - the format pool hit rate. This is the relationship between the number of format buffer requests (required FBs) and the required format buffers already translated in the format pool (translated FBs). The format pool hit rate is calculated using the following formula:

    ```
    hit rate (in %) = ((translated FBs * 100) / required FBs)
    ```

- THROWBACKS shows:
  - the number of commands waiting for session context because internal commands were running;
  - the number of commands waiting because ISNs are held by another user;

- the number of commands waiting for ET synchronization;
- the number of commands thrown back because of dynamic work pool overflow.

EXAMPLE: DISPLAY=BF_STATISTICS

```
adaopr: disp=bf_statistics
%ADAOPR-I-STARTED,      18-OCT-2016 16:05:03 Version <version number>

Database 37, startup at 18-OCT-2016 16:04:40
ADANUC Version 6.5.1.0, PID 10448

                        ADANUC Version 6.5.1.0
        Database 37    Buffer Flush Statistics  on 18-OCT-2016 16:05:02


Buffer flush statistics:
------------------------

Buffer flush         Write Number of Type    Size  Average IO Duration Rejected
start time           Limit    Blocks         (MB) time (msec)    (sec)    Locks
-------------------------------------------------------------------------------
18-OCT-2016 16:04:40     2         4 DB      0.04        0.00     0.00        0
18-OCT-2016 16:04:59     2       128 DB      0.54        0.35     0.04        0
18-OCT-2016 16:05:00     2       128 DB      0.53        0.96     0.12        1
18-OCT-2016 16:05:00     2         9 DB      0.06        0.00     0.00        0
18-OCT-2016 16:05:00     2         5 DB      0.04        3.00     0.01        0
18-OCT-2016 16:05:01     2       126 DB      0.53        0.98     0.12        1
18-OCT-2016 16:05:01     2         7 DB      0.05       11.00     0.07        0
18-OCT-2016 16:05:01     2        12 DB      0.07        0.00     0.00        0
18-OCT-2016 16:05:02     2       128 DB      0.54        0.85     0.10        0
18-OCT-2016 16:05:03     2       131 DB      0.55        0.10     0.01        0

Total number of flushes:        10
Explicit               :         1
Write limit            :         0
WORK limit             :         5
Space                  :         0
Emergency              :         0
Ignored blocks         :         4
```

This display shows the statistics of the buffer flushes; if more than 100 buffer flushes have been performed in the current nucleus session, the last 100 buffer flushes are displayed. The following information is displayed for each buffer flush:

- The start time of the buffer flush.
- The current write limit. The write limit for database blocks can be modified via ADAOPR WRITE_LIMIT. The write limit for temporary blocks cannot be changed.
- The number of blocks included in the buffer flush.

- The type of buffer flush:
  - DB means flush of database blocks
  - Temp means flush of temporary blocks
- The size in megabytes of the blocks included in the buffer flush.
- The average I/O time of the I/Os performed by the buffer flush in milliseconds.
- The duration of the buffer flush in seconds.
- The number of rejected locks is the number of blocks that were not written immediately during the buffer flush, because the block was exclusively locked when the buffer flush tried to write the block. The rejected blocks are either written after having written the other blocks - then the buffer flush waits until the lock can be granted, or by a separate ignore-blocks buffer flush.

After the table the total number of buffer flushes is displayed, and a breakdown of the reasons for the buffer flushes.

Notes:

1. The above displayed example database uses a small WORK container with the effect that the condition for a WORK limit buffer flush occurs before the write limit is exceeded. Therefore, the above example database displays WORK limit buffer flushes, but no write limit buffer flushes.
2. It may happen that two threads determine at nearly the same time that a buffer flush is required. Then both threads set a flag that a buffer flush is required. When the first thread has set the flag, the buffer flush thread starts a buffer flush and resets the flag. Then the second thread sets the flag again. When the buffer flush is finished, a new buffer flush is started immediately. Because such unnecessary buffer flushes do not cause errors, no logic is implemented to avoid such buffer flushes. In the example, the fifth and the eighth buffer flush are such unnecessary buffer flushes. They are displayed as "Ignored blocks" buffer flushes; therefore 4 Ignored blocks buffer flushes are displayed, although only 2 blocks were ignored.

EXAMPLE: DISPLAY=BP_STATISTICS

```
adaopr: display=bp_statistics

                        ADANUC Version  <version number>
        Database 34     Buffer Pool Statistics   on  5-JUN-2014 13:11:28


Buffer Pool Size   :   419,430,400

Pool Allocation                          RABNs present
---------------                          -------------
Current     (  7%) :    32,835,584       ASSO               :            33
Highwater   ( 10%) :    42,676,224       DATA               :             5
```

```
Internal    (  7%) :    30,770,176     WORK               :             0
Workpool    (  0%) :     1,408,000     NUCTMP             :             0
                                       NUCSRT             :             0

I/O Statistics                         Buffer Flushes
--------------                         --------------
Logical Reads      :           340     Total              :             3
Physical Reads     :            17     To Free Space      :             0
Pool Hit Rate      :            95.0%  Temporary Blocks   :             0

                                       Write Limit  (  2%):     8,388,600
Physical Writes    :            41     Modified     (  0%):       108,544

                                       Limit Temp.B.( 50%):   209,715,000
                                       Modified T.B.(  0%):             0
```

The information is interpreted as follows:

- POOL ALLOCATION shows:
  - the size in bytes and percentage of the buffer pool that is currently in use;
  - the size in bytes and percentage of the buffer pool high water mark (see also the display for DISPLAY=HIGH_WATER).
- RABNs PRESENT shows:

  “the number of ASSO, DATA and WORK RABNs currently in the buffer pool.”
- I/O STATISTICS shows:
  - the total number of logical and physical buffer pool read I/Os (both numbers are required in order to calculate the buffer pool hit rate);
  - the buffer pool hit rate (please refer to the example for DISPLAY=ACTIVITY for the buffer pool hit-rate formula);
  - the total number of physical buffer pool write I/Os.
- BUFFER FLUSHES shows:
  - the total number of buffer flushes;
  - the total number of buffer flushes that were made in order to get free space;
  - the total number of buffer flushes for temporary blocks;
  - the size and percentage of the buffer pool WRITE LIMIT for database blocks;
  - the size in bytes and percentage of modified database blocks;
  - the size and percentage of the buffer pool WRITE LIMIT for temporary blocks;
  - the size in bytes and percentage of modified temporary blocks.

Example: DISPLAY=COMMANDS

```
adaopr: display=commands

                        ADANUC Version <version number>
        Database 76           Commands          on 19-JAN-2014 14:58:10


ADABAS Commands:             9,884

        A1            892         L2             553        OP            25
        BT            736         L3           1,124        RC            89
        C1             40         L4             569        RE             0
        C3              0         L5             420        RI             0
        C5             10         L6             436        S1         1,511
        CL             32         L9             456        S2            81
        E1          1,006         LF              20        S4            12
        ET             72         MC               0        S8           230
        HI              0         N1             877        S9            50
        L1            643         N2               0
```

This command displays the total numbers of Adabas commands issued in the current session. For MC commands, the value displayed is the number of MC calls plus the number of single Adabas commands contained in the MC calls.

A read command that is issued while the multifetch option is set is counted as a single command.

Updates made by utilities are not included in the display.

Note: The command counts can be reset by ADAOPR RESET=COMMANDS.

Example: DISPLAY=CQ

```
adaopr: display=cq
                        ADANUC Version <version number>
        Database 2         Command Queue        on 14-NOV-2014 13:41:53


  No   Node Id  Login Id       ES Id  APU Cmd File  Status
  --   -------  --------       -----  --- --- ----  -----
   1  PC0001    miller          3316    1  RC   13  Ready to run
   2  PC0001    jones           1360    2  S8   13  Running
   3  PC0001    smith           6148    1  RC   13  Ready to run
   4  PC0001    miller          4208    1  S8   13  Running
   5  PC0001    jones           5224    2  S9   13  Ready to run
   6  PC0001    dba             7024    2  U1    0  Running
   7  PC0001    brown           3140    2  S1   13  Running
   8  PC0001    meyer           6180    2  S8   13  Running
```

```
   9  PC0001    smith           4756    1  S1   13  Running
  10  PC0001    king            1240    2  ET    0  Ready to run
  11  PC0001    meyer            836    2  RC   13  Ready to run
  12  PC0001    brown           6272    1  L6   13  Ready to run

Selected: 12, Used: 12, Queue Size: 13
```

This display shows the current command-queue entries:

- NODE ID shows the node identification string.
- LOGIN ID shows the login user identification string;
- ES ID shows the environment-specific identification (for example, the process ID);
- APU shows the assigned Adabas Processing Unit of the command queue entry if the nucleus parameter APU is set. If APU has not been specified, the column APU is not displayed;
- CMD shows the command string;
- FILE shows the file number;
- STATUS shows the status of the command-queue entry.

The final line of the display shows how many command queue entries were selected according to the currently active selection criteria, and how many entries are used in total in the command queue.

The possible status values are shown in the following table:

Status	Meaning
Completed	Command processing completion;
Marked For Deletion	Command is marked for delete, user is no longer active;
New	Command is ready to be inserted in the scheduling queue;
Ready To Run	Placed in queue and ready for scheduling;
Running	Running in a thread (see DISPLAY=TT);
Waiting For Complex	Complex command is waiting to run;
Waiting For Et Sync	Waiting for ET synchronization;
Waiting For Group Commit	Waiting for group ET. No entry in thread table;
Waiting For Isn <isn>	Waiting for ISN in file shown in column "File" in the display. No entry in thread table;
Waiting For Space	Waiting for working space. No entry in thread table.
Waiting For Uqe	Waiting for user queue entry. The required entry is locked by an active internal command;

Note: The display may show command codes such as "U0", which are only used internally by Adabas (for example, during a utility run). The "RUNNING" and "COMPLETED" values may differ even if the user has not specified an explicit selection criterion.

Example: DISPLAY=DYNAMIC_PARAMETERS

```
adaopr: display=dynamic_parameters

                        ADANUC Version <version number>
        Database 76       Dynamic Parameters     on 19-JAN-2014 14:58:10


Resources:          NISNHQ    :           100    WRITE_LIMIT:           50%

Time Slices:        TNAA      :           900    TNAX       :           900
                    TNAE      :           900    TT         :           300

Logging:            CLOG      : OFF


Read limits:        200, 10, 30

Response check with ABORT    : 84,160,164-182,243,251-252
```

This display shows the current values of the dynamic nucleus parameters.

Example: DISPLAY=FILE_IO

```
adaopr: display=file_io

                        ADANUC Version <version number>
        Database 76            File I/O          on 19-JAN-2014 14:58:10


                        Reads           Hit
    File       Logical       Physical  Rate         Writes
    ----       -------       --------  ----         ------
      11       145,341            180   99%          2,869
      12        99,070            148   99%          2,149
```

This display shows the logical and physical reads, their hit rate and the writes the buffer pool manager has made for every file since the file I/O statistiscs for the file in question were enabled (ADAOPR SET_FILE_STATS) - files for which the I/O statistics have not been enabled or for which no I/Os were performed are not displayed.

Notes:

1. The formula for the hit rate value is given in the description of DISPLAY=ACTIVITY.
2. A write operation is only counted if the block was not yet marked as modified. This means that the physical write I/Os either already done in a previous buffer flush or still pending to be performed in the next buffer flush are counted.

Example: DISPLAY=FP_STATISTICS

```
adaopr: display=fp_statistics

                        ADANUC Version <version number>
        Database 76     Format Pool Statistics   on 19-JAN-2014 14:58:10

Maximum Local  Pool Size:        251,656
Maximum Global Pool Size:        251,656

Pool Allocation                             Pool Contents
---------------                             -------------
Local  Current   ( 22%) :         57,540    Local  Format Buffers:         162
Local  Highwater ( 27%) :         70,000    Global Format Buffers:           1

Global Current   (  0%) :             84
Global Highwater (  0%) :             84


Pool Statistics                    Local          Global
---------------                    -----          ------
Scans                             11,780               3
Hits                              11,547               2
Hit Rate                              98%             66%

Replacements                           0               0
Overflows                              0               0
```

This display shows the format pool statistics:

- POOL ALLOCATION shows:
  - the size in bytes and percentage of the local and global format pools that are currently in use;
  - the size in bytes and percentage of the local and global format pool high water marks.
- POOL STATISTICS shows:
  - the total number of scans and hits of valid format buffers in the format pool (both numbers are required in order to calculate the format pool hit rate);
  - the format pool hit rate (please refer to the example DISPLAY=ACTIVITY for the format pool hit-rate formula);
  - the total number of valid format buffers that are overwritten in the format pool (replacements).
  - Overflows. This is the number of times that a format buffer exceeded the format pool size, resulting each time in a response 42.
- POOL CONTENTS shows:
  - the number of valid local format buffers in the format pool;
  - the number of valid global format buffers in the format pool.

Example: DISPLAY=HIGH_WATER

```
adaopr: display=high_water
                        ADANUC Version <version number>
        Database 2        High Water Marks      on 21-NOV-2014 11:44:19

Area/Entry                Size      In Use  High Water    %      Date/Time
----------                ----      ------  ----------    -      ---------
User Queue                 100          13          13   13 21-NOV-2014 11:44:00
Command Queue                -          12          13    - 21-NOV-2014 11:44:19
  APU 01                     -           2          12    - 21-NOV-2014 11:44:02
  APU 02                     -          13          15    - 21-NOV-2014 11:44:00
Hold Queue                   -           2           2    - 21-NOV-2014 11:44:00
Client Queue               100          13          13   13 21-NOV-2014 11:44:00
HQ User Limit                -           -           1    - 21-NOV-2014 11:44:00
Threads                      6           4           6  100 21-NOV-2014 11:44:00
Workpool           524,288,000           0 131,072,016   25 21-NOV-2014 11:42:16
  ISN Sort          65,536,000           -     380,000    0 21-NOV-2014 11:44:04
  Complex Search    65,536,000           -           0    0
Attached Buffer      1,048,576     219,136     219,136   20 21-NOV-2014 11:44:02
ATBX       (MB)             20           0           0    0
Buffer Pool(KB)      2,048,000     957,962   1,009,978   49 21-NOV-2014 11:42:16
Protection Area        127,990
  Active Area           38,397           -           4    0 21-NOV-2014 11:44:04
Group Commit                50           1           1    2 21-NOV-2014 11:42:17
Transaction Time         3,000           -           0    0
```

This display shows the high water marks for the current session:

- SIZE shows the size in bytes of pools and buffers. For queues, threads and hold queue user limit, it shows the number of entries.
- IN USE shows the size in bytes or number of entries currently in use.
- HIGH WATER shows the maximum quantity required simultaneously for the given area/entry.
- % shows the relationship between the high water mark and the size. If the high water mark exceeds the size, the value in this column can be larger than 100 %. For example, this can occur if the value is decreased by ADAOPR, or if the original area has been dynamically increased. This is normal Adabas behavior, and no changes of Adabas parameters are required.
- DATE/TIME shows the date/time at which the high water mark occurred for the first time. There is no output in this column if the high water mark is 0.

The entries in the column AREA/ENTRY correspond to the ADANUC parameters NU (user queue), NCL (client queue), NISNHQ (hold queue user limit), NT (threads), APU (Adabas Processing Units, only displayed if the nucleus parameter APU is set), LWP (workpool), LBP (buffer pool), LAB (attached buffer), TT (transaction time). The hold queue and the command queue have no predefined size and are increased dynamically if required.

The entry "ACTIVE AREA" is the largest part of WORK part 1 that can be used by a single transaction. If a transaction's protection information spans more space than allowed by "Active Area", it receives a response 9 (LP), the nucleus displays a PLOVFL message and a value of more than 100 in the "%" column of the highwater display.

Users who have set user-specific timeout values in their OP call are not included in the values for Transaction Time.

Notes:

1. 1. Values for Attached Buffer and Command Queue are not displayed correctly if the nucleus cannot be contacted by ADAOPR (for example, if the ADAOPR parameter CSA is used.
2. 2. Threads are used in a round-robin manner. Therefore, the high water mark for threads will be the same as the value shown in the Size column in most cases.
3. 3. During an autorestart following an abnormal nucleus termination, user queue elements are created for those users who are active during the time interval and for who the updates must be recovered. Therefore, directly after the start of the new nucleus session, the high water mark for the user queue can be relatively high, while the number of user queue elements in use is small.

Example: DISPLAY=HQ

```
adaopr: file=11, display=hq

                      ADANUC Version <version number>
        Database 76          Hold Queue         on 19-JAN-2014 14:58:10

        Id Node Id  Login Id      ES Id User Id  File           ISN Locks  Flg
        -- -------  --------      ----- -------  ----           --- -----  ---
        15 sunxxx01 miller         6974 *adatst    11         2,222   X     M
        19 sunxxx01 smith          7056 *adatst    11             2   X

Selected: 2, Used: 8, Queue Size: 160
```

This display shows the current hold-queue entries:

- ID shows the internal user identification of the user holding the ISN;
- NODE ID shows the node identification string. The local node is represented by an empty string;
- LOGIN ID shows the login user identification string;
- ES ID shows the environment-specific identification (for example, process ID);
- USER ID shows the user identification. Adabas utilities use the utility name preceded by an asterisk as the USER ID;
- FILE shows the number of the Adabas file in which the ISN is located;

- ISN shows the number of the ISN in hold;
- LOCKS shows the kind of lock for the ISN, where X = exclusive lock , S = shared lock.

  Note: S is displayed for shared locks starting with Adabas version 6.3 SP 1; in previous releases R is displayed.

- An M for FLG indicates that the record has been modified.

The final line of the display shows how many hold queue entries were selected according to the currently active selection criteria, and how many entries are used in total.

Entries are displayed in unsorted sequence.

Example: DISPLAY=ICQ

```
adaopr: display=icq

                        ADANUC Version <version number>
        Database 76     Internal Command Queue   on 19-JAN-2014 14:58:10


        Id  Node Id   Login Id       ES Id  Command   Status
        --  -------   --------       -----  -------   ------
  00000002            *system     00000000  SHUT      Running

Selected: 1, Used: 1, Queue Size: 101
```

This display shows the internal command queue:

Command	Meaning
AR	Autorestart
BT	Back out transaction
BTCL	Back out open transaction and close user
CANCEL	Cancel nucleus
DELUQE	Release file list and delete user queue element
SHUT	Shut down nucleus
STOP	STOP from ADAOPR
TIMEOUT	Non-activity timeout

The status of internal commands can be READY TO RUN, RUNNING, WAITING FOR ET SYNC or WAITING FOR UQE.

The final line of the display shows how many internal command queue entries were selected according to the currently active selection criteria, and how many entries are used in total.

Example: DISPLAY=IO_TIMES

```
adaopr: display=io_times

                        ADANUC Version <version number>
        Database 76        IO Statistics        on 19-NOV-2014 12:16:48


                  Number of IOs       Maximum IO time      Average IO time
                  -------------       ---------------      ---------------
ASSO Read    :               735574                14397                    1
ASSO Write   :                12136                    2                    1
DATA Read    :              2023257                13910                    1
DATA Write   :                  444                    1                    1
WORK Read    :                    4                    1                    1
WORK Write   :                  660                    2                    1
NUCSRT Read  :                 4060                  940                    1
NUCSRT Write :                 4060                    1                    0
NUCTMP Read  :                   30                    1                    1
NUCTMP Write :                  896                    1                    1
```

The number of IOs shows the number of physical read and write I/O accesses to ASSO, DATA, WORK, NUCSRT and NUCTMP.

The maximum IO time shows the maximum duration of a single I/O read and write access to ASSO, DATA, WORK, NUCSRT and NUCTMP in microseconds.

The average IO time shows the average time of a single I/O access to ASSO, DATA, WORK, NUCSRT and NUCTMP in microseconds.

Logging of I/O times is only available if ADAOPR IO_TIME is enabled..

Example: DISPLAY=PLOG_STATISTICS

```
adaopr: display=plog_statistics

                        ADANUC Version <version number>
        Database 76       PLOG Statistics       on 19-JAN-2014 14:59:41


PLOG Environment
----------------
NUCPLG     (active) : /FS/fsxxxx/sag/ada6180102/ada/db076/NUCPLG


Active PLOG
-----------
Session Number      :  37
```

```
Extent              :  2

Active Since        :  19-JAN-2014 14:59:41
Duration            :              00:00:01

Allocated Space     :         24,683 KB
Used Space   (  0%) :             32 KB
Average Growth Rate :        115,200 KB/h
```

Example: DISPLAY=REPLICATIONS

```
adaopr: display=replications
                        ADANUC Version <version number>
        Database 34         Replications        on 19-JAN-2014 09:47:48


       ID  From FNR  To DB  To FNR  Status         Remark
       --  --------  -----  ------  ------         ------
        1     111      37     111   Inactive
       86      86      37      86   Active


           2 transactions pending:
        ----------------------------

        To DB   Transactions
        -----   ------------
           37             2


           5 commands pending:
      --------------------------

      From FNR     Commands
      --------     --------
           86             5
          111             0
```

This display shows the Adabas - Adabas replications currently defined. This is only relevant for customers who are using the Adabas Event Replicator with Adabas - Adabas replication.

Note: Replications to other replication targets, for example SQL databases, are not displayed. Such replications can only be displayed with the administration tools of the event replication.

The display shows the following information:

- "ID" is the ID of the replication that is also used in the replication administration.
- "From FNR" is the file number of the file to be replicated to another Adabas file.
- "To DB" and "To FNR" are the database ID and file number of the target file for the replication.

- "Status" can have the following values and meanings:

Status	Meaning
Inactive	Currently no data are replicated to the target file, and at the moment no activities have been made to initiate the replication.
Prepare	This status indicates that it is planned to perform the initial state processing for the replication. This status is the prerequisite for creating a backup of files to be replicated via ADABCK with parameter REPLICATION.
Initialization	This status indicates that ADABCK with parameter REPLICATION is running and creates a backup containing the initial state of files to be replicated.
Recording	Adabas is currently recording the update transactions within the replication command file and the replication transaction file, but currently does not replicate the update operations to the target database.
Active	The replication is active; all modifications of the source file are replicated to the target file.
Error	An unexpected error occurred during replication. In order to continue replication, a new initial state processing is required.

- "Pending Transactions" is the number of transactions that have not yet been replicated to the target file.

Notes:

1. The number contains both transactions that have already been committed but not yet replicated to the target database, and transactions that are still open and which can only be replicated after an end of transaction.
2. If a transaction contains commands to be replicated to more than one target database, the transaction is counted only once, independent of the number of target databases. Therefore the total number of pending transactions can be smaller than the sum of the transactions for the different target databases.

- "Pending Commands" is the number of commands that have not yet replicated to the target file.

Notes:

1. The number contains both commands belonging to transactions that have already been committed but not yet replicated to the target database, and commands belonging to transactions that are still open and which can only be replicated after an end of transaction.
2. If a file is replicated to more than one target file, database modification commands of the source file are counted only once, independent of the number of target files to which a command has to be replicated.

If ADAOPR DISPLAY=REPLICATIONS is executed in non-interactive mode, ADAOPR returns one of the following exit status values:

Value	Meaning
0	At least one replication has been defined, and no replication is in status Error.
12	There is a replication in status Error.
15	Replication has not been activated, or no replication has been defined.

Example: DISPLAY=RPL_STATS

```
adaopr: start_rpl_stats
adaopr: display=rpl_stats

                        ADANUC Version <version number>
        Database 6      Replication Statistics   on 18-JUL-2016 11:24:47



Replication Statistics Summary - All Times in usec
-------------------------------------------------------------------------------
Transact not yet Repl (Cur/Max)                0                2
Replicated Transactions                      281
Transact Repl Time (Avg/Min/Max)           3,055                9          171,013
Transact Latency (Avg/Min/Max)             3,173               14          171,021

Replicated Commands                        4,984
Command Repl Time (Avg/Min/Max)                1                1               18
Replicated A1 Commands                     1,536
A1 Repl Time (Avg/Min/Max)                     1                1               11
Replicated E1 Commands                     1,711
E1 Repl Time (Avg/Min/Max)                     1                1               12
Replicated NX Commands                     1,737
NX Repl Time (Avg/Min/Max)                     1                1               18
Command Wait Counter                           2
Command Wait Time (Avg/Min/Max)           15,518               41           30,995
```

Notes:

1. Before displaying the replication statistics, the replication statistics must be activated with the command START_RPL_STATISTICS.
2. On Windows 7, the functions currently used to get the current time only have an accuracy of 1 millisecond; if the millisecond has not changed since the previous call, the time is increased by 1 microsecond. This means that the time values displayed are not very precise - if a value is significantly less than 1000, this only means that the time is less than one millisecond, but it will be probably significantly larger than the value displayed.

The display shows the following information:

Value	Meaning
Transact not yet Repl	The number of replications that have been committed, but have not yet been replicated. If the values are large, this means that the system is overloaded; Adabas is not able to replicate update operations in time. An exception where large values are normal is when the target database is down; then no transactions can be replicated, and the number of transactions not yet replicated increases.
Replicated Transactions	The number of transactions that have been replicated since the replication statistics were activated.
Transact Repl Time	The time to replicate a single transaction.
Transact Latency	The time between the commit of a transaction in the source database and the commit of the replicated transaction in the target database. **Note:** When the target database is down, transactions must wait for replication until the database is up again. This means you will get large values for transaction latency.
Replicated Commands	The number of commands that have been replicated since the replication statistics were activated.
Command Repl Time	The time required to replicate one command.
Replicated A1 Commands	The number of A1 commands that have been replicated since the replication statistics were activated.
Command A1 Repl Time	The time required to replicate one A1 command.
Replicated E1 Commands	The number of E1 commands that have been replicated since the replication statistics were activated.
Command E1 Repl Time	The time required to replicate one E1 command.
Replicated NX Commands	The number of N1 or N2 commands that have been replicated since the replication statistics were activated.
Command NX Repl Time	The time required to replicate one N1 or N2 command.
Command Wait Counter	If more than one transaction is replicated at the same time, it may happen that the replication of a command must wait for the termination of the replication of another command belonging to another transaction in order to guarantee the consistency of the replication. The counter shows how often this happened since the replication statistics were activated.
Command Wait Time	The time until the replication of a command could continue when the command replication had to wait for the termination of the replication of another command belonging to another transaction.

Example: DISPLAY=STATIC_PARAMETERS

```
adaopr: display=static_parameters
                       ADANUC Version <version number>
        Database 22      Static Parameters      on 21-NOV-2014 11:13:25


Resources:         LAB       :     1,048,576    NT        :             6
                   LBP       :   104,857,600    NU        :            50
                   LWP       :     1,000,000    NCL       :            50
                   LABX      :    20,971,520
                   APU       :  ( 2,  3,  2)

TCP/IP Port:       49152
TCP/IP Receiver:   4

Logging:           PLOG, BI
Options:           AUTO_EXPAND
```

This display shows the static nucleus parameters.

Note: The nucleus parameter APU is only displayed if it has been specified.

Example: DISPLAY=TCPCONNECTIONS

```
adaopr: display=tcpconnections

                       ADANUC Version <version number>
        Database 100        Connections         on  5-SEP-2022 09:59:37

Connect Time         Conn ID Recv ID User ID  Remote Host                      IP ↩
Address                        Port
-------------------- ------- ------- -------- ------------------------------- ↩
--------------------------------- ---------
 5-SEP-2022 11:03:44       1       1 bal      Node1                            ↩
192.169.10.98                     00000
 5-SEP-2022 11:04:50       2       0 bal      Node2                            ↩
192.169.10.99                     00000
```

The ‘Remote Host’, also known as ‘Node Id’ in the user queue, might not be the real client's host name. The host (node id) can be set by the client application (see also the client function lnk_set_adabas_id() in the section *Command Reference* > *Calling Adabas*). If the real host's name is wanted, the environment variable *ADATCP_DNSLOOKUP* can be set to ‘YES’. The variable can be set in the section ‘ENVIRONMENT’ of the *DBnnn.INI* file database specific, or in the shell environment. The default value is ‘NO’. This prevents performance issues because a DNS (Domain Name Service) lookup can be very time consuming.

Output after setting the environment variable:

```
adaopr: display=tcpconnections

                        ADANUC Version <version number>
        Database 100        Connections         on  5-SEP-2022 09:59:37
Connect Time         Conn ID Recv ID User ID  Remote Host                      IP ↩
Address                        Port
-------------------- ------- ------- -------- ------------------------------- ↩
--------------------------------- ---------
 5-SEP-2022 11:08:04       1       0 bal      pcbal1.softwareag.com             ↩
192.169.10.98                     36096
 5-SEP-2022 11:08:14       2       1 bal      pcbal2.softwareag.com             ↩
192.169.10.99                     48454
```

The 'Port' number is the local port number of the connection which helps to identify a connection with operating system tools.

Example: DISPLAY=TT

```
adaopr: display=tt
                        ADANUC Version <version number>
        Database 2          Thread Table        on 21-NOV-2014 11:49:38


 No  APU     Cmd Count  File  Cmd  Status
 --  ---     ---------  ----  ---  ------
  1    2       120,715    13   S9  Complex, waiting for DATA / 2785
  2    1       120,146    13   S8  Complex, waiting for TEMP / 35794
  3    2       124,364     0       Free
  4    1       122,300    13   S8  Complex, waiting for TEMP / 168654
  5    2       120,325    13   S8  Complex, active
  6    1       123,210    13   S1  Simple , active
```

This display shows the entries in the thread table. The number of displayed entries is simultaneously the high water mark for threads.

- APU shows the assigned Adabas Processing Unit of the thread if the nucleus parameter APU is set. If APU has not been specified, the column APU is not displayed.
- CMD COUNT shows the total number of Adabas commands processed from the corresponding thread context. The sum of these counts will normally differ from the sum shown by DISPLAY=COMMANDS, because internal commands are also counted.
- FILE shows the file number of the Adabas command that is currently being processed from the corresponding thread context. The file number is 0 if the corresponding thread context is not active, or if the command is a global one which is not linked to a particular file.

- CMD shows the command string of the Adabas command that is currently being processed from the corresponding thread context. There is no output in this column if the corresponding thread context is not active.
- STATUS shows the command type and the status of the corresponding thread context.

Possible command types are:

- Update
- Simple
- Complex

Possible entries for the thread status are shown in the following table:

Status	Meaning
free	available for allocation
ready	ready to run
active	running
waiting for io <rabn>/<block type>	waiting for I/o completion of block <rabn>
waiting for <rabn>/<block type>	waiting for access/update synchronization of block <rabn>
waiting for space <size> bytes	waiting for <size> bytes of work pool space
PLOG processing	Log entries for PLOG and WORK are created.
Waiting for PLOG processing	The thread wants to perform PLOG processing, but another thread is already performing PLOG processing - only one thread can create log entries at the same time. **Note:** The thread status entries are displayed one by one. Therefore, it can happen that for more than one thread status "PLOG processing" is displayed, or that status "Waiting for PLOG processing" is displayed for a thread, although for no other thread status "PLOG processing" is displayed.

Note: The display of the thread status is done for one thread after another. For this reason, it can happen that status "PLOG processing" is displayed for more than one thread, or that status "Waiting for PLOG processing" is displayed, although for no other thread status "PLOG processing" is displayed.

The block type can be ASSO, DATA, WORK, FILE or PLOG.

Example: DISPLAY=UCB

```
adaopr: display=ucb

                        ADANUC Version <version number>
        Database 76             UCB             on 19-JAN-2014 14:59:45


       Date/Time        Entry Id  Utility  Mode Files
       ---------        --------  -------  ---- -----
19-JAN-2014 14:59:41          42   adaopr  UTO  13
```

This display shows the utility communication block.

- DATE/TIME shows the date and time on which the given files were locked.
- ENTRY ID shows the allocated identification of the entry.
- UTILITY shows the name of the utility.
- MODE shows the mode in which the files are being accessed. The possibilities are:
  - ACC open for access
  - UPD open for update
  - EXU open for exclusive update (parallel access allowed)
  - UTO open for utilities only
  - UTI open for exclusive access (no parallel access or update allowed)
- Files shows the file numbers of the files that are locked.

Example: DISPLAY=UQ

```
adaopr: display=uq

                        ADANUC Version <version number>
        Database 76         User Queue          on 19-JAN-2014 14:58:10


        Id  Node Id   Login Id       ES Id   User Id    Type    Status
        --  -------   --------       -----   -------    ----    ------
        26  sunxxx01  dba             4473   *adaopr    UT
        23  sunxxx01  smith           3075              ET          E
        20  sunxxx01  jones           3178              ET         I
        19  sunxxx01  jones           1946              ET         IE
        18  sunxxx01  smith           4689              ET
        16  sunxxx01  smith           4661              ET
        17  sunxxx01  jones           4638   ########             T
        14  sunxxx01  miller          4379              ET      R
```

```
        13  sunxxx01  dba             3967   *adatst    AC
        12  sunxxx01  dba             3651   *adatst    EX,ET      E
        11  sunxxx01  dba             4025   DBADMIN    EX      RU

Selected: 11, Used: 11, Queue Size: 100
```

This display shows the current user queue entries.

- ID shows the internal user identification;
- NODE ID shows the node identification string;
- LOGIN ID shows the login identification string;
- ES ID is the process ID of the client process;

  Note: ES ID means "Environment Specific ID". This term was used, because in previous Adabas versions on Windows, instead of the process ID, a random number was used as the ES ID in order to avoid double usage of the same Adabas session ID - this was because on Windows, the process IDs could be reused after a short time. After adding a timestamp to the Adabas session ID, reusage of the same Adabas session ID can no longer happen, therefore the process ID can also be used as the ES ID on Windows. The timestamp is displayed only with ADAOPR DISPLAY=UQ_FULL.

- USER ID shows the user identification specified in Additions 1 in the Open command for the current Adabas session;

  Note: If you don't use the nucleus option OPEN_REQUIRED, the USER ID information is deleted following a non-activity timeout. When this happens, the USER ID is displayed as "#######". If the nucleus option OPEN_REQUIRED is used, not only the user information, but also the complete user queue element is deleted; this means that DISPLAY=UQ no longer displays such user queue elements.

- TYPE shows the user type:
  - AC access only user
  - ET ET user
  - EX exclusive update user
  - EX,ET exclusive update user with ET logic
  - UT utility user.
- STATUS shows the status of the user:
  - E user at ET status
  - I user session started with an implicit OPEN
  - R restricted file list
  - T user has received a time-out

- U user specific timeout interval value

Note: The description for the components of the Adabas session ID (Node ID, Login ID, ES ID and the timestamp not displayed by ADAOPR DISPLAY=UQ) is only correct if the function lnk_set_adabas_id is not used (see Command Reference). This function lets you define your own Adabas session IDs.

The final line of the display shows how many user queue entries were selected according to the currently active selection criteria, and how many entries are used in total.

Example: DISPLAY=UQ_FILES

```
adaopr: display=uq_files

                        ADANUC Version <version number>
        Database 76         User Files          on 19-JAN-2014 14:58:10


        Id  Type   Mode Files
        --  ----   ---- -----
        26  UT
        23  ET     UPD  11-12
        20  ET     UPD  11-12
        19  ET     UPD  11-12
        18  ET     UPD  11-12
        16  ET     UPD  11-12
        14  ET     UPD  11-12
        13  AC
        12  EX,ET EXU   14
        11  EX     ACC  11
                   EXU  13

Selected: 10, Used: 11, Queue Size: 100
```

This display shows the file lists for active users.

- ID shows the internal user identification;
- TYPE shows the user type (please refer to the DISPLAY=UQ example for more information).
- MODE shows the mode in which the files are being accessed:
  - ACC open for access
  - EXF open for exclusive access (no parallel access or update allowed)
  - EXU open for exclusive update (parallel access allowed)
  - UPD open for update
  - UTI open for exclusive access (no parallel access or update allowed)

  - UTO open for utilities only
- FILES shows the Adabas file list of the user entry. If the list is too large to be displayed in one line, several lines will be used: file numbers are not omitted.

The final line of the display shows how many user queue entries were selected according to the currently active selection criteria, and how many entries are used in total.

Example: DISPLAY=UQ_FULL

```
adaopr: disp=uq_full
                       ADANUC Version <version number>
        Database 36     Full User Queue Entry   on  3-SEP-2014 17:12:24

User Entry:  Id            :  8              ES Id        :  17937
             Node Id       :  sunada05       Login Id     :  smith
             User Id       :  *adaopr
             Timestamp Id  :   3-SEP-2014 17:12:18:182,671

             User Type     :  UT             User Status  :

Time Stamps: Session Start :  3-SEP-2014 17:12:17
             Trans. Start  :
             Last Activity :

Time Limits: TT            :             0   TNA          :             0

Resources:   ISN Lists     :             0   ISNs Held    :             0
             Open Files    :             0

Activity:    ADABAS Calls  :             1   Transactions :             0

Settings:    User Encoding :  UTF-8
-------------------------------------------------------------------------------

User Entry:  Id            :  6              ES Id        :  15808
             Node Id       :  sunada05       Login Id     :  jones
             User Id       :  JONES001
             Timestamp Id  :   3-SEP-2014 17:11:32:113,750

             User Type     :  ET             User Status  :

Time Stamps: Session Start :  3-SEP-2014 17:11:31
             Trans. Start  :  3-SEP-2014 17:11:56
             Last Activity :  3-SEP-2014 17:11:56

Time Limits: TT            :           300   TNA          :           300

Resources:   ISN Lists     :             0   ISNs Held    :             1
             Open Files    :             1
```

```
Activity:    ADABAS Calls  :             3   Transactions :             1

Settings:    User Encoding :  UTF-8
```

This display shows detailed information about user queue elements.

Additionally to the information shown by ADAOPR DISPLAY=UQ, the following information is shown:

- TIMESTAMP ID shows the timestamp added to the Adabas session ID to guarantee the uniqueness of the Adabas session ID;
- The timestamps show when the current Adabas user session was started, when the last transaction of the session was started, and when the last activity for the session was performed.;
- The time limits show the transaction time limit and the non-activity time limit defined for the Adabas user session;

  Note: Normally the time limits are the default values defined via ADANUC parameters, but it is possible to override these default values in the Open command of the Adabas user session.

- Resources shows the number of ISN lists currently active for the Adabas user session, the number of ISNs in the hold queue for the session, and the number of Adabas files in use in the session;
- Activity displays the number of Adabas calls and the number of transactions performed in the Adabas user session.
- Settings displays the default user encoding for W fields used in the current Adabas sessions as specified in the Open command of the session. If nothing was specified, the default UTF8 is used.

Example: DISPLAY=UQ_TIME_LIMITS

```
adaopr: display=uq_time_limits

                        ADANUC Version <version number>
        Database 76       User Time Limits      on 19-JAN-2010 14:58:10

TNAA Interval   :             00:15:00  TNAX Interval   :             00:15:00
TNAE Interval   :             00:15:00  TT   Interval   :             00:05:00

        Id St Limit   Timeout Interval     Remaining Time     Start Date/Time
        -- -- -----   ----------------     --------------     ---------------
        23    TNAE            00:15:00           00:15:00  19-JAN-2014 14:58:10
              TT              00:05:00
        22    TNAE            00:15:00           00:15:00  19-JAN-2014 14:58:10
              TT              00:05:00
```

```
        21    TNAE            00:15:00           00:15:00  19-JAN-2014 14:58:10
              TT              00:05:00           00:05:00  19-JAN-2014 14:58:10
        20    TNAE            00:15:00           00:15:00  19-JAN-2014 14:58:10
              TT              00:05:00           00:05:00  19-JAN-2014 14:58:10
        19    TNAE            00:15:00           00:15:00  19-JAN-2014 14:58:10
              TT              00:05:00
        18    TNAE            00:15:00           00:15:00  19-JAN-2014 14:58:10
              TT              00:05:00           00:04:50  19-JAN-2014 14:58:00
        17    TNAA            00:15:00           00:15:00  19-JAN-2014 14:58:10
        16    TNAE            00:15:00           00:15:00  19-JAN-2014 14:58:10
              TT              00:05:00           00:05:00  19-JAN-2014 14:58:10
        14    TNAE            00:15:00           00:15:00  19-JAN-2014 14:58:10
              TT              00:05:00           00:05:00  19-JAN-2014 14:58:10
        13    TNAA            00:15:00           00:10:01  19-JAN-2014 14:53:11
        12    TNAE            00:15:00           00:10:01  19-JAN-2014 14:53:11
              TT              00:05:00
        11  U TNAX            00:40:00           00:34:57  19-JAN-2014 14:53:07

Selected: 12, Used: 14, Queue Size: 100
```

This display shows the current timeout limits for the user queue entries.

- ID shows the internal user identification;
- ST shows the status of the entry. Possible values are:
  - U user specific timeout value
  - T a timeout is pending, response 9 has not been collected yet by the client.
- LIMIT describes the timeout type;
- TIMEOUT INTERVAL shows the current active timeout intervals.
- REMAINING TIME shows the amount of time remaining until the next timeout mark.
- START DATE/TIME shows the starting date and time of the entry.

The final line of the display shows how many user queue entries were selected according to the currently active selection criteria, and how many entries are used in total.

TCPENABLE and TCPDISABLE

```
TCPENABLE and TCPDISABLE
```

This control parameter enables or disables TCP connectivity by turning TCP functionality on or off during database operation. When executing ADAOPR TCPDISABLE, all open TCP connections will be closed. Any subsequent TCP/IP connection attempts will fail and return response code 148.

The default is TCPENABLE when using the **ADATCP Nucleus Parameter**.

ES_ID

```
ES_ID = number
```

This function influences the output of the DISPLAY options CQ, HQ, ICQ, UQ, UQ_FILES, UQ_FULL, UQ_TIME_LIMITS. Only entries with the specified environment-specific ID are displayed.

[NO]ET_SYNC

```
[NO]ET_SYNC
```

This option controls the behavior of the FEOF=PLOG function. It must be specified before specifying FEOF=PLOG. Refer to the FEOF=PLOG function for more information.

The default is NOET_SYNC.

This parameter setting is not supported when CONNECTION_STRING is used.

[NO]EVENTING

```
[NO]EVENTING
```

This starts and stops the Adabas Event Analytics for a running adanuc process. The adanuc process will start to generate events based on the Adabas Event Analytics configuration in the database INI file. If Adabas Event Analytics is configured to send the events to the Analytics Server, please make sure that the Analytics Server is started.

Note: If Adabas Event Analytics is not configured, the default events will be written to a NUCELG file located in the database directory.

The default is NOEVENTING.

This parameter setting is not supported when CONNECTION_STRING is used.

EXT_BACKUP

```
EXT_BACKUP = [PREPARE | CONTINUE | ABORT]
```

This function is used to backup a database using an external backup system, which can be considerably faster with very large databases than using ADABCK.

The keyword PREPARE prepares the database for backup. During this phase, the following restrictions apply:

- new transactions will be stalled
- no updating utility functions (e.g. ADADBM) can be started

- the functions SHUTDOWN, CANCEL, LOCK, STOPUSER, UNLOCK and FEOF=PLOG are not permitted once the EXT_BACKUP = PREPARE call has finished processing
- all non-activity timeout checks are disabled

The keyword CONTINUE is used to resume normal database operations following completion of the external backup. The following actions are performed:

- open a new PLOG with a new session number
- re-enable non-activity timeout checks
- re-enable update utilities
- wake up all waiting users (start of new transactions)

The keyword ABORT is used to abort an external backup for which a PREPARE has already been issued. In this case, the PLOG isn't switched and no checkpoint is written.

Caution: Take care to ensure that your external restore does not overwrite the protection logs created after the external backup. Without the protection logs, you cannot re-apply the changes perforrmed after the external backup with ADAREC REGENERATE.

This parameter setting is not supported when CONNECTION_STRING is used.

Example

The following scenario shows a backup and restore using a third-party backup tool (tar is not a real alternative, and is used for demonstration purposes only):

Dumping the database

```
adaopr db=37 ext_backup=prepare
%ADAOPR-I-STARTED,      13-DEC-2023 03:29:10, Version 7.2.0.0 (Linux 64Bit)

Database 37, startup at 13-DEC-2023 03:29:07
ADANUC Version 7.2.0.0, PID 2079603

%ADAOPR-I-EXTBPREP, preparing for external backup, 13-DEC-2023 03:29:10

%ADAOPR-I-TERMINATED,   13-DEC-2023 03:29:10, elapsed time: 00:00:00

adaopr db=37 ext_backup=continue
%ADAOPR-I-STARTED,      13-DEC-2023 03:31:24, Version 7.2.0.0 (Linux 64Bit)

Database 37, startup at 13-DEC-2023 03:29:07
ADANUC Version 7.2.0.0, PID 2079603
During ET Sync (phase 2), for external backup

%ADAOPR-I-EXTBCONT, continue from external backup, 13-DEC-2023 03:31:24

%ADAOPR-I-TERMINATED,   13-DEC-2023 03:31:24, elapsed time: 00:00:00
```

```
adarep
%ADAREP-I-STARTED,      13-DEC-2023 03:34:24, Version 7.2.0.0 (Linux 64Bit)
adarep: dbid=37
%ADAREP-I-DBON, database 37 accessed online
adarep: checkpoints=(12-dec-2023,14-dec-2023)


Name        Date/Time        Session   User Id / Function
----        ---------        -------   ------------------
SYNP   13-DEC-2023 03:27:57        1   ADAORD IMPORT=9
SYNP   13-DEC-2023 03:27:57        1   ADAORD IMPORT=14
SYNP   13-DEC-2023 03:27:57        1   ADAORD IMPORT=12
SYNP   13-DEC-2023 03:27:57        1   ADAORD IMPORT=11
SYNP   13-DEC-2023 03:27:57        1   ADAORD IMPORT=13
SYNC   13-DEC-2023 03:29:07        1   ADANUC 7.2.0.0
SYNX   13-DEC-2023 03:29:10        1   ADAOPR EXT_BACKUP STARTED
SYNX   13-DEC-2023 03:31:24        1   ADAOPR EXT_BACKUP
```

Restoring and recovering the database

```
% tar xvf $BACKUPDIR/backup.tar # external restore
% mv $ADADIR/db037/plog.0096 .  # Assume current directory is not $ADADIR/db037
% adastart 37
% adarep
adarep: checkpoints=(12-dec-2023,14-dec-2023)

Name        Date/Time        Session   User Id / Function
----        ---------        -------   ------------------
SYNC   13-DEC-2023 03:29:07        1   ADANUC 7.2.0.0
SYNX   13-DEC-2023 03:29:10        1   ADAOPR EXT_BACKUP STARTED
SYNX   13-DEC-2023 03:31:24        1   ADAOPR EXT_BACKUP
SYNC   13-DEC-2023 03:39:58        1   ADANUC CANCEL
SYNC   13-DEC-2023 03:40:12        2   ADANUC 7.2.0.0
adarep: q
%ADAREP-I-TERMINATED,   13-DEC-2023 03:48:50, elapsed time: 00:08:34
% setenv RECPLG plog.0096 # Set RECPLG for ADAREC (C shell)
% adarec dbid=37 regenerate=\* plog=96
```

After the restore, the checkpoint file contains the EXT_BACKUP STARTED checkpoint written by EXT_BACKUP=PREPARE, but not the checkpoints written by EXT_BACKUP=CONTINUE. The session number displayed for the current nucleus session is the number of the first PLOG that must be used for ADAREC REGENERATE for re-applying the changes done after the external backup.

The external backup is logged in the ADANUC log file

```
%ADANUC-I-DBSTART, Database 37, session 16 started, 14-NOV-2012 16:17:10
%ADANUC-I-EXTBPREP, preparing for external backup, 14-NOV-2012 16:18:30
%ADANUC-I-DBSTART, Database 37, session 17 started, 14-NOV-2012 16:18:45
%ADANUC-I-PLOGCRE, plog NUCPLG, file 'plogs/plog.0017' created
%ADANUC-I-EXTBCONT, continue from external backup, 14-NOV-2012 16:18:45
```

FEOF

```
FEOF = (keyword [,keyword])
```

In accordance with the keywords specified, the log file(s) are closed and a new log file is created.

Keyword	Meaning
CLOG	Closes command log file.
PLOG	Closes protection log file. This depends on the [NO]ET_SYNC option: If ET_SYNC is specified: The current protection log file (PLOG) will be closed when all currently active ET logic users have come to ET status, and a new PLOG is created with the next higher PLOG number. If ET_SYNC is not specified: As soon as the last block of the current PLOG extent is full, a new PLOG extent is created. The last block is moved to the new extent. **Note:** The protection log files are chained together with a "continuation pattern" to ensure they are processed in the correct order. Creating a new extent means: The current extent file will be extended with chaining information and at the same time creating the new extent file, leading to two extent files with the same modification date. See *Adabas Basics > Locations of Database Containers, Backup Files, and Protection Logs* in the *Adabas for Linux and Cloud* documentation for specific notes on protection logs.
ELOG	Closes event log file. The ELOG-keyword is only applicable if Adabas Analytics for LUW (EAL) is installed.
ALOG	Close NUCADT log file.

The FEOF command will be rejected if the keyword PLOG is used while running ADAREC RE-GENERATE = *.

This parameter setting is not supported when CONNECTION_STRING is used.

FILE

```
FILE = number
```

This influences the output of the DISPLAY options HQ, ICQ, UQ, UQ_FILES, UQ_FULL and UQ_TIME_LIMITS. Only entries related to the specified file number are displayed.

FREE_CLQ

```
FREE_CLQ
```

Normally, obsolete entries in the client queue are released automatically when the client queue is full. With ADAOPR FREE_CLQ, you can enforce the client queue cleanup before the client queue becomes full.

This parameter setting is not supported when CONNECTION_STRING is used.

ID

```
ID = number
```

This function influences the output of the DISPLAY options CQ, HQ, ICQ, UQ, UQ_FILES, UQ_FULL and UQ_TIME_LIMITS. Only entries related to the specified internal ID are displayed.

[NO]IO_TIME

```
[NO]IO_TIME
```

The parameter IO_TIME enables logging of the I/O times for the ASSO, DATA, WORK, NUCSRT and NUCTMP containers. The times are given in microseconds.

If logging of I/O times is already enabled, enabling it again resets all I/O time and I/O counter statistics.

The default is NOIO_TIME.

ISN

```
ISN = ( number [- number] [,number [- number] ] ... )
```

This function influences the output of the DISPLAY option HQ. Only entries related to the specified ISNs are displayed.

[UN]LOCK

```
[UN]LOCK = (number [,number]...)
```

The file(s) specified by the file number(s) are locked or unlocked. The specified files are locked for all non-utility use; Adabas utilities can use the file(s) normally. Specifying 0 means lock/unlock the complete database.

For users who have one or more files to be locked in their open file list, a STOP <user-ID> command is issued internally. Refer to the description of the ADAOPR STOP parameter for more details.

Notes:

1. You can also lock non-existent file numbers; if you subsequently create files with these numbers, the files are locked.
2. Locking a LOB file does not prevent users from storing LOB data in the LOB file; disabling the access to LOB data in the LOB file is part of locking the corresponding base file. Locking a LOB file is only useful if you plan to use this file number for a base file at some time in the future.
3. LOCK=0 is equivalent to OPTIONS=UTILITIES_ONLY plus stopping all users; UNLOCK=0 is equivalent to OPTIONS=NOUTILITIES_ONLY.
4. If files were locked on the file level, they must also be unlocked on the file level; UNLOCK=0 does *NOT* unlock such files.
5. If LOCK=() is used, it is equivalent to LOCK=0, which means that the full database will be locked. Empty brackets are considered to be equivalent to 0. If you use brackets, you should also use a file number.

This parameter setting is not supported when CONNECTION_STRING is used.

LOGGING

```
LOGGING = (keyword [,keyword]...)
```

This parameter starts command logging for the buffers specified in the list of keywords.

The following keywords can be used:

Keyword	Meaning
CB	Enables logging of control block
FB	Enables logging of format buffers
RB	Enables logging of record buffers
SB	Enables logging of search buffer
VB	Enables logging of value buffer
IB	Enables logging of ISN buffer

Keyword	Meaning
ABD	Enables logging of Adabas buffer descriptions
IO	Enables I/O list logging
NAT	Enables logging of Natural information (Requires additional configuration in NATPARM module. Please refer to the Natural documentation for further information.)
OFF	Stops logging of all buffers, but keeps the command log file open

If the nucleus was started with LOGGING=OFF and buffer logging is requested, then the CLOG file will be created.

This parameter setting is not supported when CONNECTION_STRING is used.

LOGIN_ID

```
LOGIN_ID = string
```

This function influences the output of the DISPLAY options CQ, HQ, ICQ, UQ, UQ_FILES, UQ_FULL and UQ_TIME_LIMITS. Only entries with a login ID that begin with the specified string will be selected. Please note that the string specification must be case sensitive. If you want to select explicitly a login ID shorter than 8 characters, but not other login IDs beginning with this login ID, you must add "^ " (Windows platforms) or "\ " (non-Windows platforms) to the login ID.

NISNHQ

```
NISNHQ = number
```

This parameter specifies the maximum number of records that can be placed into hold at any time by a single user.

If the specified value is less than the corresponding high-water value, a warning is issued.

The minimum value is 0, where 0 means unlimited.

This parameter setting is not supported when CONNECTION_STRING is used.

NODE_ID

```
NODE_ID = string
```

This function influences the output of the DISPLAY options CQ, HQ, ICQ, UQ, UQ_FILES, UQ_FULL and UQ_TIME_LIMITS. Only entries with a node ID that begin with the specified string will be selected. Please note that the string specification must be case sensitive. If you want to select explicitly a node ID shorter than 8 characters, but not other node IDs beginning with this node ID, you must add "^ " (Windows platforms) or "\ " (non-Windows platforms) to the node ID.

OPTIONS

```
OPTIONS = (keyword[,keyword])
```

The available keywords are:

Keyword	Meaning
[NO]LOCAL_UTILITIES	If LOCAL_UTILITIES is specified, the nucleus rejects all remote utility calls, i.e. the Adabas utilities cannot be run from a remote node across a network.
[NO]UTILITIES_ONLY	If UTILITIES_ONLY is selected, all calls other than for utilities will be rejected. Note, however, that this restriction only applies to new users; users who were already active when OPTIONS=UTILITIES_ONLY was specified can continue processing normally. If you want exclusive utility control over files or the entire database, use the LOCK function of ADAOPR instead.

These options can be disabled using the prefix 'NO', e.g. OPTIONS=NOUTILITIES_ONLY.

This parameter setting is not supported when CONNECTION_STRING is used.

RESET

```
RESET = keyword
```

RESET=HIGH_WATER resets the high water mark values to the value currently in use.

RESET=COMMANDS resets the command counts displayed by ADAOPR DISPLAY=COMMANDS.

RESET=RPL_STATS resets the replication statistic counters for all replicator threads, or in combination with the THREAD parameter for a specific thread only.This keyword is only relevant for customers who are using the Adabas Event Replicator with Adabas - Adabas Replication.

[NO]RESPONSE_ABORT

```
[NO]RESPONSE_ABORT
```

If response checking is enabled with the RESPONSE_CHECK parameter of ADAOPR, the RESPONSE_ABORT option determines whether the nucleus aborts when one of the specified responses occurs (RESPONSE_ABORT), or whether the nucleus resumes operation and a database section file is written to disk (NORESPONSE_ABORT).

The default is NORESPONSE_ABORT.

Refer to the RESPONSE_CHECK parameter for further information.

This parameter setting is not supported when CONNECTION_STRING is used.

RESPONSE_CHECK

```
RESPONSE_CHECK = [(number[-number][,number[-number]]...)]
```

This function enables the DBA to gather information if one of a list of Adabas response codes occurs. The information written may be used to analyze possible problems in the database's operation. If a response check for an Adabas response code is enabled, the database section file is written to disk if this response code occurs.

Depending on the setting of the RESPONSE_ABORT option, the nucleus either aborts or continues operation:

- if the RESPONSE_ABORT option is set, the database section file (Adabas.xxx.hh:mm:ss [Linux], or Adabas.xxx.hh-mm-ss [Windows]) is written to the database's default directory. The database section file is also called the CSA dump file. See ADANUC and the environment variable ADA_CSA_DUMP for more information.

  When the CSA dump file is written, the SMP dump file is also written (Linux platforms only); the name of the SMP dump file is SMPPOS.APP:hh:mm:ss.
- if the NORESPONSE_ABORT option is set (default setting), the nucleus continues running and the database section file (Adabas.xxx.RSPyyy.hh:mm:ss [Linux], or Adabas.xxx.RSPyyy.hh-mm-ss [Windows]) is written to the database's default directory. See ADANUC and the environment variable ADA_CSA_DUMP for more information. Only one dump is generated for one response code; if a response code occurs, the RESPONSE_CHECK option is deactivated for that response code, but if it has been activated for other response codes, it remains active for the other response codes.

Refer to the RESPONSE_ABORT action for further information.

By default, no response is trapped and the nucleus continues operation.

To disable response trapping, use "RESPONSE_CHECK =" without arguments.

This parameter setting is not supported when CONNECTION_STRING is used.

SET_FILE_STATS

```
SET_FILE_STATS = [(number[-number][,number[-number]]...)]
```

This function enables the file level I/O statistics for the specified files. Only these files will be displayed by DISPLAY = FILE_IO.

SHUTDOWN

```
SHUTDOWN
```

This function terminates the Adabas session normally. No new users are accepted. ET-user updating is continued until the end of the current transaction for each user. When all update activity has ended as described above, the Adabas session is terminated.

The communication link to the database is cut but the shared memory is still held. In this case, display functions are still possible with ADAOPR but parameter modification commands are no longer permitted.

This parameter setting is not supported when CONNECTION_STRING is used.

STATUS

```
STATUS = (keyword [,keyword] ,... )
```

This function influences the output of the DISPLAY parameter options HQ, ICQ, UQ, UQ_FILES, UQ_TIME_LIMITS, UQ_FULL. Only entries in the specified state will be displayed.

The valid keywords are:

Keyword	Meaning
[NO]TIMEOUT	User without or with "T" status.
[NO]ET_STATUS	Users at "ET" status with open transactions.
[NO]PENDING_ET	Users without or with "P" status.

STOP

```
STOP = (number[-number][,number[-number]]...)
```

This parameter terminates the user with the specified ID (internal identification). The ID can be retrieved with DISPLAY = UQ.

The message "Stop handling started for n users" is displayed, where "n" is the number of users who will be stopped.

Note: Utilities cannot always be stopped in this way.

The actions that Adabas takes when a user is stopped depend on the user type, and also whether the nucleus requires an explicit OP (open) command at the start of a user session, as shown in the following table.

The abbreviation SUQE used in the table means "Stop user queue element", and consists of the following actions: release all Command IDs, scratch the file list, scratch the user ID, scratch the user type, set response 9 for the next call.

User Type	Adabas Actions without ADANUC OPTIONS=OPEN_REQUIRED	Adabas Actions with ADANUC OPTIONS=OPEN_REQUIRED
ACC	For ID user: SUQE For non-ID user: session closed	session closed
ET, ET Status	For ID user: SUQE For non-ID user: session closed	session closed
ET, no ET Status	Backout transaction, SUQE	Backout transaction, session closed
EX	SUQE, CLSE checkpoint	session closed
EX, ET with ET status	SUQE, CLSE checkpoint	session closed
EX, ET, no ET status	Backout transaction, SUQE, CLSE checkpoint	Backout transaction, session closed
UT	session closed	session closed

If a STOP command is issued for a user while running

```
ADAREC REGENERATE = *
```

it will be rejected.

This parameter setting is not supported when CONNECTION_STRING is used.

STOPI

```
STOPI = number
```

Use the `STOPI` command to stop all users who have not executed a command during the specified number of seconds. Any open transactions of the stopped users will be backed out. A stopped user who returns (by sending a command) will receive response code 9.

This command does not stop EXF or UTI users.

THREAD

```
THREAD = number
```

This parameter is only relevant for customers who are using the Adabas Event Replicator with Adabas - Adabas Replication.

If you specify the parameter is before DISPLAY=RPL_STATS, the replication statistics are displayed only for the replicator thread specified. Thread numbering starts with 1. If you specify "THREAD=" without a number, the subsequent DISPLAY=RPL_STATS will display the statistics for all threads and the summary of all threads.

This parameter setting is not supported when CONNECTION_STRING is used.

TNAA

```
TNAA = number
```

This parameter sets the non-activity time limit (in seconds) for access-only users who have not explicitly specified a TNAA value in the OP command.

Note that the figure you specify for this parameter is only approximate. In any particular instance, the actual amount of time can vary from this value by up to 10 seconds.

The minimum value is 20, the maximum value is 2592000.

This parameter setting is not supported when CONNECTION_STRING is used.

TNAE

```
TNAE = number
```

This parameter sets the non-activity time limit (in seconds) for ET logic users who have not explicitly specified a TNAE value in the OP command.

Note that the figure you specify for this parameter is only approximate. In any particular instance, the actual amount of time can vary from this value by up to 10 seconds.

The minimum value is 20, the maximum value is 2592000.

This parameter setting is not supported when CONNECTION_STRING is used.

TNAX

```
TNAX = number
```

This parameter sets the non-activity time limit (in seconds) for EXU and EXF users who have not explicitly specified a TNAX value in the OP command.

Note that the figure you specify for this parameter is only approximate. In any particular instance, the actual amount of time can vary from this value by up to 10 seconds.

The minimum value is 20, the maximum value is 2592000.

This parameter setting is not supported when CONNECTION_STRING is used.

TT

```
TT = number
```

This parameter sets the transaction time limit for ET logic users who have not explicitly specified a TT value in the OP command.

If the specified value is less than the corresponding high-water value, a warning is issued.

Note that the figure you specify for this parameter is only approximate. In any particular instance, the actual amount of time can vary from this value by up to 10 seconds.

The minimum value is 20, the maximum value is 2592000.

This parameter setting is not supported when CONNECTION_STRING is used.

USER_ID

```
USER_ID = string
```

This function influences the output of the DISPLAY parameter options CQ, HQ, ICQ, UQ, UQ_FILES, UQ_TIME_LIMITS, UQ_FULL. Only entries with a user ID that begin with the specified string will be selected. Please note that the string specification must be case sensitive. If you want to select explicitly a user ID shorter than 8 characters, but not other user IDs beginning with this user ID, you must add "^ " (Windows platforms) or "\ " (non-Windows platforms) to the user ID.

WCHARSET

```
WCHARSET = <ICU encoding>
```

This parameter specifies the default encoding for W fields for user sessions. This encoding is used if no other encoding is specified in the record buffer of the OP call, or in the format buffer of L or A/N calls.

This parameter setting is not supported when CONNECTION_STRING is used.

Example

```
adanuc: wcharset=utf-16be
```

WRITE_LIMIT

```
WRITE_LIMIT = [number]
```

This parameter specifies the percentage of modified blocks permitted in the buffer pool before an implicit buffer flush is taken.

Note that "WRITE_LIMIT=" (keeping the equals sign but omitting the number) is equivalent to "WRITE_LIMIT=0".

Supported values are 1-50; the default value is 50. For compatibility reasons, values of 0 and 51-70 are also allowed - they are equivalent to 50.

This parameter setting is not supported when CONNECTION_STRING is used.

RBAC Protection

The RBAC protection allows you to secure access to specific functions in this utility. See the list of the secured functions in *Adabas Role-Based Security (ADARBA)* in *Adabas Security Features* in the Adabas documentation.

17 ADAORD (Reorder Database Or Files, Export/Import Files)

- Functional Overview 264
- Procedure Flow 265
- Checkpoints 267
- Control Parameters 267
- Restart Considerations 276
- Examples 276

This chapter describes the utility "ADAORD".

Functional Overview

The reorder utility ADAORD provides functions to reorganize a whole database (REORDER) and to migrate files between databases (EXPORT/IMPORT).

Depending on the function selected, ADAORD produces or requires a sequential file (ORDEXP).

The main reasons for running ADAORD are:

- To change the layout of a complete database. This includes increasing or decreasing the maximum number of files permitted;
- To change the space allocation or placement of a file, to reduce the number of logical extents assigned to its index, Address Converter or Data Storage and to change or re-establish the padding factors;
- To create one or more test files that all contain the same data. This procedure requires a file to be exported and then imported using a different file number;
- To archive and subsequently reestablish a file, independent of its original placement and the database device types used.

When exporting files from a database, the Adabas nucleus is not required. If a system file is processed, the nucleus must be inactive. For detailed information, please refer to the table of nucleus requirements.

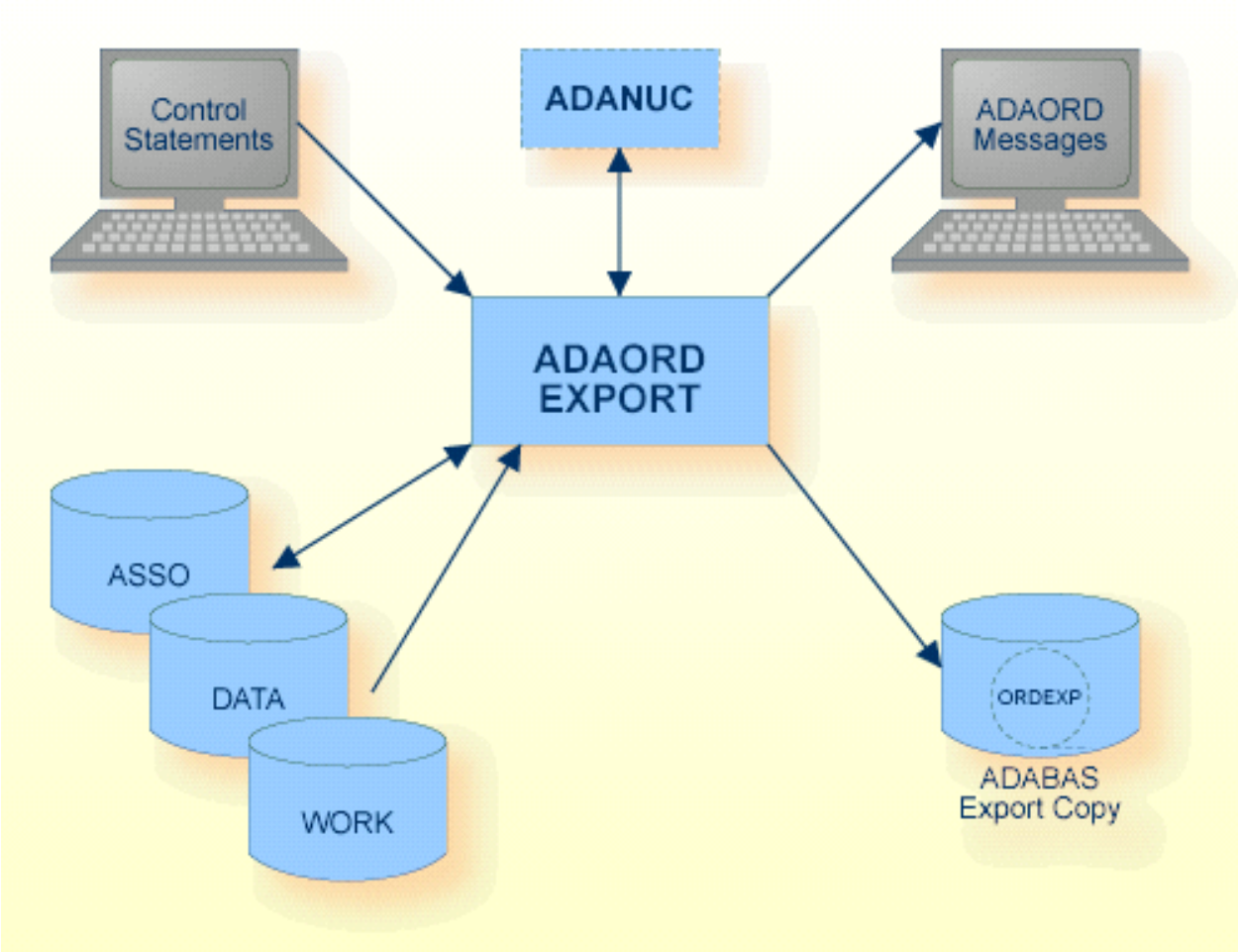
When importing files into a database, the Adabas nucleus is not required to be active. The nucleus may be either started or shut down during this procedure.

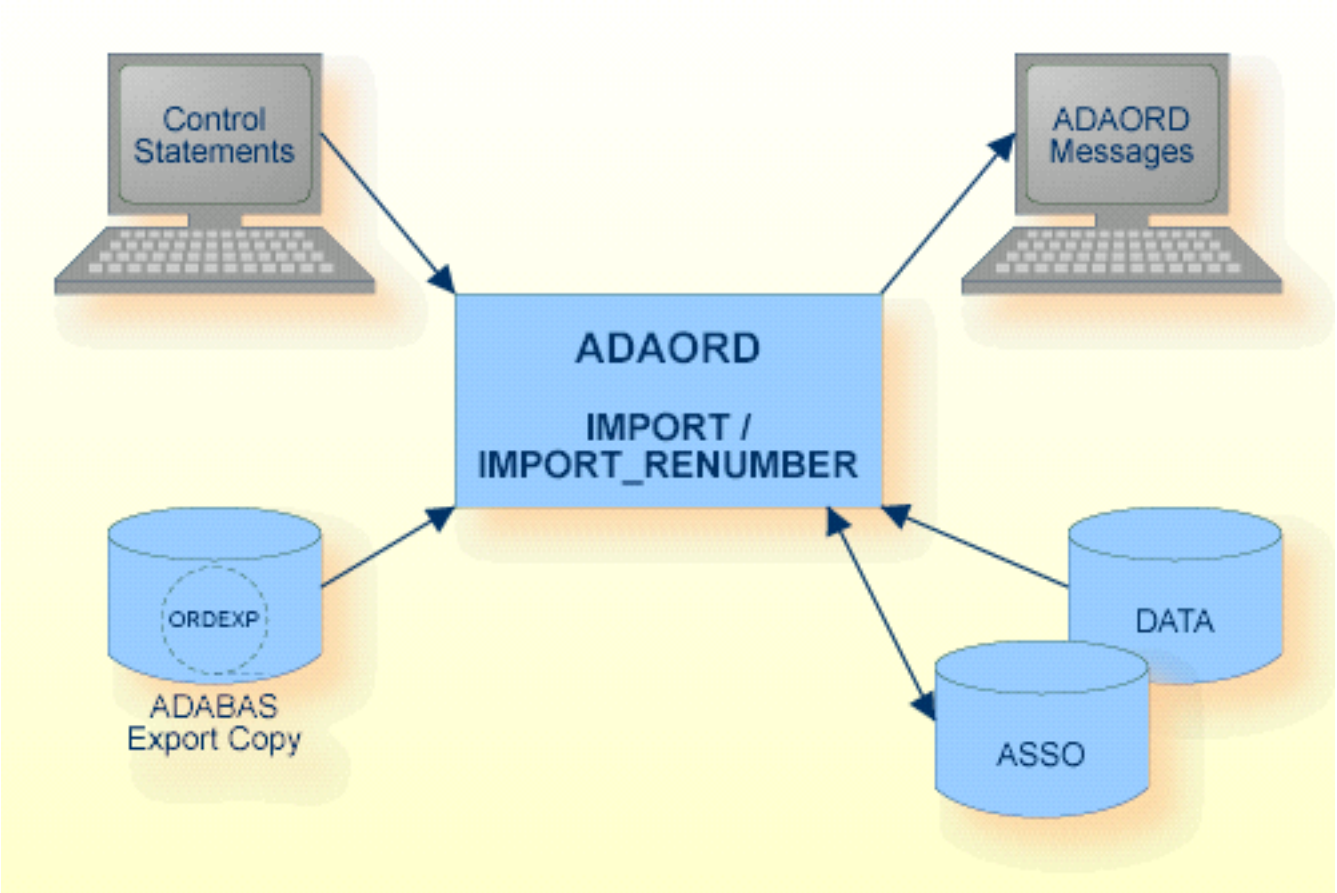
When reordering the database, the nucleus must be inactive.

Note: The IMPORT and IMPORT_RENUMBER functions can process export files created with earlier Adabas versions, but not export files created with later Adabas versions.

This utility is a single-function utility. For more information about single- and multi-function utilities, see *Adabas Basics*, *Using Utilities* in the Adabas documentation.

Procedure Flow





Data Set	Environment Variable / Logical Name	Storage Medium	Additional Information
Associator	ASSOx	Disk	
Data storage	DATAx	Disk	
Export copy	ORDEXP	Disk (* see note)	Export (out), Reorder (in/out), other functions (in)
Control statements	stdin		Utilities Manual
ADAORD messages	stdout		Messages and Codes
Work storage	WORK1	Disk	

Note: (*) A named pipe cannot be used for this sequential file. See *Adabas Basics, Using Utilities* in the Adabas documentation for more information.

Checkpoints

The following table shows the nucleus requirements for each function and the checkpoints written:

Function	Nucleus must be active	Nucleus must NOT be active	Nucleus is NOT required	Checkpoint written
CONTENTS			X	-
EXPORT		X(* see note)	X	SYNX
IMPORT		X(* see note)	X	SYNP
IMPORT_RENUMBER		X(* see note)	X	SYNP
REORDER		X	X	SYNP

Note: (*) When processing an Adabas system file

In the case of the EXPORT function, ADAORD writes a single checkpoint and removes the UCB entry when all of the specified files have been exported and the sequential output file (ORDEXP) has been closed.

In the case of the IMPORT function, ADAORD writes a checkpoint and informs the nucleus that the file has been loaded every time a file is successfully imported.

The UCB entry is removed when all of the specified files have been imported. When the utility is executed offline, writing multiple checkpoints increases the probability of a checkpoint block (CPB) overflow. The checkpoint file should, therefore, always be present to allow the Adabas nucleus to be started in order to empty the CPB.

In the case of the REORDER function, ADAORD writes a single checkpoint and removes the UCB entry when the function terminates.

Control Parameters

The following control parameters are available:

```
       CONTENTS

       DBID = number

       EXPORT = (number[-number][,number[-number]]...)
                [,FDT]
D               [,SORTSEQ = ({descriptor_name|ISN|PHYSICAL},...)]
```

```
FILES = (number[[-number], number[-number]] ...)

IMPORT = (number[-number][,number[-number]]...)
         [,ACRABN = number]
         [,ASSOPFAC = number]
         [,DATAPFAC = number]
         [,DSRABN = number]  [,DSSIZE = number[B|M] ]
         [,LOBACRABN = number]
         [,LOBDSRABN = number]
         [,LOBNIRABN = number]
         [,LOBSIZE = numberM]
         [,LOBUIRABN = number]
         [,MAXISN = number]
         [,NIRABN = number|(number,number)]
         [,NISIZE = number[B|M]|(number[B|M],number[B|M])]
         [,UIRABN = number|(number,number)]
         [,UISIZE = number[B|M]|(number[B|M],number[B|M])]

IMPORT_RENUMBER = (number, number[,number])
                  [,ACRABN = number]
                  [,ASSOPFAC = number]
                  [,DATAPFAC = number]
                  [,DSRABN = number]  [,DSSIZE = number[B|M] ]
                  [,LOBACRABN = number]
                  [,LOBDSRABN = number]
                  [,LOBNIRABN = number]
                  [,LOBSIZE = numberM]
                  [,LOBUIRABN = number]
                  [,MAXISN = number]
                  [,NIRABN = number|(number,number)]
                  [,NISIZE = number[B|M]|(number[B|M],number[B|M])]
                  [,UIRABN = number|(number,number)]
                  [,UISIZE = number[B|M]|(number[B|M],number[B|M])]

REORDER = *
```

CONTENTS

```
CONTENTS
```

This function displays the list of files contained in the sequential output file (ORDEXP) created by a previous run of the EXPORT function.

DBID

```
DBID = number
```

This parameter selects the database to be used.

EXPORT

```
EXPORT = (number[-number][,number[-number]]...)
         [,FDT]
         [,SORTSEQ = ({descriptor_name|ISN|PHYSICAL},...)]
```

This function exports (copies) one or more files from the database to a sequential output file (ORDEXP). In order to maintain referential integrity in the export copy, all files that are connected via referential constraints to a specified file are also exported. The file numbers specified are only taken into consideration if they are the file numbers of base files; the corresponding LOB files for the selected files are exported automatically with the base files without having to be specified. If files have referential constraints, and the base files are not specified during the export, the dependent files will need to be specified during the import. An EXPORT consists of copying each file's Data Storage, together with the information that is required to reestablish its index. All of the files to be processed are written to ORDEXP in the sequence in which they are specified. Overlapping ranges and numbers are removed.

Note: If the checkpoint file is included in the file list, it will be processed last.

FDT

This parameter displays the FDT of the file to be processed.

SORTSEQ = ({descriptor_name|ISN|PHYSICAL} ,...)

This parameter controls the sequence in which the Data Storage is processed. If specifies either the field name of a descriptor, subdescriptor or superdescriptor, or the keyword 'ISN' or 'PHYSICAL'.

The default is physical sequence.

The following values can be specified:

Value	Sequence
descriptor_name	If the name of a descriptor, sub- or superdescriptor is specified, the data records are processed in ascending logical sequence of the descriptor values to which the field name refers. A field with the MU, MC or NU option or one that is contained in a periodic group or a sub- or superdescriptor derived from such a field must not be specified. Logical sequence can be used only if a single file has been selected.
ISN	If ISN is specified, the data records are processed in ascending ISN sequence.
PHYSICAL	If PHYSICAL is specified or if the SORTSEQ parameter is omitted, the data records are processed in the physical sequence in which they are stored in the Data Storage.

The performance when processing in logical sequence and ISN sequence is better if the database is online (provided that the buffer pool is large enough).

If one value is specified for SORTSEQ, that value is valid for all files. If more than one value is specified, the number of values must be the same as the number of file ranges specified for the EXPORT parameter. In this case, the first file range is exported in the first specified sort sequence, the second file range is exported in the second specified sort sequence, and so on.

Example

```
EXPORT = (1, 20-30, 40)
SORTSEQ = (AA, PHYSICAL, ISN)
```

File 1 is exported in the sequence of descriptor AA, files 20-30 are exported in physical sequence and file 40 is exported in ISN sequence.

FILES

```
FILES = (number[[-number], number[-number]] ...)
```

This parameter is used to display information concerning the status of the specified files contained on the sequential input file (ORDEXP).

IMPORT

```
IMPORT = (number[-number][,number[-number]]...)
         [,ACRABN = number]
         [,ASSOPFAC = number]
         [,DATAPFAC = number]
         [,DSRABN = number] [,DSSIZE = number[B|M] ]
         [,LOBACRABN = number]
         [,LOBDSRABN = number]
         [,LOBNIRABN = number]
         [,LOBSIZE = numberM]
         [,LOBUIRABN = number]
```

```
          [,MAXISN = number]
          [,NIRABN = number|(number,number)]
          [,NISIZE = number[B|M]|(number[B|M],number[B|M])]
          [,UIRABN = number|(number,number)]
          [,UISIZE = number[B|M]|(number[B|M],number[B|M])]
```

This function imports one or more files into a database, using the data on the sequential file (ORDEXP) produced by a previous run of ADAORD. In order to maintain referential integrity, all files connected via referential constraints to a specified file are also imported. The file numbers specified are only taken into consideration if they are the file numbers of base files; the corresponding LOB files for the selected files are imported automatically with the base files without having to be specified. If there are files that are connected via referential constraint, the base files need to be specified during the import. The file numbers specified are sorted into ascending sequence. Overlapping ranges and numbers are removed.

The file numbers specified must not be loaded in the database.

By default, ADAORD controls the file placement and the allocation quantities. The parameters that can be used to overwrite these defaults may be used only if a single file has been selected.

Please refer to the **IMPORT_RENUMBER** function for the description of the parameters.

IMPORT_RENUMBER

```
IMPORT_RENUMBER = (number, number[,number])
                  [,ACRABN = number]
                  [,ASSOPFAC = number]
                  [,DATAPFAC = number]
                  [,DSRABN = number] [,DSSIZE = number[B|M] ]
                  [,LOBACRABN = number]
                  [,LOBDSRABN = number]
                  [,LOBNIRABN = number]
                  [,LOBSIZE = numberM]
                  [,LOBUIRABN = number]
                  [,MAXISN = number]
                  [,NIRABN = number|(number,number)]
                  [,NISIZE = number[B|M]|(number[B|M],number[B|M])]
                  [,UIRABN = number|(number,number)]
                  [,UISIZE = number[B|M]|(number[B|M],number[B|M])]
```

This function imports a file into a database, using the data on the sequential file (ORDEXP) produced by a previous run of ADAORD. It is not possible to import and renumber a file that is connected to another file via referential integrity. Constraints must either dropped before exporting the files, or the files must be imported without renumbering and be renumbered later (ADADBM RENUMBER). The first number given defines the base file to be imported, and the second number is the new file number to be assigned to the file. The third, optional number is the new file number for the LOB file. If the third number is not specified, the LOB file number (if it exists) remains unchanged.

The new file number must not be loaded in the database.

Unless otherwise specified, ADAORD controls the file placement and the allocation quantities.

ACRABN = number

This parameter specifies the RABN at which the space allocation for the Address Converter (AC) is to start.

If this parameter is omitted, ADAORD assigns the starting RABN.

ASSOPFAC = number

This parameter specifies the new padding factor to be used for the file's index. The number specified is the percentage of each index block which is not to be used by ADAORD or a subsequent run of the mass update utility ADAMUP. This padding area is reserved for future use if additional entries have to be added to the block by the Adabas nucleus. This avoids the necessity of having to relocate overflow entries to another block.

A value may be specified in the range of 0 to 95.

A small padding factor (0 to 10) should be specified if little or no descriptor updating is expected. A larger padding factor (10 to 50) should be specified if a large amount of descriptor updating is expected in which new descriptor values are created.

If this parameter is omitted, the current padding factor in effect for the file's index is used.

DATAPFAC = number

This parameter specifies the new padding factor to be used for the file's Data Storage. The number specified is the percentage of each data block which is not to be used by ADAORD. This padding area is reserved for future use if any record in a block requires additional space as result of record updating by the Adabas nucleus. This avoids the necessity of having to relocate overflow entries to another block.

A value may be specified in the range of 0 to 95.

A small padding factor (0 to 10) should be specified if there is little or no record expansion. A larger padding factor (10 to 50) should be specified if there is a large amount of record updating which will cause expansion.

If this parameter is omitted, the current padding factor in effect for the file's Data Storage is used.

DSRABN = number

This parameter specifies the RABN at which the space allocation for the file's Data Storage (DS) is to start.

If this parameter is omitted, ADAORD assigns the start RABN.

DSSIZE = number[B|M]

This parameter specifies the number of blocks (B) or megabytes (M) to be initially assigned to the file's Data Storage (DS). By default, the size is given in megabytes.

If this parameter is omitted, ADAORD calculates the size based on the old number of blocks allocated and the difference between the old and new padding factor.

LOBACRABN=number

This parameter specifies the RABN at which the space allocation for the LOB file's Address Converter (AC) is to start.

If this parameter is omitted, ADAORD assigns the start RABN.

LOBDSRABN=number

This parameter specifies the RABN at which the space allocation for the LOB file's Data Storage (DS) is to start.

If this parameter is omitted, ADAORD assigns the start RABN.

LOBNIRABN=number

This parameter specifies the RABN at which the space allocation for the LOB file's Normal Index (NI) is to start.

If this parameter is omitted, ADAORD assigns the start RABN.

LOBSIZE=numberM

This parameter specifies the number of megabytes to be initially assigned to the LOB file's Data Storage (DS). The AC size, NI size and UI size for the LOB file are derived from this size.

If this parameter is omitted, ADAORD calculates the size based on the old number of blocks allocated and the difference between the old and new padding factor.

LOBUIRABN=number

This parameter specifies the RABN at which the space allocation for the LOB file's Upper Index (UI) is to start.

If this parameter is omitted, ADAORD assigns the start RABN.

MAXISN = number

This parameter specifies the highest permissible ISN for the file. ADAORD uses this parameter to determine the amount of space to be allocated for the file's Address Converter (AC).

Because there is no automatic extension of the initial allocation, a value that is smaller than the file's current first free ISN will cause ADAORD to terminate execution and return an error status if there are ISNs outside the Address Converter.

If this parameter is omitted, the value of MAXISN currently in effect for the file's Address Converter is used.

A contiguous-best-try allocation is used.

NIRABN = number|(number,number)

This parameter specifies the RABN(s) at which the space allocation for the file's Normal Index (NI) is to start. Adabas usually stores small descriptor values (<= 253 bytes) in small index blocks (block size < 16 KB) and large descriptor values in large index blocks (block size >= 16 KB. For this reason, it is possible to specify 2 RABNs - if you specify 2 RABNs, one must have a block size < 16 KB, and the other must have a block size >=16 KB.

If this parameter is omitted, ADAORD assigns the start RABN.

NISIZE = number[B|M]|(number[B|M],number[B|M])

This parameter specifies the number of blocks (B) or megabytes (M) to be initially assigned to the file's Normal Index (NI). By default, the size is given in megabytes. If two values are specified and the NIRABN parameter is also specified, the first value corresponds to the first value of the NIRABN parameter, and the second value corresponds to the second value of the NIRABN parameter. If two values are specified and the NIRABN parameter is not specified, the first value specifies the size of small normal index blocks (< 16 KB), and the second value specifies the size of large NI blocks (>= 16 KB).

If this parameter is omitted, ADAORD calculates the size based on the old number of blocks allocated and the difference between the old and new padding factor.

UIRABN = number|(number,number)

This parameter specifies the RABN(s) at which the space allocation for the file's Upper Index (UI) is to start. Adabas usually stores small descriptor values (<= 253 bytes) in small index blocks (block size < 16 KB) and large descriptor values in large index blocks (block size >= 16 KB. For this reason, it is possible to specify 2 RABNs - if you specify 2 RABNs, one must have a block size < 16 KB, and the other must have a block size >=16 KB.

If this parameter is omitted, ADAORD assigns the start RABN.

UISIZE = number[B|M]|(number[B|M],number[B|M])

This parameter specifies the number of blocks (B) or megabytes (M) to be initially assigned to the file's Upper Index (UI). By default, the size is given in megabytes. If two values are specified and the UIRABN parameter is also specified, the first value corresponds to the first value of the UIRABN parameter, and the second value corresponds to the second value of the UIRABN parameter. If two values are specified and the UIRABN parameter is not specified, the first value specifies the size of small upper index blocks (< 16 KB), and the second value specifies the size of large UI blocks (>= 16 KB).

If this parameter is omitted, ADAORD calculates the size based on the old number of blocks allocated and the difference between the old and new padding factor.

REORDER

```
REORDER = *
```

This function is used to change the layout of a whole database. It rearranges the database's global areas, eliminates fragmentation in the DSST and the files' Address Converter, Data Storage, Normal Index and Upper Index extents by physically changing their placement. It also re-establishes the files' padding factors. Exclusive control of the database container files is required.

A REORDER database implicitly exports the files, deletes them from the database and then re-imports them. The sequential file (ORDEXP) that is created during the REORDER is kept.

Note: ADAORD uses a best-fit algorithm for the allocation of the disk space for the files. Therefore, it may occur that the first container of a given type remains empty if it is followed by another container with adequate block size which is smaller than the first one.

Restart Considerations

ADAORD has no restart capability.

An abnormally terminated EXPORT must be rerun from the beginning.

An abnormally terminated IMPORT of one or more files will result in lost RABNs for the last file being imported. These RABNs can be recovered by executing ADADBM's RECOVER function. The files preceding the one being processed when the interrupt occurred will be available in the database. Therefore, the IMPORT function should be rerun starting with the file number at which the interrupt occurred.

An abnormally terminated IMPORT_RENUMBER will result in lost RABNs for the file being imported. These RABNs can be recovered by executing ADADBM's RECOVER function. The IMPORT_RENUMBER function has to be rerun from the beginning.

An abnormally terminated REORDER at the database level may result in a database that cannot be accessed if the interrupt occurred while reordering the database's global areas (GCB, FST, DSST, etc.). In this case, either a new empty database has to be created using ADAFRM or the old database has to be reestablished from an Adabas backup copy, using ADABCK's RESTORE database function. If the interrupt occurred during the re-import phase, it will result in lost RABNs for the last file being imported. These RABNs can be recovered by executing ADADBM's RECOVER function. The files preceding the one being processed when the interrupt occurred will be available in the database. The remaining files can be obtained from the sequential work file (ORDEXP) by using ADAORD's IMPORT function.

Examples

In the examples below, the files 1, 2, 4, 6, 7, 8, 10, 11, 12 and 25 are loaded in database 1. Database 2 contains files 3, 6 and 11.

Example 1

```
adaord: dbid = 1
adaord: export = (1-4,7,10-25)
```

Files 1, 2, 4, 7, 10, 11, 12 and 25 are exported from database 1.

Example 2

```
adaord: dbid = 2
adaord: import = (1-10,12)
```

Files 1, 2, 4, 7, 10 and 12 are imported into database 2. It is not possible to specify "`import=(1-12)`" because ADAORD first checks to see if one of the files to be imported is already loaded , and if it is, then the whole import is rejected - in this case file 11 is already loaded.

Example 3

```
adaord: dbid = 2
adaord: import_renumber = (11,19), acrabn = 131, datapfac = 20
```

File 11 is imported into database 2 using a new file number of 19 (because 11 is already in use). The file's Address Converter (AC) is to be allocated at ASSO RABN 131. The new padding factor for the Data Storage (DS) is 20 percent.

Example 4

```
adaord: dbid = 1
adaord: reorder = *
```

The whole database is reordered.

18 ADAPLP (Protection Log Printout)

- Functional Overview 280
- Procedure Flow 281
- Checkpoints 282
- Control Parameters 282
- ADAPLP Output 291

This chapter describes the utility "ADAPLP".

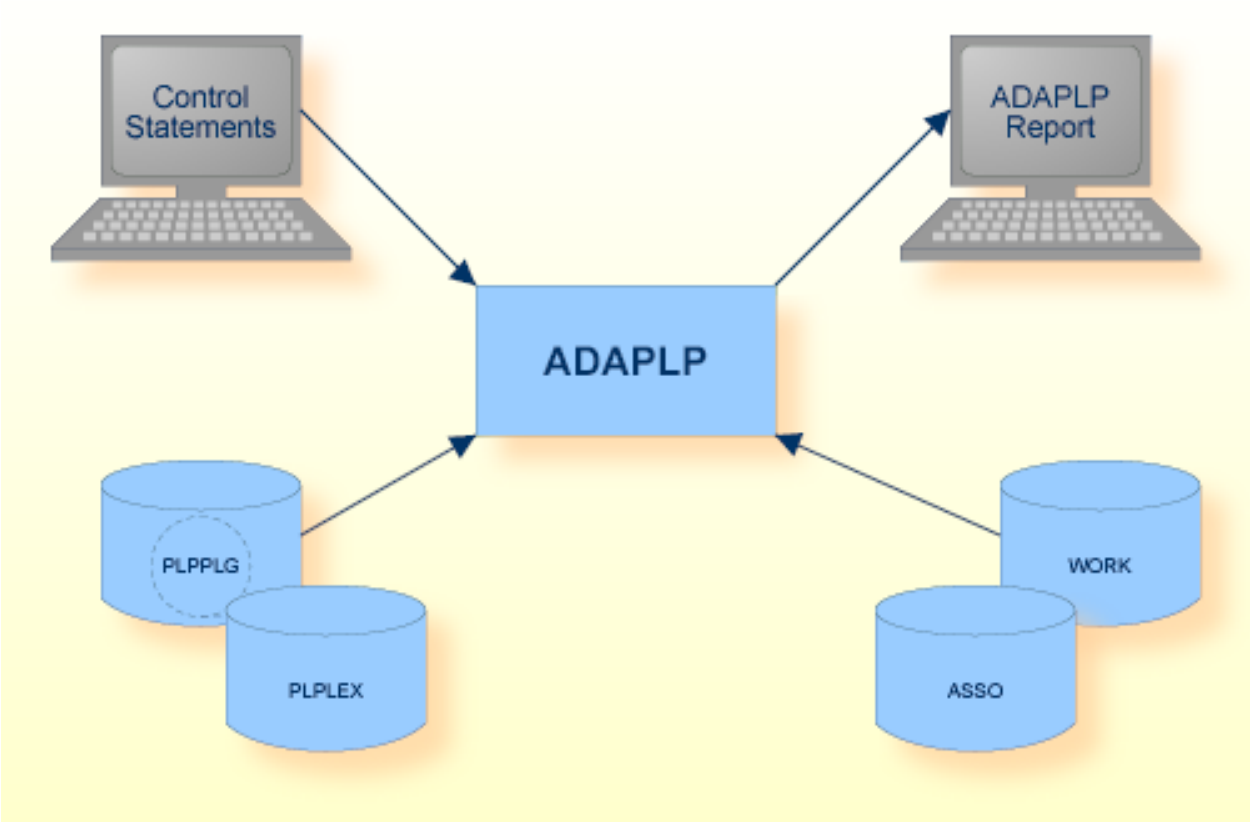
Functional Overview

The ADAPLP utility prints the Protection Log or WORK.

This utility is a multi-function utility. For more information about single- and multi-function utilities, see *Adabas Basics*, *Using Utilities* in the Adabas documentation.

Note: LOB values are split into several records in a LOB file; when LOB values are stored in the database, the Protection Log contains the log records for the modifications of the LOB file. This means that ADAPLP does not display the LOB values as one value, but rather it displays the modifications of the corresponding records in the LOB file instead. It is not possible to decompress the LOB file records because the LOB records are too large to fit into one block, and the continued Protection Log records cannot be decompressed.

Procedure Flow



Data Set	Environment Variable / Logical Name	Storage Medium	Additional Information
Associator	ASSOx	Disk	
Protection log	PLPPLG	Disk	Utilities Manual
Protection log (last extent)	PLPLEX	Disk	Only required if you process a PLOG with an extension count > 1 and if you use the DECOMPRESS or DELTA option: you must provide the last PLOG extent before the PLOG extent to be processed.
Control statements	stdin		Utilities Manual
ADAPLP report	stdout		Messages and Codes
Work storage	WORK1	Disk	

The sequential file PLPPLG can have multiple extents. For information about sequential files with multiple extents, see *Adabas Basics*, *Using Utilities* in the Adabas documentation.

Checkpoints

The utility writes no checkpoints.

Control Parameters

The following control parameters are available:

```
M    DATASET = keyword

     DBID = number

D    [NO]DECOMPRESSED

     DELTA

D    [NO]DUMP

     FILES = (number [-number][,number [-number]]...)

D    [NO]HEADER

     INTERNAL_ID = number

     ISN = (number [,number] ... )

     MODIFIED_RABN = number

     NOFILETYPE

     NONULL

     PLOG = (number [,number])

M    RABN = {*|number[-number]}

     RECORD ={*| (number [- number]  [, number  [- number]]...) }

     SEQ = number

D    [NO]SHORT

     THREAD = number

     TSN = number
```

```
D    TYPE = (keyword [,keyword]...)

     USER_ID = string
```

DATASET

```
DATASET = keyword
```

This parameter selects the file containing the Protection Log information to be processed. The keyword can take the values PLOG or WORK.

DBID

```
DBID = number
```

This parameter selects the database to be used.

This parameter must be used when DATASET=WORK is requested. This parameter must be the first parameter to be specified. Otherwise, if this parameter is not specified, the DBID stored in the PLOG is used.

[NO]DECOMPRESSED

```
[NO]DECOMPRESSED
```

This option indicates whether for each selected DATA record from a protection log, one line per field is printed with the field name and its decompressed value in hex (DECOMPRESSED) or not (NODECOMPRESSED).

For an inserted record, an after image containing the field values of the record after the insert is displayed.

For an updated record, a before image containing the field values before the update, and an after image containing the field values of the record after the update are displayed.

For a deleted record, a before image containing the field values before the delete operation is displayed.

If you specify DECOMPRESSED and NONULL, no output is produced for the following:

- Fields with NU or NC option with null-value;
- MU fields with NU or NC option without a value that is not the null-value;
- PE groups containing only fields with NU or NC option without a value that is not the null-value;
- Group names if the group is not PE.

The default is NODECOMPRESSED.

Note: Decompression (DECOMPRESSED output) is not possible for CONTINUED records. CONTINUED records are created if a PLOG record plus the block header is larger than 32 KB in the PLOG or larger than the block size used for WORK.

Example output for DECOMPRESSED (without NONULL option)

```
>>> After Image <<<


Length = 20, ISN = 2

Field    : AA: ^30372E31322E3034
Group    : AB
Field    : AC: ^2020202020202020202020202020202020202020
Field    : AE: ^2020202020202020202020202020202020202020
Field    : AD: ^2020202020202020202020202020202020202020
Field    : AF: ^20
Field    : AG: ^20
Field    : AH: ^0000000C
Group    : A1
MU-field : AI, count = ^01
           AI(  1): ^2020202020202020202020202020202020202020
Field    : AJ: ^2020202020202020202020202020202020202020
Field    : AK: ^20202020202020202020
Field    : AL: ^202020
Group    : A2
Field    : AN: ^202020202020
Field    : AM: ^2020202020202020202020202020
Field    : AO: ^202020202020
Field    : AP: ^2020202020202020202020202020202020202020202020202020
PE-group : AQ, count = ^01
  PE index (1)
Field    : AR: ^202020
Field    : AS: ^000000000C
MU-field : AT, count = ^01
           AT(  1): ^000000000C
End of PE-group : AQ
Group    : A3
Field    : AU: ^3030
Field    : AV: ^3030
PE-group : AW, count = ^01
  PE index (1)
Field    : AX: ^3030303030303030
Field    : AY: ^3030303030303030
End of PE-group : AW
MU-field : AZ, count = ^01
           AZ(  1): ^202020
```

Example output for DECOMPRESSED (with NONULL option)

```
>>> After Image <<<


Length = 15, ISN = 2

Field    : AA: ^30372E31322E3034
Field    : AF: ^20
Field    : AG: ^20
```

DELTA

```
DELTA
```

This parameter indicates that only changed fields after an update are displayed.

For an inserted record, the same output is produced as with the options DECOMPRESSED, NONULL.

For an updated record, a Delta containing the modified field values is displayed. Note that for MU/PE fields, the value count displayed can be smaller than the displayed MU/PE indices if the MU/PE count has been decreased - this is because all field values have been set to the null value.

If a record is deleted, no output is produced, however, you can see the deletion if you display protection log entries of the type CE.

Note: Decompression (DELTA output) is not possible for CONTINUED records. CONTINUED records are created if a PLOG record plus the block header is larger than 32 KB in the PLOG or larger than the block size used for WORK.

[NO]DUMP

```
[NO]DUMP
```

This option indicates whether the variable part of a Protection Log record is included in the printout (DUMP) or not (NODUMP).

If DUMP is specified, the variable part of each Protection Log record is displayed in both hexadecimal and uninterpreted ASCII format.

DUMP implicitly resets SHORT.

The default is NODUMP.

FILES

```
FILES = (number [-number][,number [-number]]...)
```

The Protection Log records are only displayed if they belong to the file(s) specified by this parameter.

Only records of the types DA, DV, EXT, INDEX and FCB are displayed.

Please refer to the tables at the end of this section for a description of the various types of Protection Log records.

[NO]HEADER

```
[NO]HEADER
```

This option indicates whether for each block of the Protection Log a header is displayed (HEADER) or not (NOHEADER).

The default is HEADER.

INTERNAL_ID

```
INTERNAL_ID = number
```

This option displays only the records with the specified internal ID.

ISN

```
ISN = (number [,number] ... )
```

The Protection Log records are only displayed if they belong to the ISNs specified by this parameter. Only records of the types DA and DV are displayed. Please refer to the tables at the end of this section for a description of the various types of Protection Log records. This parameter can only be used in conjunction with the FILE parameter.

MODIFIED_RABN

```
MODIFIED_RABN = number
```

This option displays only the records in which modifications for the specified RABN are logged.

Please refer to the tables at the end of this section for a description of the various types of Protection Log records.

NOFILETYPE

```
NOFILETYPE
```

This keyword specifies that record types that are independent of file numbers (for example ET and BT records) will be displayed in addition to the record types that are bound to file numbers.

Example

```
adaplp dbid=6 file=25 type=(da,dv,et,bt) nofiletype
```

The DA and DVT records of file 25 together with all ET and BT records will be displayed.

NONULL

```
NONULL
```

The NONULL parameter is only relevant if the DECOMPRESSED parameter is also specified. Please refer to the **DECOMPRESSED** parameter for further information.

PLOG

```
PLOG = (number[,number])
```

This parameter is optional. The PLOG number and the extension count can be specified. If an extension count is specified, then only the specified extent will be processed. If no extension count is specified, Adabas will open subsequent extents when necessary. The parameter PLOG must be specified before DATASET=PLOG is specified.

Example:

```
Section layout

          .
          .
          .
250000 260000  10001  30  PLG.36 created

377000 378000   1001  30  PLG.36(3) created

adaplp: plog=36
adaplp: dataset=plog
```

PLG.36 will be opened

```
adaplp: plog=(36,3)
adaplp: dataset=plog
```

PLG.36(3) will be opened.

RABN

```
RABN ={*| number [- number] }
```

This parameter selects one block or a range of consecutive blocks on the WORK or Protection Log file. The information contained in the specified blocks is displayed.

If you specify "*", all blocks are displayed.

Notes:

1. If you start ADAPLP without specifying RABN, the utility will run, but will not produce any output.
2. After specifying the RABN parameter, the requested output is generated immediately. Therefore, you must specify all other parameters required for the output generation, for example the DATASET parameter, before the RABN parameter.

Example

```
adaplp: rabn = 123

adaplp: rabn = 123 - 1246
```

RECORD

```
RECORD ={*| (number [- number]  [, number  [- number]]...) }
```

This parameter selects the records or ranges of records to be printed. All of the records are printed if '*' or nothing is specified.

Example:

```
adaplp: record = (2-5,9,11)
```

The records 2, 3, 4, 5, 9 and 11 are written while printing one or more PLOG blocks.

SEQUENCE

```
SEQUENCE = number
```

This option displays only the records written by the specified sequence number.

[NO]SHORT

```
[NO]SHORT
```

This option indicates whether only Protection Log block headers are printed out (SHORT), or whether all the records in each block are included in the display (NOSHORT).

SHORT implicitly resets DUMP.

By default, the Protection Log block header is displayed followed by all of the records contained in the block.

The default is NOSHORT.

THREAD

```
THREAD = number
```

This option displays only the records with the specified thread.

TSN

```
TSN = number
```

This option displays only the records with the specified transaction sequence number (TSN).

Please refer to the tables at the end of this section for a description of the various types of Protection Log records.

TYPE

```
TYPE = (keyword [,keyword]...)
```

This option displays only the protection log records specified by the given keyword(s). Each keyword corresponds to one or more protection log record types, as shown in the following table.

Keyword	Protection Log record type
AB	AB
ASSO	the record type AC and all record types that are selected by the keywords EXT, FCB and INDEX
AT	AT
BF	BS, BE, BF
BT	BT
C1	C1
C5	C5
CE	CE
CF	CF
CT	CT
DA	DA
DATA	all record types that are selected by the keywords BT, CE, DA, DV, ET and OP.
DC	DC
DT	DT
ET	ET, CL
EXT	ACEXT, UIEXT, NIEXT, DSEXT
FCB	FCBDS, FCBIX, SPISN
INDEX	FE, INDEX, IB, INSRU, REMRU
OP	OP

Please refer to the tables at the end of this section for a description of the various types of Protection Log records.

The default is to display all protection log record types.

USER_ID

```
USER_ID = string
```

This option displays only the records which start with the specified user ID.

Only records of the type BT, C1, C5, CL, DA, DV, ET, FCBDS, FCBIX and INDEX are displayed.

Please refer to the tables at the end of this section for a description of the various types of Protection Log records.

ADAPLP Output

Each block of the Protection Log or WORK is preceded by a header, which consists of the following:

- the block sequence number;
- the size of the block;
- the number of the session that the block belongs to (identical to the PLOG number);
- the time stamp showing when the block was created (internal time stamp for WORK).

The output for a record consists of the following entries:

- a record sequence number (starting at 1 for each block);
- the internal length of the record;
- the command sequence number (uniquely identifies a command);
- the type of PLOG record (see the following table for more information);
- the number of the thread that executed the command.

In addition, most records also have the following entries:

"the internal user identification (in hexadecimal notation) that is uniquely assigned for each command that opens a transaction."

The table below shows the types of PLOG records:

Type	Description
AB	logs WORK wrap around (WORK only).
AC	logs the relocation of a record during backout transaction (WORK only).
ACEXT	logs the extension of the address converter (WORK only).
AT	logs the adding of a field (ADADBM).
BE	logs the end of a buffer flush (WORK only).
BF	logs the start and end of a buffer flush (WORK only).
BS	logs the start of a buffer flush (WORK only).
BT	logs the start of BT processing.
C1	log record from a C1 command. Contains the checkpoint name (PLOG only).
C5	log record from a C5 command (PLOG only).
CE	indicates the last entry of a command (last entry with this sequence number). If the command was a delete operation, the file number and the ISN of the deleted record is displayed. **Example**

Type	Description
	`>>> DELETE FILE 10 ISN 2 <<<`
CF	logs the creation of an FDT (ADAFDU).
CL	logs the CLOSE of a user.
CT	logs the creation of a file (ADAFDU).
DA	logs a data record change. The file, RABN, and ISN of the data record are displayed. The record is either an after image (AI), a before image (BI), or a delta image (DI) and is displayed when DUMP is enabled. 'TSN' is an internal transaction sequence number. All entries that originate from one transaction have the same TSN (see also the description of the ET command in the Command Reference Manual). The output of 'WB' is only displayed if DATASET=WORK has been specified. It shows the WORK block where the previous PLOG record of the same TSN can be found. A 'clu' value that is not zero indicates an exclusive or privileged user.
DC	logs the dropping of a field (ADADBM).
DSEXT	logs the extension of data storage (WORK only).
DT	logs the deletion of a file (ADADBM).
DV	logs a descriptor update (should always be preceded by a DA record). The entries for the file, ISN, TSN, clu, and WB are the same as for the DA record type.
ET	log entry from an ET command. The ET TSN gives the TSN of the last user data written by an ET command.
FCBDS	logs an FCB change for data storage (WORK only).
FCBIX	logs an FCB change for the normal index (WORK only).
FE	logs a change of an index block's first entry (WORK only).
IB	logs an index block that is modified (WORK only).
INDEX	logs an index block that is split (WORK only).
INSRU	logs the insertion of an index block into a reusage chain (WORK only).
NIEXT	logs the extension of the normal index (WORK only).
OP	logs the OPEN of a user.
REMRU	logs the deletion of an index block from a reusage chain (WORK only).
SPISN	logs changes in ISN reusage or space reusage.
UIEXT	logs the extension of the upper index (WORK only).

There are also several flags that may be displayed with DA or DV records:

Flag	Description
AI	the data of this PLOG record contain an after image of the data record (record type DA).
BACKOUT	indicates that the record was written during a backout within a single command.
BI	the data of this PLOG record contain a before image of the data record (record type DA).
BT	indicates that the record was written during the backout of a transaction.
DI	the data of this PLOG record contain a delta image of the data record (record type DA).

Flag	Description
FDATA	indicates that this is the first DA record of this command.
FIRST_ENTRY	indicates that this is the first record with a given sequence number.
HIMERGE	merge of the highest index level.
HISPLIT	split of the highest index level.
USERD	transaction carries user data.

19 ADAPRI (Print Adabas Blocks)

- Functional Overview 296
- Procedure Flow 297
- Checkpoints 298
- Control Parameters 298

This chapter describes the utility "ADAPRI".

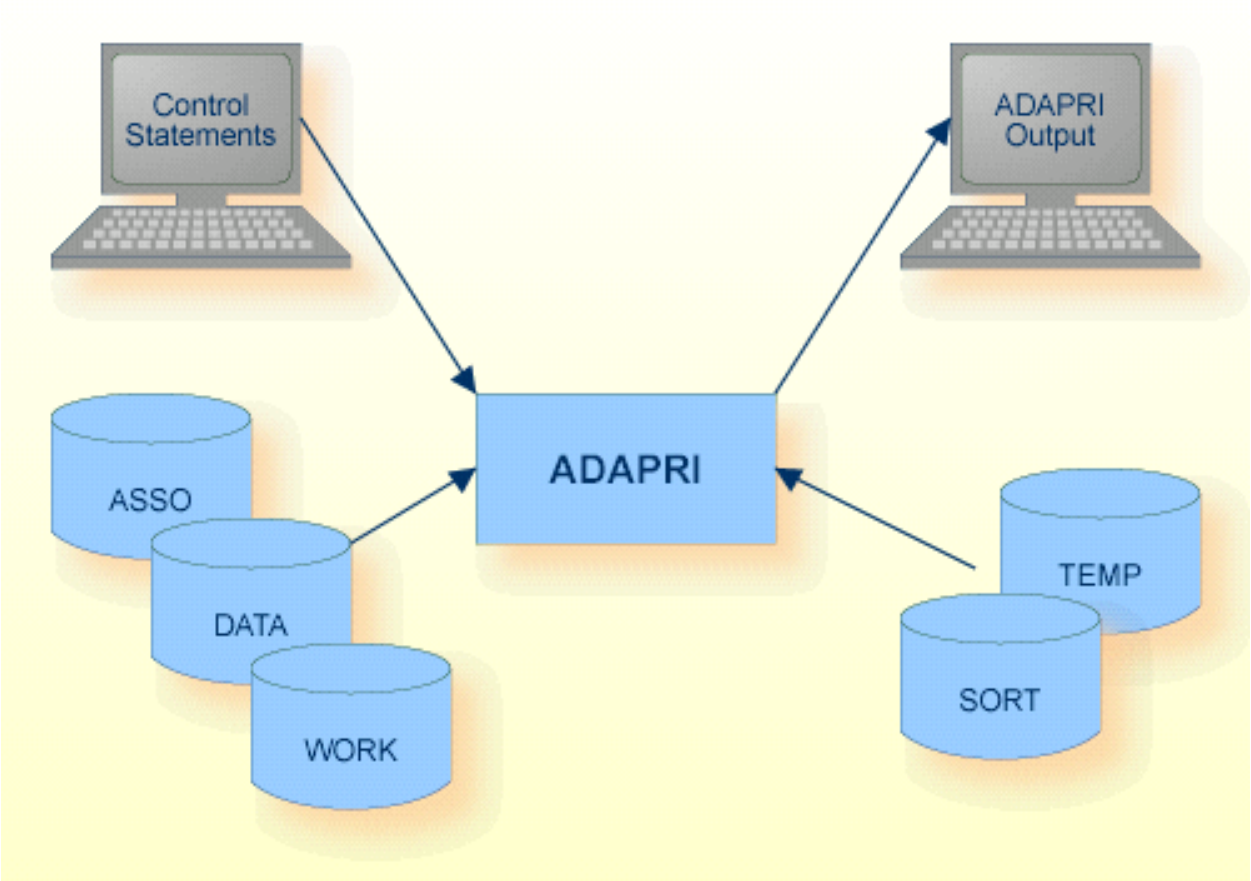
Functional Overview

The ADAPRI utility prints the contents of a block (or range of blocks) in the Associator, Data Storage, WORK, TEMP, or SORT for maintenance or auditing purposes.

The output is in hexadecimal and ASCII format. Subsequent identical lines and blocks are suppressed.

This utility is a multi-function utility. For more information about single- and multi-function utilities, see *Adabas Basics*, *Using Utilities* in the Adabas documentation.

Procedure Flow



Data Set	Environment Variable / Logical Name	Storage Medium	Additional Information
Associator	ASSOx	Disk	
Data storage	DATAx	Disk	
Sort storage	SORTx	Disk	
Control statements	stdin		Utilities Manual
ADAPRI output	stdout		
Temporary storage	TEMPx	Disk	
Work storage	WORK1	Disk	

Assignments to the ASSO container files are required in order to be able to process the DATA or WORK container files.

Checkpoints

The utility writes no checkpoints.

Control Parameters

The following control parameters are available:

```
DATASET = keyword

DBID = number

RABN = number [- number]
```

DATASET

```
DATASET = keyword
```

This parameter specifies the part of the database to be dumped. Valid keywords are:

Keyword	Meaning
ASSO	Associator
DATA	Data Storage
SORT	Sort Area
TEMP	Temporary Area
WORK	Work Area

Example

```
adapri: dataset = asso, rabn = 123 - 321
```

The Associator is dumped from RABN 123 to RABN 321

DBID

```
DBID = number
```

This parameter selects the database to be used.

This parameter is not required if DATASET = TEMP or SORT.

RABN

```
RABN = number [- number]
```

This parameter specifies one RABN or a range of RABNs to be dumped.

Examples

```
adapri: dbid = 1, dataset = data, rabn = 123
```

DATA RABN 123 of database 1 is to be dumped.

```
adapri: dataset = sort, rabn = 123 - 129
```

The RABNs from 123 to 129 on the data set SORT are to be dumped.

20 ADAREC (Recovery Of Database Or Files)

- Functional Overview 302
- Procedure Flow 304
- Checkpoints 305
- ADAREC Input Data 305
- Control Parameters 305
- REGENERATE Examples 313
- BACKOUT Examples 319
- ADAREC Restart Considerations 322

This chapter describes the utility "ADAREC".

Functional Overview

The ADAREC utility consists of the following database functions:

- The LIST function lists information about a Protection Log.
- The REGENERATE function re-applies updates.
- The BACKOUT function reverts the updates.

Checkpoints written in the Protection Log can be specified as start and end points of the processing. The checkpoints used are normally the result of a checkpoint command (C1) but may also be internal checkpoints taken by OP commands from EXU users or utility actions. If the whole database is processed, certain files may be excluded by using the EXCLUDE_FILES option. The files specified with this option are not updated, and the updates that are excluded are reported.

If REGENERATE terminates at a SYNP checkpoint, ADAREC "looks ahead" on the current PLOG to find an alternative restart point for the next run of this PLOG.

BACKOUT processes the Protection Log backwards. It stops at SYNP and SYNC checkpoints for utility functions that are not automatically undone. When BACKOUT stops at a checkpoint, it looks for other checkpoints immediately before that checkpoint to find a restart point.

The utility then displays a list of other utility functions that have been executed when the Protection Log was written. These functions must be handled before ADAREC can be restarted. If one or more SYNP checkpoints were found, ADAREC terminates.

- with exit code 14, if the PLOG contains further transactions to be applied via a restart of ADAREC,
- otherwise with exit code 12.

The calculated restart point can be reset or overridden by entering BLOCK = or CHECKPOINT =. Refer to the database report utility ADAREP in this manual for a description of the possible system checkpoint types.

Normally, REGENERATE completes all fully-logged and confirmed transactions. This function is most frequently used when the database (or one or more files) has been restored to a previous status with the RESTORE function of the ADABCK utility.

If the utility writes records to the error file, it will exit with a non-zero status.

Notes:

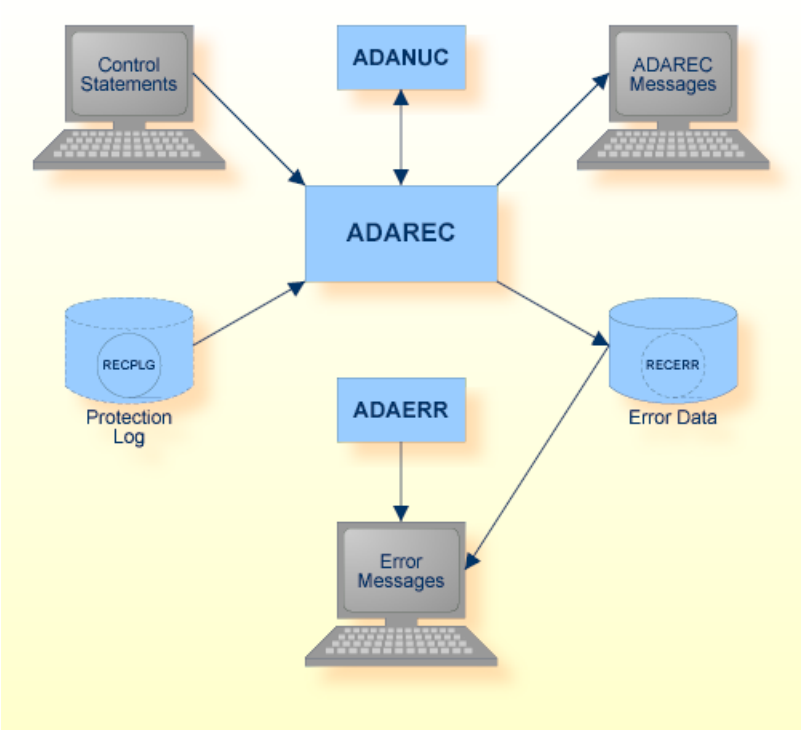
1. If ADAREC is used more than once at the same time to regenerate files, you should first increase the value of the nucleus parameter LBP - this is because ADAREC performs a large number of

database updates, and failure to provide a large enough value of LBP may lead to an Adabas response code 162 being returned.

2. Exit code 12 was introduced with Version 6.3 SP2 - previous releases of Adabas always terminated with exit code 14 when a SYNP checkpoint was found.

This utility is a single-function utility. For more information about single- and multi-function utilities, see *Adabas Basics*, *Using Utilities* in the Adabas documentation.

Procedure Flow



REGENERATE and BACKOUT Functions

Data Set	Environment Variable / Logical Name	Storage Medium	Additional Information
Control statements	stdin		Utilities Manual
ADAREC messages	stdout		Messages and Codes
Rejected data	RECERR	Disk (* see note)	Output of ADAREC
Protection log	RECPLG	Disk	

Note: (*) A named pipe can be used for this sequential file.

The sequential file RECPLG can have multiple extents. For detailed information about sequential files with multiple extents, see *Adabas Basics, Using Utilities* in the Adabas documentation

Checkpoints

The following table shows the nucleus requirements for each function and the checkpoints written:

Function	Nucleus must be active	Nucleus must NOT be active	Nucleus is NOT required	Checkpoint written
LIST			X	-
REGENERATE	X			SYNX
BACKOUT	X			SYNX

ADAREC Input Data

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 regenerate the updates.

Control Parameters

The following control parameters are available:

```
M    DBID = number

     LIST = keyword

     REGENERATE = {* [,EXCLUDE_FILES =
                  (number[-number] [,number[-number] ] ... ) } |
                  (number[-number] [,number[-number] ] ... )
                  PLOG = number
D                 [,[NO]BI_CHECK]
                  [,BLOCK = ([number][,number])
                       ,CHECKPOINT = ([string][,string])]
D                 [,[NO]ERROR_LOG]
D                 [,ON_ERROR = keyword]

     BACKOUT = {* [,EXCLUDE_FILES =
                  (number[-number] [,number[-number] ] ... ) } |
                  (number[-number] [,number[-number] ] ... )
                  PLOG = number
D                 [,[NO]BI_CHECK]
                  [,BLOCK = ([number][,number])
                       ,CHECKPOINT = ([string][,string])]
```

```
D                  [,[NO]ERROR_LOG]
D                  [,ON_ERROR = keyword]
```

DBID

```
DBID = number
```

This parameter selects the database to be used.

Note: Program functions which do not require the nucleus to be running need the environment variables/logical names set for the container files.

LIST

```
[PLOG=number,] LIST = keyword
```

Valid keywords are BRIEF, FULL and RESTART. BRIEF lists the Protection Log number and its creation date. FULL lists additional information about the records on the Protection Log, e.g. the checkpoints, the number of modifications for each file, etc. RESTART displays the restart points that ADAREC writes when it encounters checkpoints while processing.

Note: The timestamps displayed for checkpoints are the timestamps that were made when the checkpoints were included in the PLOG. When offline checkpoints are created, the checkpoints are first written to the checkpoint block in the ASSO. The next time the nucleus is started, they are written to the checkpoint file with the actual checkpoint creation date, and to the PLOG with the current date. This implies that the timestamps for offline checkpoints displayed with ADAREP CHECKPOINT and by ADAREC LIST are different.

The LIST=FULL function also checks the structure of the Protection Log to ensure that it is internally consistent. If a structural error is detected, a message is output indicating the error type as well as the record and block numbers.

If the Protection Log is within a disk section, the PLOG parameter must be set before LIST can be specified.

Examples

```
adarec: list=brief

Protection log 1 - 26-OCT-2006 11:39:03
```

The creation date of PLOG 1 is displayed.

REGENERATE

This function is used to regenerate a whole database or files within a database.

Database Regeneration

```
REGENERATE = *, PLOG = number
             [,EXCLUDE_FILES = (number[-number][,number[-number]]...)]
             [,[NO]BI_CHECK]
             [,BLOCK = ([number][,number]),
                   CHECKPOINT = ([string][,string])]
             [,[NO]ERROR_LOG]
             [,ON_ERROR = keyword]
```

This option of the REGENERATE function regenerates a database. A file exclusion list can be used to exclude certain files from the regenerate. ET logic is supported.

During REGENERATE processing, ADAREC sets the database to utility-only mode. Processing terminates if a SYNP checkpoint is encountered. In this case, ADAREC inspects the Protection Log in order to calculate an alternative restart point. This restart point is then displayed together with a list of utility functions that must be executed before processing can be continued. The next call to REGENERATE automatically sets up at this point. The use of the calculated restart point can be overridden by specifying "BLOCK=" or "CHECKPOINT=" (that is, supplying empty values for these keywords). This procedure is repeated until the end of the PLOG is reached. After ADAREC has terminated, the database remains in utility-only mode, because more calls to REGENERATE may follow. After the database regeneration has finished, you can enable the database for normal processing with the ADAOPR command OPTIONS=NOUTILITIES_ONLY.

[NO]BI_CHECK

If this option is set to BI_CHECK, ADAREC checks the consistency of the before images in the Protection Log against the data in the database (is the ISN in use; does the record exist; is there a before image mismatch?). If a mismatch is encountered, ADAREC issues messages containing the relevant information and does not perform the update.

If this option is set to NOBI_CHECK, the consistency check is still made and the ERROR_LOG is implicitly enabled; however, on finding a BI inconsistency, the update is made and the mismatch is reported to the ERROR_LOG (see below). If errors are encountered, only the first error for each file will be displayed, all subsequent errors are logged to the ERROR_LOG. Note that the index might become inconsistent in this case.

However, if the PLOG was written with the NOBI option of the nucleus, it will not contain any before images and the BI_CHECK option cannot be set.

The default is BI_CHECK.

BLOCK = ([number][,number])

This parameter specifies the numbers of the blocks in the Protection Log files that contain the corresponding checkpoint names. The block numbers can be taken from ADAREC LIST=FULL.

CHECKPOINT = ([string][,string])

This parameter specifies the starting and ending checkpoint names. The checkpoint names can be taken from the ADAREP database status report or ADAREC LIST=FULL.

If processing is to start at the beginning of the Protection Log file, the first parameter must be omitted.

[NO]ERROR_LOG

Setting this option to ERROR_LOG enables the automatic logging of any BI inconsistencies that may be detected when using the NOBI_CHECK option. The contents of the error file produced can be examined using the ADAERR utility. Do not print this error file using the standard operating system print utilities, since the records contain nonprintable characters. See *ADAERR* for further information.

The default is NOERROR_LOG.

EXCLUDE_FILES = (number[-number][,number[-number]]...)

This parameter specifies the files to be excluded when regenerating a complete database. The updates that are excluded are written to a report.

ON_ERROR = keyword

Valid keywords are ABORT and EXCLUDE. The keyword used determines what action to take if ADAREC detects non-fatal errors during processing (e.g. response code 17, file not loaded). ABORT abnormally terminates regenerate processing, and EXCLUDE excludes the file in question from the regenerate if Data Storage errors occur (nucleus response codes 17, 49, 75, 77 and 113).

If, however, an error occurs while updating a file's index (nucleus response codes 75, 76, 77, 98, 165, 166, 167 and 176), only the regeneration of the Data Storage for this file will continue. When the regeneration process is complete, the index of this file is marked as invalid. The ADAINV REINVERT function with the ALL_FIELDS option then has to be run for this file (please refer to the ADAINV utility in this manual for more detailed information). If index errors occur and if the regenerate includes several Protection Logs, all of the Protection Logs should be processed before reinverting the index. Reinverting the index each time a Protection Log results in index errors would waste considerable amounts of time and computer resources.

The default is ON_ERROR=EXCLUDE.

PLOG = number

This parameter specifies the log number of the Adabas Protection Log to be used as input for the REGENERATE function. This number can be found with ADAREC using the LIST = BRIEF function.

File Regeneration

```
REGENERATE = (number[-number][,number[-number]]...), PLOG = number
             [,[NO]BI_CHECK]
             [,BLOCK = ([number][,number]),
                   CHECKPOINT = ([string][,string])]
             [,[NO]ERROR_LOG]
             [,ON_ERROR = keyword]
```

This option of the REGENERATE function re-applies all updates in a Protection Log for the specified files or ranges of files. LOB files specified are ignored, but the LOB files assigned to all base files specified are dumped too.

During regenerate processing, ADAREC locks the files for exclusive use. The regenerate terminates if a SYNP checkpoint is found while processing a protection log. In this case, ADAREC inspects the Protection Log in order to calculate an alternative restart point. This restart point is then displayed with a list of utility functions that must be executed before processing can be continued. The next call to REGENERATE automatically sets up at this point. The use of the calculated restart point can be overridden by specifying "BLOCK=" or "CHECKPOINT=" (that is, supplying empty values for these keywords). This procedure is repeated until the end of the Protection Log is reached.

The files remain locked, because more calls to REGENERATE may follow. After the files regeneration is finished, you must unlock the files with the ADAOPR command UNLOCK.

The following functions are not allowed while ADAREC is active:

- ADAOPR ET_SYNC FEOF = PLOG
- ADABCK DUMP
- ADAOPR STOP to a sub-user while the associated ADAREC user exists

[NO]BI_CHECK

If this option is set to BI_CHECK, ADAREC checks the consistency of the before images in the Protection Log against the data in the database (is the ISN in use; does the record exist; is there a before image mismatch?). If a mismatch is encountered, ADAREC issues messages containing the relevant information and does not perform the update.

If this option is set to NOBI_CHECK, the consistency check is still made and the ERROR_LOG is implicitly enabled; however, on finding a BI inconsistency, the update is made and the mismatch is reported to the ERROR_LOG (see below). If errors are encountered, only the first error for each

file will be displayed, all subsequent errors are logged to the ERROR_LOG. Note that the index might become inconsistent in this case.

NOBI_CHECK improves performance at the expense of possible loss of data consistency. We advise you therefore not to use NOBI_CHECK for mission critical databases.

The default is BI_CHECK.

BLOCK = ([number] [,number])

This parameter specifies the blocks in the Protection Log files that contain the corresponding checkpoint names. The block numbers can be taken from ADAREC LIST=FULL.

CHECKPOINT = ([string] [,string])

This parameter specifies the starting and ending checkpoint names. The checkpoint names can be taken from the ADAREP database status report.

If processing is to start at the beginning of the Protection Log file, the first parameter must be omitted. However, if the first checkpoint name is supplied, it must be found in the first Protection Log file.

If processing is to stop at the end of the last Protection Log file, the second checkpoint name must be omitted.

[NO]ERROR_LOG

Setting this option to ERROR_LOG enables the automatic logging of any BI inconsistencies that may be detected when using the NOBI_CHECK option. The contents of the error file produced can be examined using the ADAERR utility. . Please refer to the ADAERR utility in this manual for more detailed information.

The default is NOERROR_LOG.

ON_ERROR = keyword

Valid keywords are ABORT and EXCLUDE. The keyword used determines what action to take if ADAREC detects non-fatal errors during processing (e.g. response code 17, file not loaded). ABORT abnormally terminates regenerate processing, and EXCLUDE excludes the file in question from the regenerate if Data Storage errors occur (nucleus response codes 17, 49, 75, 77 and 113).

If, however, an error occurs while updating a file's index (nucleus response codes 75, 76, 77, 98, 165, 166, 167 and 176), only the regeneration of the Data Storage for this file will continue. When the regeneration process is complete, the index of this file is marked as invalid. The ADAINV REINVERT function with the ALL_FIELDS option then has to be run for this file (please refer to the ADAINV utility in this manual for more detailed information). If index errors occur and if the regenerate includes several Protection Logs, all of the Protection Logs should be processed before

reinverting the index. Reinverting the index each time a Protection Log results in index errors would waste considerable amounts of time and computer resources.

The default is ON_ERROR=EXCLUDE.

PLOG = number

This parameter specifies the log number of the Adabas Protection Log to be used as input for the REGENERATE function. This number can be found with ADAREC using the LIST = BRIEF function.

BACKOUT

This function is used to back out changes in a whole database or files within a database.

Database Backout

```
BACKOUT = *, PLOG = number
             [,EXCLUDE_FILES = (number[-number][,number[-number]]...)]
             [,[NO]BI_CHECK]
             [,BLOCK = ([number][,number]),
                   CHECKPOINT = ([string][,string])]
             [,[NO]ERROR_LOG]
             [,ON_ERROR = keyword]
```

This option of the BACKOUT function backs out changes in a database. A file exclusion list can be used to exclude certain files from the backout. ET logic is supported. The Protection Log is processed backwards.

During BACKOUT processing, ADAREC sets the database to utility-only mode. Processing terminates if a SYNP or certain SYNX checkpoints is encountered.

ADAREC stops if the SYNX checkpoint was written for a delete, refresh or create file.

When ADAREC stops at a checkpoint, the Protection Log is inspected in order to calculate an alternative restart point. This restart point is then displayed together with a list of utility functions that should be undone before processing is continued. The next call to BACKOUT/REGENERATE automatically sets up at this point. The use of the calculated restart point can be overridden by specifying "BLOCK=" or "CHECKPOINT=" (that is, supplying empty values for these keywords). This procedure is repeated until the begin of the PLOG is reached. After ADAREC has terminated, the database remains in utility-only mode, because more calls to BACKOUT may follow. After the database regeneration has finished, you can enable the database for normal processing with the ADAOPR command OPTIONS=NOUTILITIES_ONLY.

Note: The parameters described for REGENERATE also apply analogously to BACKUP. See *REGENERATE > Database Regeneration* for detailed information on the parameters that can be used for database backout.

File Backout

```
BACKOUT = (number[-number][,number[-number]]...), PLOG = number
             [,[NO]BI_CHECK]
             [,BLOCK = ([number][,number]),
                   CHECKPOINT = ([string][,string])]
             [,[NO]ERROR_LOG]
             [,ON_ERROR = keyword]
```

This option of the BACKOUT function undoes all updates in a Protection Log for the specified files or ranges of files. LOB files specified are ignored, but the changes to LOB files assigned to files specified are backed out too.

During backout processing, ADAREC locks the files for exclusive use. The backout terminates if a SYNP or a certain SYNX checkpoint is found while processing a Protection Log.

ADAREC stops if the SYNX checkpoint was written for a delete, refresh or create file.

When ADAREC stops at a checkpoint, the Protection Log is inspected in order to calculate an alternative restart point. This restart point is then displayed with a list of utility functions that should be undone before processing is continued. The next call to BACKOUT or REGENERATE automatically sets up at this point. The use of the calculated restart point can be overridden by specifying "BLOCK=" or "CHECKPOINT=" (that is, supplying empty values for these keywords). This procedure is repeated until the begin of the Protection Log is reached.

The files remain locked because more calls to BACKOUT may follow. After the files regeneration is finished, you must unlock the files with the ADAOPR command UNLOCK.

The following functions are not allowed while ADAREC is active:

- ADAOPR ET_SYNC FEOF = PLOG
- ADABCK DUMP
- ADAOPR STOP to a sub-user while the associated ADAREC user exists

Note: The parameters described for REGENERATE also apply analogously to BACKUP. See *REGENERATE > File Regeneration* for detailed information on the parameters that can be used for file backout.

REGENERATE Examples

Example 1

In this example, database 2 is to be regenerated using the Protection Log 2. File 12 is to be excluded from the regenerate.

```
adarec: regenerate=*,plog=2
adarec: exclude_files=12
adarec:

Protection log 2 - 26-OCT-2006 11:48:59


Block     3 - checkpoint SYNC - 11:49:00 - USERID ADANUC <version>
%ADAREC-I-CHKIGN, Checkpoint ignored

The following utility functions were executed in the original session:

Block     4 - checkpoint SYNP - 11:50:02 - USERID ADADBM REFRESH=13

Block     5 - checkpoint SYNX - 11:50:03 - USERID ADADBM RESET=UCB,IDENT=7

Block     6 - checkpoint SYNP - 11:50:03 - USERID ADADBM RECOVER

Re-execute all SYNP utility functions starting from block 4.

REGENERATE summary


Calculated RESTART point - BLOCK=6,CHECKPOINT=SYNP
```

Processing of the Protection Log terminated at the SYNP checkpoint in block 4. However, no updates were found on looking ahead and processing can be continued from the calculated restart point in block 6. ADAREC displays a list of the utility functions that must be executed before processing continues. The next call to REGENERATE=* will automatically continue at this calculated restart point.

```
adarec: regenerate=*,plog=2
%adarec-I-restartp, calculated restart point - block=6,checkpoint=synp
adarec: exclude_files=12
adarec:

Protection log 2 - 26-OCT-2006 11:48:59.86
```

```
Block     6 - checkpoint SYNP - 11:50:03.86 - USERID ADADBM RECOVER
%ADAREC-I-CHKSTP, starting checkpoint

     1 modifications in file  11
     1 modifications EXCLUDED from file  12

     4 ET commands issued

Block     7 - checkpoint SYNC - 11:52:38.98 - USERID ADANUC SHUTDOWN
%ADAREC-I-CHKIGN, Checkpoint ignored


REGENERATE summary


Protection log 2 processed
```

Processing of the Protection Log continues at the calculated restart point. The regenerate terminates successfully.

Example 2

In this example, database 2 is to be regenerated using the Protection Log 2. Processing is to start at the checkpoint SYNP in block 6 of the Protection Log. If Data Storage errors occur, the file in question will be excluded from the regenerate. If index errors occur, the file's index will be excluded from the regenerate and marked as invalid.

```
adarec: regenerate=*,plog=2,block=6,checkpoint=synp
adarec: on_error=exclude
adarec:

Protection log 2 - 26-OCT-2006 11:48:59.86

%ADAREC-W-UTIENA, OPTIONS=UTILITIES enabled in nucleus by ADAREC
%ADAREC-W-RECUPD, Updates performed between Nucleus and REGENERATE'S startup
%ADAREC-W-RECCMD, 1 N1  command(s)

Block     6 - checkpoint SYNP - 11:50:03.86 - USERID ADADBM RECOVER
%ADAREC-I-CHKSTP, starting checkpoint

     1 modifications in file  11

     3 ET commands issued

%ADAREC-E-ISNINUSE, ISN 774 in use in file  12
%ADAREC-I-PLOGRB, from record 14 in block 7 in PLOG 2
%ADAREC-I-UPDEXC, ALL following updates in file  12 will be EXCLUDED

     1 modifications EXCLUDED from file  12
```

```
     1 ET command issued

Block     7 - checkpoint SYNC - 11:52:38.98 - USERID ADANUC SHUTDOWN
%ADAREC-I-CHKIGN, Checkpoint ignored


REGENERATE summary


Protection log 2 processed
```

An ISN conflict occurred in file 12 and all subsequent updates to this file were excluded. The cause of the error has to be investigated. However, the nucleus was started without 'OPTIONS=UTILITIES_ONLY' and an N1 command was issued before the regenerate was started.

The Protection Log was processed to its end, the abort system message is used only to indicate a fatal error.

Example 3

This example is similar to the previous one, with the exception that processing will abort if Data Storage or index errors are encountered.

```
adarec: regenerate=*,plog=2,block=6,checkpoint=synp
adarec: on_error=abort
adarec:

Protection log 2 - 26-OCT-2006 11:48:59.86

%ADAREC-W-UTIENA, OPTIONS=UTILITIES enabled in nucleus by ADAREC
%ADAREC-W-RECUPD, Updates performed between Nucleus and REGENERATE'S startup
%ADAREC-W-RECCMD, 1 N1  command(s)

Block     6 - checkpoint SYNP - 11:50:03.86 - USERID ADADBM RECOVER
%ADAREC-I-CHKSTP, starting checkpoint

     1 modifications in file  11

     3 ET commands issued

%ADAREC-E-ISNINUSE, ISN 774 in use in file  12
%ADAREC-I-PLOGRB, from record 14 in block 7 in PLOG 2
```

An ISN conflict occurred in file 12 and further processing was aborted.

Example 4

In this example, database 2 is to be regenerated using the Protection Log 3. The before images in the Protection Log will be checked against the data in the database and mismatches will be displayed on the terminal.

```
adarec: regenerate=*,plog=3
adarec:

Protection log 3 - 26-OCT-2006 12:10:25.12

%ADAREC-W-UTIENA, OPTIONS=UTILITIES enabled in nucleus by ADAREC

Block     1 - checkpoint SYNC - 12:10:25.12 - USERID ADANUC 3.2/0 PL 0
%ADAREC-I-CHKIGN, Checkpoint ignored

     1 ET command issued

%ADAREC-E-RECMIS, Before image mismatch for ISN 3 in file  11
%ADAREC-I-PLOGRB, from record 7 in block 2 in PLOG 3
%ADAREC-I-UPDEXC, ALL following updates in file  11 will be EXCLUDED

     1 modifications EXCLUDED from file  11
     1 modifications in file  12

     3 ET commands issued

Block     2 - checkpoint SYNC - 12:11:44.30 - USERID ADANUC SHUTDOWN
%ADAREC-I-CHKIGN, Checkpoint ignored


REGENERATE summary


Protection log 3 processed
```

One before image mismatch occurred during processing. As a result, one update was excluded from file 11.

Having restored the files, the same example can be rerun with no consistency check of the before images and with BI error logging enabled.

The Protection Log was processed to its end, the abort system message is used only to indicate a fatal error.

```
adarec: regenerate=*,plog=3,nobi_check
adarec:

Protection log 3 - 26-OCT-2006 12:10:25.12

%ADAREC-W-UTIENA, OPTIONS=UTILITIES enabled in nucleus by ADAREC

Block     1 - checkpoint SYNC - 12:10:25.12 - USERID ADANUC 3.2/0 PL 0
%ADAREC-I-CHKIGN, Checkpoint ignored

%ADAREC-W-RECMIS, Before image mismatch for ISN 3 in file  11
%ADAREC-I-PLOGRB, from record 7 in block 2 in PLOG 3

     1 modifications in file  11
     1 modifications in file  12

     4 ET commands issued

     1 BI_CHECK error in file  11

Block     2 - checkpoint SYNC - 12:11:44.30 - USERID ADANUC SHUTDOWN
%ADAREC-I-CHKIGN, Checkpoint ignored


REGENERATE summary

     1 BI_CHECK error in file  11


Protection log 3 processed
```

One BI_CHECK error occurred during processing.

The Protection Log was processed to its end, the abort system message is used only to indicate a fatal error.

The source of the errors is written to an error file which can be displayed using the ADAERR utility. The first error is logged and also written to the error file. All subsequent errors are written to ERROR_LOG.

The following error file was produced:

```
%ADAERR-E-RECMIS, Before image mismatch for ISN 3 in file  11
%ADAERR-I-PLOGRB, from record 7 in block 2 in PLOG 3
```

Example 5

In this example, database 2 is to be regenerated using the Protection Log 3.

```
adarec: regenerate=*,plog=3
adarec:

Protection log 3 - 26-OCT-2006 12:10:25.12

%ADAREC-W-UTIENA, OPTIONS=UTILITIES enabled in nucleus by ADAREC

Block     1 - checkpoint SYNC - 12:10:25.12 - USERID ADANUC 3.2/0 PL 0
%ADAREC-I-CHKIGN, Checkpoint ignored

%ADAREC-E-ERRIUP, Error response 165 during index update
%ADAREC-E-Adabas_165, * Invalid descriptor name in DVT
%ADAREC-I-DESNAM, Descriptor name XA
%ADAREC-I-ISNFILE, from ISN 3 in file  11
%ADAREC-I-PLOGRB, from record 7 in block 2 in PLOG 3
%ADAREC-I-REINVERT, REINVERT all descriptors to re-establish INDEX
%ADAREC-I-REGDAT, Regenerating ONLY data-storage for file  11

     1 modifications in file  11
     1 modifications in file  12

     4 ET commands issued

Block     2 - checkpoint SYNC - 12:11:44.30 - USERID ADANUC SHUTDOWN
%ADAREC-I-CHKIGN, Checkpoint ignored


 REGENERATE summary


Protection log 3 processed
```

An invalid descriptor name was encountered during processing. As a result, only the data storage of file 11 was regenerated. All of the descriptors will have to be reinverted in order to reestablish the index.

The Protection Log was processed to its end, the abort system message is used only to indicate a fatal error.

If index errors occur and if the regenerate includes several Protection Logs, all of the Protection Logs should be processed before reinverting the index. Reinverting the index each time a Protection Log results in index errors would waste considerable amounts of time and computer resources.

Example 6

In this example, database 2 is to be regenerated using the Protection Log 4 after the regenerate processing of Protection Log 3 resulted in an index error.

```
adarec: regenerate=*,plog=4
adarec:

Protection log 4 - 26-OCT-2006 12:12:00.15


Block     1 - checkpoint SYNC - 12:12:00.15 - USERID ADANUC <version>
%ADAREC-I-CHKIGN, Checkpoint ignored

%ADAREC-E-FCBNAC, file  11's index not accessible
%ADAREC-I-REGDAT, Regenerating ONLY data-storage for file  11

     1 modifications in file  11
     1 modifications in file  12

     4 ET commands issued

Block     2 - checkpoint SYNC - 12:12:19.35 - USERID ADANUC SHUTDOWN
%ADAREC-I-CHKIGN, Checkpoint ignored


REGENERATE summary


Protection log 4 processed
```

The index error that occurred while processing Protection Log 3 (see example 5) means that file 11's index is no longer accessible. Only the Data Storage of file 11 is regenerated, whereas both the Data Storage and the index of file 12 are regenerated.

The Protection Log was processed to its end, the abort system message is used only to indicate a fatal error.

BACKOUT Examples

Example 1

Backout until checkpoint W8NW in block 520.

```
adarec dbid=101 backout=*,plog=2,checkpoint=(SYNC,W8NW),block=(927,520)

%ADAREC-I-STARTED,      15-SEP-2025 15:44:29, Version 7.4.0.0 (Linux 64Bit)
%ADAREC-I-DBON, database 101 accessed online


Protection log 2 - 15-SEP-2025 15:38:42


Block   927 - checkpoint SYNC - 15-SEP-2025 15:38:42 - USER-ID ADAOPR FEOF=PLOG
%ADAREC-I-CHKSTP, starting checkpoint

   195 modifications in file 9
   235 modifications in file 11
   247 modifications in file 12
   383 modifications in file 13
    16 modifications in file 14

   223 ET commands issued

Block   524 - checkpoint IKH6 - 15-SEP-2025 15:38:42 - USER-ID

     4 modifications in file 9
     4 modifications in file 11
     6 modifications in file 12
     9 modifications in file 13

     4 ET commands issued

Block   520 - checkpoint W8NW - 15-SEP-2025 15:38:42 - USER-ID
%ADAREC-I-CHKIGN, Checkpoint ignored


%ADAREC-I-RESTARTP, Calculated RESTART point - BLOCK=520,CHECKPOINT=W8NW

%ADAREC-I-IOCNT, 1 IOs on dataset ASSO
%ADAREC-I-IOCNT, 410 IOs on dataset RECPLG
%ADAREC-I-TERMINATED,   15-SEP-2025 15:44:29, elapsed time: 00:00:00
```

Example 2

Continue at restart point until backout stops at refresh checkpoint.

```
adarec dbid=101 backout=*,plog=2

%ADAREC-I-STARTED,      15-SEP-2025 15:45:29, Version 7.4.0.0 (Linux 64Bit)
%ADAREC-I-DBON, database 101 accessed online
%ADAREC-I-RESTARTP, Calculated RESTART point - BLOCK=520,CHECKPOINT=W8NW

Protection log 2 - 15-SEP-2025 15:38:42

   223 ET commands issued
```

```
     4 ET commands issued

Block   520 - checkpoint W8NW - 15-SEP-2025 15:38:42 - USER-ID
%ADAREC-I-CHKSTP, starting checkpoint

    12 modifications in file 9
     6 modifications in file 11
    14 modifications in file 12
    19 modifications in file 13

     8 ET commands issued


     1 ET command issued
....

    14 modifications in file 13

     7 ET commands issued

   The following Utility functions were executed in the original session

Block   403 - checkpoint SYNX - 15-SEP-2025 15:38:42 - USER-ID ADADBM REFRESH=11

%ADAREC-I-RESTARTP, Calculated RESTART point - BLOCK=403,CHECKPOINT=SYNX

%ADAREC-I-IOCNT, 1 IOs on dataset ASSO
%ADAREC-I-IOCNT, 527 IOs on dataset RECPLG
```

Example 3

After refresh, file 11 must be excluded because it is empty.

```
adarec dbid=101 backout=*,plog=2,exclude=11

%ADAREC-I-STARTED,      15-SEP-2025 15:46:22, Version 7.4.0.0 (Linux 64Bit)
%ADAREC-I-DBON, database 101 accessed online
%ADAREC-I-RESTARTP, Calculated RESTART point - BLOCK=403,CHECKPOINT=SYNX

Protection log 2 - 15-SEP-2025 15:38:42

   223 ET commands issued

     4 ET commands issued

....

     2 ET commands issued

     7 ET commands issued
```

```
Block   403 - checkpoint SYNX - 15-SEP-2025 15:38:42 - USER-ID ADADBM REFRESH=11
%ADAREC-I-CHKSTP, starting checkpoint

   163 modifications in file 9
   241 modifications in file 12
   391 modifications in file 13

   209 ET commands issued

Block    87 - checkpoint CGKL - 15-SEP-2025 15:38:42 - USER-ID

%ADAREC-I-PROCESSED, Protection log 2 processed

%ADAREC-I-MODIFIED,      5 modifications in file 9
%ADAREC-I-MODIFIED,      5 modifications in file 12
%ADAREC-I-MODIFIED,      9 modifications in file 13

%ADAREC-I-ETSTATS,      6 ET commands issued
%ADAREC-I-IOCNT, 1 IOs on dataset ASSO
%ADAREC-I-IOCNT, 928 IOs on dataset RECPLG
```

ADAREC Restart Considerations

An interrupted ADAREC run which leaves a UCB entry has to be re-started from the beginning. Because modifications have already been made, a RESTORE database or RESTORE file has to be executed before re-starting ADAREC. However, if there is no UCB entry, the database has not been modified and ADAREC can be re-started.

An abnormally terminated ADAREC (RESTORE/RECOVER) leaves the database in a consistent state, although it is not possible to tell exactly in which state. ADAREC cannot determine which transactions have already been recovered, so it is necessary to repeat the RESTORE operation and restart the ADAREC from the beginning in order to ensure that everything is recovered.

Having performed the first update, ADAREC writes a 'started' checkpoint to the checkpoint file, e.g.

```
   SYNX   22-MAR-2007 16:49:46       192   ADAREC REG STARTED
```

21 ADAREP (Database Report)

- Functional Overview 324
- Procedure Flow 325
- Checkpoints 325
- Control Parameters 326

This chapter describes the utility "ADAREP".

Functional Overview

The ADAREP utility generates the database status report. This contains information about the current physical layout and logical contents of the database. Unless otherwise stated, the functions can be executed when the nucleus is active or inactive.

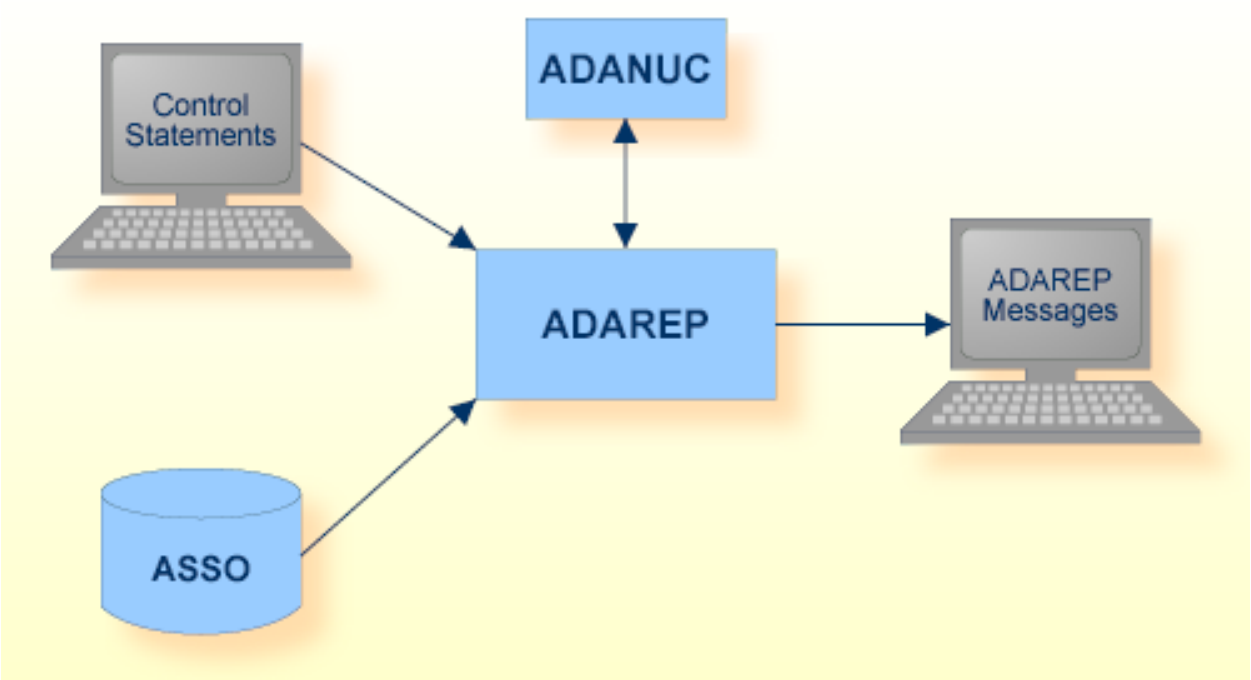
The information contained in this report includes:

- The amount and location of space currently allocated for the Associator and Data Storage;
- The amount and location of unused space available for the Associator and Data Storage;
- Database file summary;
- Checkpoint information;
- Security information;
- Encryption information;
- Information about each file in the database (space allocation, space available, number of records loaded, MAXISN setting, field definitions, etc.);

Only the CHECKPOINTS control parameter (see description below) requires the nucleus to be active.

This utility is a multi-function utility. For more information about single- and multi-function utilities, see *Adabas Basics*, *Using Utilities* in the Adabas documentation.

Procedure Flow



Data Set	Environment Variable / Logical Name	Storage Medium	Additional Information
Associator	ASSOx	Disk	
Control statements	stdin		Utilities Manual
ADAREP report	stdout		Messages and Codes, Utilities Manual
Output data FDT	OUTPUTFDT	Disk	Utilities Manual

Checkpoints

The utility writes no checkpoints.

Control Parameters

The following control parameters are available:

```
 CHECKPOINTS = { * |  ( [absolute-date] [,[absolute-date] ] ) }

        CONNECTION_STRING = string

 CONSTRAINTS

 CONTENTS

 COUNT

M DBID = number

D [NO]FDT

 FILES = { * | (number [-number][,number[-number]]...) }

 [NO]OUTPUTFDT

D [NO]FULL

 FREE_SPACE

D [NO]LARGE

 LAYOUT

 SUMMARY
```

CHECKPOINTS

```
CHECKPOINTS = { * | ( [absolute-date] [,[absolute-date]] ) }
```

This function displays selected information from the checkpoint list and requires the nucleus to be active.

Five types of system checkpoints (SYNP, SYNC, SYNX, OPEN and CLSE) are written to the checkpoint file and to the protection log, together with the user checkpoints written by C1 commands.

SYNC indicates a checkpoint made during nucleus initialization, termination or cancel processing; during the ADAOPR function FEOF = PLOG; due to ADABCK NEW_PLOG processing; or during the function ADAOPR EXT_BACKUP=CONTINUE.

SYNP indicates a checkpoint made by an Adabas utility that requires privileged control, i.e. the module can make updates without using the nucleus. A SYNP checkpoint is, for example, written at the end of an ADAMUP UPDATE run.

SYNX indicates a checkpoint made by a utility that requires exclusive control of one or more files. A SYNX checkpoint is, for example, written by ADAULD.

An OPEN checkpoint is written by the OP command of EXU/EXF users.

A CLSE checkpoint is written by the CL command of EXU/EXF users.

Note: If the ADAREC 'REGENERATE' function is executed using the Protection Log, this utility stops at each SYNP checkpoint since DBA intervention is required.

If an asterisk '*' is entered, all checkpoints are displayed.

The date strings must correspond to the following absolute data and time format:

```
dd-mmm-yyyy[:hh[:mm[:ss]]]
```

Leading zeroes in the date and time specification may be omitted. Any numbers not specified are set to 0, for example 28-jul-2006 is equivalent to 28-jul-2006:00:00:00.

The following table shows the possible values for parameter CHECKPOINTS, and the corresponding checkpoints displayed by this value:

Value specified for parameter CHECKPOINTS	Checkpoints displayed for this specification
* or (,)	All checkpoints
absolute-date	Only the checkpoints written exactly at the date and time specified
(absolute-date,)	Checkpoints written from date and time specified onwards
(,absolute-date)	Checkpoints written up to the date and time specified
(absolute-date,absolute-date)	Checkpoints written from first date and time value specified onwards up to the second date and time value specified

Example

```
adarep: checkpoints=*


Name        Date/Time        Session   User Id / Function
----        ---------        -------   ------------------
SYNP   28-JUL-2006 12:50:34        8   ADADBM DELCP
SYNX   28-JUL-2006 12:50:36        8   ADABCK DUMP=* STARTED
SYNX   28-JUL-2006 12:50:37        8   ADABCK DUMP=*
OPEN   28-JUL-2006 17:23:53        8   otto
```

```
OPEN   28-JUL-2006 17:24:15        8   otto
CLSE   28-JUL-2006 17:24:24        8   otto
```

All checkpoints are displayed.

The column "User ID / Function" contains

- for user checkpoints created via OP/CL commands for EXU/EXF users or via C1 command: the user specified in the Additions 1 field of the relevant OP command;
- for utility checkpoints: the utility function executed.

Taking the output of the example above (checkpoints=*), the selection criteria can be used to filter the checkpoints selected as shown below.

Specifying

```
checkpoints=28-jul-2006:12:50:36
```

will produce the following output:

```
Name        Date/Time        Session   User Id / Function
----        ---------        -------   ------------------
SYNX   28-JUL-2006 12:50:36        8   ADABCK DUMP=* STARTED
```

Specifying

```
checkpoints=(28-jul-2006:12:50:36,)
```

will produce the following output:

```
Name        Date/Time        Session   User Id / Function
----        ---------        -------   ------------------
SYNX   28-JUL-2006 12:50:36        8   ADABCK DUMP=* STARTED
SYNX   28-JUL-2006 12:50:37        8   ADABCK DUMP=*
OPEN   28-JUL-2006 17:23:53        8   otto
OPEN   28-JUL-2006 17:24:15        8   otto
CLSE   28-JUL-2006 17:24:24        8   otto
```

Specifying

```
checkpoints=(,28-jul-2006:12:50:36)
```

will produce the following output:

```
Name        Date/Time        Session   User Id / Function
----        ---------        -------   ------------------
SYNP   28-JUL-2006 12:50:34        8   ADADBM DELCP
SYNX   28-JUL-2006 12:50:36        8   ADABCK DUMP=* STARTED
```

Specifying

```
checkpoints=(28-jul-2006:17, 28-jul-2006:17:24)
```

will produce the following output:

```
Name        Date/Time        Session   User Id / Function
----        ---------        -------   ------------------
OPEN   28-JUL-2006 17:23:53        8   otto
```

CONNECTION_STRING

```
CONNECTION_STRING = string
```

This parameter selects the (remote) database to which all subsequent ADAREP commands apply. ‘String’ is string format to establish the (remote) database connection. This parameter connects the adarep utility to a (remote) database via TCP/IP (ADATCP/ADATCPS). [protocol], [hostname] and [port number] are mandatory fields to establish connection to the remote database via AD-ATCP/ADATCPS. [hostname] address and [port number] are defined with the (remote) database's host name or IP address and port number. (Remote) data-base is accessible if ADATCP feature is enabled for the database.

The CONNECTION_STRING or DBID parameter must be the first ADAREP parameter specified.

The parameters DBID and CONNECTION_STRING are mutually exclusive.

String format:

CONNECTION_STRING=[protocol]://[hostname]:[port number]

It must be succeeding the defined syntax, otherwise it will not establish the connection to remote database.

Example:

CONNECTION_STRING = adatcp://localhost: 50001

CONSTRAINTS

```
CONSTRAINTS
```

This function displays information about all referential constraints in the database that you specify with the DBID parameter.

Example

```
adarep: constraints

     Primary file          Foreign file    Name   Action
     ---------------------------------------------------
           9 (AA)    <---       12 (AC)     HO     DX UX
```

The referential constraint HO links the primary key field AA in the primary file 9 with the foreign key field AC in the foreign file 12. The associated actions are delete no action (DX) and update no action (UX).

CONTENTS

```
CONTENTS
```

This function displays information about the files in the database that you specify with the DBID parameter.

Example

```
adarep: contents



Content of Database 163          30-MAY-2017 11:36:07


                                                   Index   Extents   Pad % flags
 File Filename          loaded on        Top ISN level  N  U  A  D  A  D ALRCPM
-------------------------------------------------------------------------------
    1 EMPLOYEES        30-MAY-2017         1,107     3  1  1  1  1  5  5   R  M
    2 VEHICLES         30-MAY-2017           773     3  1  1  1  1  5  5   R  M
    3 MISCELLANEOUS    30-MAY-2017         1,779     3  1  1  1  1  5  5   R  M
   60 EMPL-REF         30-MAY-2017         1,107     3  1  1  1  1  5  5   R  M
  251 SECURITY-FILE    30-MAY-2017             0     3  1  1  1  1  5  5   R  M
  254 USER-DATA-FILE   30-MAY-2017             0     3  1  1  1  1  5  5   R  M
  255 CHECKPOINT-FILE  30-MAY-2017             0     3  1  1  1  1  5  5   S  M


                                  Allocated blocks
 File Filename                    NI            UI            AC            DS
```

```
--------------------------------------------------------------------------------
    1 EMPLOYEES                    90            15            10            75
    2 VEHICLES                     40            20             2            40
    3 MISCELLANEOUS                50            10            10            50
   60 EMPL-REF                     90            15            10            75
  251 SECURITY-FILE                 2             2             1             5
  254 USER-DATA-FILE               24             2             6            57
  255 CHECKPOINT-FILE               1             1             6            32
--------------------------------------------------------------------------------
      Total                       297            65            45           334


                                   Unused blocks
 File Filename                     NI            UI            DS
--------------------------------------------------------------------------------
    1 EMPLOYEES                     4             2            18
    2 VEHICLES                     16            12            26
    3 MISCELLANEOUS                17             5            23
   60 EMPL-REF                      4             2            18
  251 SECURITY-FILE                 2             1             4
  254 USER-DATA-FILE               24             1            56
  255 CHECKPOINT-FILE               1             0            31
--------------------------------------------------------------------------------
      Total                        68            23           176
```

The column 'Extents' shows the number of logical extents currently assigned to the Normal Index (N), the Main/Upper Indices (U), the Address Converter (A) and Data Storage (D).

The column 'Pad' shows the block padding factors in percent defined for the Associator (A) and Data Storage (D) (please refer to the ASSOPFAC and DATAPFAC parameters of ADAFDU, ADAMUP or ADAORD for more detailed information).

The column 'Flags' contains the following information:

Subcolumn	Flag	Meaning
A	A	Indicates an Adam file
L	L	File is a LOB file
	B	File is a base file with a corresponding LOB file
R	R	ISN and space reusage enabled for the file
	I	ISN reusage, but no space reusage enabled for the file
	S	Space reusage, but no ISN reusage enabled for the file
C	C	Ciphering enabled for the file
P	P	Program Refresh enabled for the file
M	M	The file was modified since the last backup

If ISNs are to be reused, the ISNs of deleted records can be reassigned to new records. If space is to be reused, the space released within a block as a result of deleting a record can be reused for a

new record (please refer to the REUSE parameter of ADADBM or ADAFDU for more detailed information).

The second and third tables show the number of blocks allocated for Normal Index (NI) Main/Upper Indices (UI), Address Converter (AC) and Data Storage (DS) for each file. The remaining columns show the number of unused blocks in the Main/Upper Indices (UI) and Data Storage (DS).

COUNT

```
COUNT
```

This parameter displays the record count of the files in the database that you specify with the DBID parameter.

Example

```
adarep: count
Record Count of Database 2           20-APR-2017 16:08:52


 File Filename                    Records loaded
-------------------------------------------------------------------------------
    1 CHECKPOINT-FILE                          0
    2 SECURITY-FILE                            0
    3 USER-DATA-FILE                           0
    9 EMPLOYEES                            1,272
   11 EMPLOYEES-NAT                        1,107
   12 VEHICLES                               773
   13 MISCELLANEOUS                        1,779
   14 LOBFILE of 9                           210
```

DBID

```
DBID = number
```

This parameter selects the database to be used. Multiple DBIDs are supported within one session.

The DBID or CONNECTION_STRING parameter must be the first ADAREP parameter specified.

The parameters DBID and CONNECTION_STRING are mutually exclusive.

Example

```
adarep: dbid = 1, contents
       .
       .
adarep: dbid = 2, contents
       .
       .
adarep: dbid = 3, contents
```

[NO]FDT

```
[NO]FDT
```

If this parameter is set to FDT, the Field Definition Tables (FDTs) will be included in the status information subsequently displayed by the FILES function.

The default is NOFDT.

FILES

```
FILES = { * | (number [-number][,number [-number]]...) }
```

This function displays status information for the files selected.

Example

```
adarep: fdt
adarep: file=9
Database 189, File     9  (EMPLOYEES       )          20-APR-2026 16:32:30


Highest Index Level:          3     Padding Factors:       ASSO   5%, DATA   5%
Top ISN:                  1,272     Maximum ISN expected:                 8,191
SYFMAX :                      9
Records loaded:           1,272     Corresponding LOB file:                  14
Last FDT Modification:              20-APR-2026 16:32:30.934851
Last updated:                       20-APR-2026 16:32:31.134560

ISN reusage:     Enabled, inactive  Space reusage:    Enabled
Program refresh: Disabled           Ciphering:        Disabled
Modified since last backup
Recordspanning: Disabled


Container  Block  Extent    Extents in Blocks       Allocated         Unused
  File     Size   Type       from          to      Blocks   MB      Blocks   MB
-------------------------------------------------------------------------------
```

```
ASSO:
        2  32KB   AC        7,682       7,682           1    0           0    0
        1   8KB   NI           55         144          90    0          60    0
        1   8KB   UI          145         159          15    0           0    0

DATA:
        1  32KB   DS           14          45          32    1          21    0

--------------------------------------------------------------------------------

Field Definition Table:

   Level  | Name | Length | Format |   Options          | Flags   | Long name
-------------------------------------------------------------------------------------------------------------
  1       |  AO  |        |        |                    |         | PERSONNEL-DATA
   2      |  AA  |    8   |    A   | DE,UQ,NC,NN        | RP      | PERSONNEL-ID
   2      |  AB  |        |        |                    |         | ID-DATA
    3     |  AC  |    4   |    F   | DE                 |         | PERSONNEL-NO
    3     |  AD  |    8   |    B   | NU,HF              |         | ID-CARD
    3     |  AE  |    0   |    A   | NU,NV,NB,LA        |         | SIGNATURE
  1       |  BO  |        |        |                    |         | FULL-NAME
   2      |  BA  |   40   |    W   | NU                 |         | FIRST-NAME
   2      |  BB  |   40   |    W   | NU                 |         | MIDDLE-NAME
   2      |  BC  |   50   |    W   | DE,NU              | SP      | NAME
  1       |  CA  |    1   |    A   | FI                 |         | MAR-STAT
  1       |  DA  |    1   |    A   | FI                 |         | SEX
  1       |  EA  |    4   |    P   | DE,NC              |         | BIRTH
  1       |  FO  |        |        | PE                 |         | PRIVATE-ADDRESS
   2      |  FA  |   60   |    W   | NU,MU              |         | ADDRESS-LINE
   2      |  FB  |   40   |    W   | DE,NU              |         | CITY
   2      |  FC  |   10   |    A   | NU                 |         | POST-CODE
   2      |  FD  |    3   |    A   | NU                 |         | COUNTRY
   2      |  F1  |        |        |                    |         | PHONE-EMAIL
    3     |  FE  |    6   |    A   | NU                 |         | AREA-CODE
    3     |  FF  |   15   |    A   | NU                 |         | PRIVATE-PHONE
    3     |  FG  |   15   |    A   | NU                 |         | PRIVATE-FAX
    3     |  FH  |   15   |    A   | NU                 |         | PRIVATE-MOBILE
    3     |  FI  |   80   |    A   | DE,NU,MU           |         | PRIVATE-EMAIL
  1       |  IO  |        |        | PE                 |         | BUSINESS-ADDRESS
   2      |  IA  |   40   |    W   | NU,MU              |         | ADDRESS-LINE
   2      |  IB  |   40   |    W   | DE,NU              |         | CITY
   2      |  IC  |   10   |    A   | NU                 |         | POST-CODE
   2      |  ID  |    3   |    A   | NU                 |         | COUNTRY
   2      |  IE  |    5   |    A   | NU                 |         | ROOM NUMBER
   2      |  I1  |        |        |                    |         | PHONE-EMAIL
    3     |  IF  |    6   |    A   | NU                 |         | AREA-CODE
    3     |  IG  |   15   |    A   | NU                 |         | BUSINESS-PHONE
    3     |  IH  |   15   |    A   | NU                 |         | BUSINESS-FAX
    3     |  II  |   15   |    A   | NU                 |         | BUSINESS-MOBILE
    3     |  IJ  |   80   |    A   | DE,NU,MU           |         | BUSINESS-EMAIL
  1       |  JA  |    6   |    A   | DE                 | SB,SP   | DEPT
  1       |  KA  |   66   |    W   | DE,NU              |         | JOB-TITLE
```

```
  1       |  LO  |        |        | PE                 |         | INCOME
   2      |  LA  |    3   |    A   | NU                 | SP      | CURR-CODE
   2      |  LB  |    6   |    P   | NU                 | SP      | SALARY
   2      |  LC  |    6   |    P   | DE,NU,MU           |         | BONUS
  1       |  MA  |    4   |    G   | NU                 |         | TOTAL INCOME(EUR)
  1       |  NO  |        |        |                    |         | LEAVE-DATE
   2      |  NA  |    2   |    U   |                    | SP      | LEAVE-DUE
   2      |  NB  |    3   |    U   | NU                 | SP      | LEAVE-TAKEN
  1       |  OO  |        |        | PE                 |         | LEAVE-BOOKED
   2      |  OA  |    8   |    U   | NU                 |         | LEAVE-START
          |      |        |        | DT(DATE)           |         |
   2      |  OB  |    8   |    U   | NU                 |         | LEAVE-END
          |      |        |        | DT(DATE)           |         |
  1       |  PA  |    3   |    A   | DE,NU,MU           |         | LANGUAGE
  1       |  QA  |    7   |    P   |                    |         | LAST UPDATE
  1       |  RA  |    0   |    A   | NU,NV,NB,LB        |         | PICTURE
  1       |  SO  |        |        | PE                 |         | DOCUMENTS
   2      |  SA  |   80   |    W   | NU                 |         | ↩
DOCUMENT-DESCRIPTION
   2      |  SB  |    3   |    A   | NU                 |         | DOCUMENT-TYPE
   2      |  SC  |    0   |    A   | NU,MU,NV,NB,LB     |         | DOCUMENT
  1       |  TC  |   20   |    U   | DT(TIMESTAMP)      |         | CREATION TIME
          |      |        |        | SY=TIME,CR         |         |
  1       |  TU  |   20   |    U   | MU                 |         | LAST UPDATES
          |      |        |        | DT(TIMESTAMP)      |         |
          |      |        |        | SY=TIME            |         |
-------------------------------------------------------------------------------------------------

   Type   | Name | Length | Format |   Options          | Parent field(s)   Fmt | ↩
Long name
-------------------------------------------------------------------------------------------------
 COLL     |  CN  |1,176   |        | NU,HE              | BC  de@collation=phone| ↩
PHONEBOOK
          |      |        |        |                    |     book              |  ↩
                                                                                   ↩

          |      |        |        |                    |     PRIMARY           |  ↩
                                                                                   ↩

 SUPER    |  H1  |    5   |    B   | NU                 | NA (   1 -   2 )   U  | ↩
SUPERDESCRIPTOR
          |      |        |        |                    | NB (   1 -   3 )   U  |
 SUB      |  S1  |    2   |    A   |                    | JA (   1 -   2 )   A  | ↩
SUBDESCRIPTOR 1
 SUPER    |  S2  |   46   |    A   | NU                 | JA (   1 -   6 )   A  | ↩
SUBDESCRIPTOR 2
          |      |        |        |                    | BC (   1 -  40 )   W  |
 SUPER    |  S3  |    9   |    A   | NU,PE              | LA (   1 -   3 )   A  | ↩
SUBDESCRIPTOR 3
          |      |        |        |                    | LB (   1 -   6 )   P  |
-------------------------------------------------------------------------------------------------
                            Referential Integrity
```

```
-----------------------------------------------------------------------------------------------------------
  Type    | Name | Refer. | Primary| Foreign | Rules
          |      | file   |  field |  field  |
-----------------------------------------------------------------------------------------------------------
PRIMARY   |  HO  |    12  |   AA   |   AC    | DELETE_NOACTION  UPDATE_NOACTION
-----------------------------------------------------------------------------------------------------------
```

The FILES parameter shows the file number and file name as well as the date and time the file was loaded, the highest index level, the padding factors for ASSO and DATA, the highest and maximum ISNs, the number of records loaded, the corresponding base file number or LOB file number if it exists, as well as the switches for ISN reusage, space reusage, program refresh and ciphering. The time and date of the last FDT modification are also displayed.

The layout of the ASSO and DATA elements of a file are displayed: the block size on which the various list elements are stored, the location of these extents and the number of corresponding blocks/megabytes allocated or unused.

In addition, the FDT function displays the Field Definition Table of the file.

The flags which can be displayed in the Field Definition Table are as follows:

Flag	Meaning
HY	the field is part of a hyperdescriptor
P	the field is phoneticized
RF	the field is the foreign key of a referential constraint
RP	the field is the primary key of a referential constraintreferential constraint
SB	part of this field is subdescriptor
SP	part of superdescriptor

[NO]OUTPUTFDT

```
[NO]OUTPUTFDT
```

If this option is set to OUTPUTFDT, the FDT information of the file is written to the sequential file REPFDT*n*, where *n* is the file number.

The default is NOOUTPUTFDT.

You must use the keyword OUTPUTFDT before the FILES parameter.

FREE_SPACE

```
FREE_SPACE
```

This function displays a summary of free blocks in ASSO and DATA. This is a subset of the information that is displayed by the LAYOUT function.

Example

```
adarep: free_space

Free space of Database 76       28-NOV-2006 12:51:24



Container       Extents in Blocks   Number of  Block
  File           from          to    Blocks    Size
-------------------------------------------------------------------------------

ASSO:
      1-2         611       1,546         936  2,048

DATA:
        1         245         768         524  4,096
        2         769         868         100  3,072
      3-4         869         888          20  6,144
```

[NO]FULL

```
[NO]FULL
```

If FULL is specified together with FDT, the following additional information is displayed for the FDT:

- Dropped fields are included in the display of the fields of the file (but without the field names).
- The ICU version is included in the display of collation descriptors.

The default is NOFULL.

[NO]LARGE

```
[NO]LARGE
```

This parameter changes the layout of adarep utility with the file parameter.

This layout can display large values. Default is NOLARGE.

Example

```
adarep: large
adarep: file=9
Database 44, File    12  (VEHICLES        )          15-JUL-2014 17:46:11


Highest Index Level:          3     Padding Factors:       ASSO   5%, DATA   5%
Top ISN:                    773     Maximum ISN expected:                73,727
Records loaded:             773
Last FDT Modification:              15-JUL-2014 17:46:12.363000

ISN reusage:     Enabled, inactive  Space reusage:    Enabled
Program refresh: Disabled           Ciphering:        Disabled
Modified since last backup
Recordspanning: Disabled

Container  Block  Extent          Extents in Blocks               Allocated       ↩
          Unused
  File     Size   Type          from             to         Blocks         MB     ↩
    Blocks         MB
-----------------------------------------------------------------------------------------



ASSO:
        2  32KB   AC             100      4,483,647      4,483,548    140,110     ↩
 4,475,962    139,873
        1   8KB   NI             100      4,483,647      4,483,548     35,027     ↩
 4,483,473     35,027
        1   8KB   UI             100      4,483,647      4,483,548     35,027     ↩
 4,483,434     35,026

DATA:
        1  32KB   DS             100      4,483,647      4,483,548    140,110     ↩
 4,483,389    140,105

-----------------------------------------------------------------------------------------
```

LAYOUT

```
LAYOUT
```

This function displays a summary of the blocks in ASSO and DATA and reports lost blocks. Lost blocks are blocks that are not listed in the Free Space Table (FST) and are not allocated to a file, the DSST or the database's global area. This function also reports double-allocated blocks.

Example

```
adarep: layout

Layout of Database 76           28-NOV-2006 12:51:24


Container       Extents in Blocks   Number of  Block  Extent  File
  File           from          to    Blocks    Size   Type   Number
-------------------------------------------------------------------------------

ASSO:
        1           1          30          30  4,096  CB
        1          31          31           1  4,096  FCB       1
        1          32          32           1  4,096  FDT       1
        1          33          35           3  4,096  AC        1
        1          36          36           1  4,096  UI        1
        1          37          37           1  4,096  NI        1
        1          38          38           1  4,096  FCB       2
        1          39          39           1  4,096  FDT       2
        1          40          40           1  4,096  AC        2
        1          41          42           2  4,096  UI        2
        1          43          43           1  4,096  NI        2
        1          44          44           1  4,096  FCB       3
        1          45          45           1  4,096  FDT       3
        1          46          48           3  4,096  AC        3
        1          49          50           2  4,096  UI        3
        1          51          62          12  4,096  NI        3
        1          63         152          90  4,096  NI        9
        1         153         167          15  4,096  UI        9
        1         168         168           1  4,096  FCB       9
        1         169         169           1  4,096  FDT       9
        1         170         170           1  4,096  NI       14
        1         171         172           2  4,096  UI       14
        1         173         173           1  4,096  FCB      14
        1         174         174           1  4,096  FDT      14
        1         175         264          90  4,096  NI       11
        1         265         279          15  4,096  UI       11
        1         280         280           1  4,096  FCB      11
        1         281         281           1  4,096  FDT      11
        1         282         321          40  4,096  NI       12
        1         322         341          20  4,096  UI       12
```

```
        1         342         342           1  4,096  FCB      12
        1         343         343           1  4,096  FDT      12
        1         344         393          50  4,096  NI       13
        1         394         403          10  4,096  UI       13
        1         404         404           1  4,096  FCB      13
        1         405         406           2  4,096  FDT      13
        1         407       2,560       2,154  4,096  FREE
        2       2,561       2,561           1 32,768  DSST
        2       2,562       2,562           1 32,768  AC        9
        2       2,563       2,563           1 32,768  AC       14
        2       2,564       2,572           9 32,768  AC       11
        2       2,573       2,581           9 32,768  AC       12
        2       2,582       2,582           1 32,768  AC       13
        2       2,583       2,880         298 32,768  FREE

DATA:
        1           1           4           4 32,768  DS        1
        1           5           5           1 32,768  DS        2
        1           6          13           8 32,768  DS        3
        1          14          45          32 32,768  DS        9
        1          46         170         125 32,768  DS       14
        1         171         202          32 32,768  DS       11
        1         203         212          10 32,768  DS       12
        1         213         222          10 32,768  DS       13
        1         223         640         418 32,768  FREE
```

LAYOUT provides a summary of all blocks in ASSO and DATA. The locations and lengths of sections of contiguous blocks, the block size, the type of usage and the numbers of the corresponding files are displayed. These blocks may be free (FREE) or used for the Global Blocks (CB), the File Control Block (FCB), the FCB extension (FCBE), the FCB Root Block (FCBR), the Field Definition Table (FDT), the Free Space Table (FST), the Data Space Storage Table (DSST), the Normal Index (NI), the Upper/Main Index (UI), the Address Converter (AC) or the Data Storage (DS).

Note: The first FCBR block and the first FST block are part of the global blocks. For this reason, the layout only displays FCBR and FST blocks if the database contains more than one of these blocks.

SUMMARY

```
SUMMARY
```

SUMMARY provides general information about the database and the physical layout of ASSO, DATA and WORK.

Example

```
adarep: summary

Summary of Database 76          28-JUL-2020 12:51:24


DATABASE NAME                    DOKU-DATABASE
DATABASE ID                         76
MAXIMUM NUMBER OF FILES             30
SYSTEM FILES                         1 (CHK),     2 (SEC),     3 (USR)
                                   150 (RBAC)
ACTUAL FILES LOADED                  6
AC SIZE                              3
CURRENT PLOG NUMBER                  8
CURRENT CLOG NUMBER                  0
SECURITY                         ACTIVE
ENCRYPTION                       AES_256_XTS
KMSTARGET                        FILE
KEKNAME                          ↩
1598876857F83BB64A01916FFA3ACF9D39DB18537E80E3E48FD8A2333A3D77C8



Container  Device      Extents in Blocks    Number of      Block    Total Size
  File     Type         from          to     Blocks        Size     (Megabytes)
-------------------------------------------------------------------------------

 ASS01     file            1       1,536        1,536      2,048           3.00
 ASS02     file        1,537       1,546           10      2,048           0.02

 DATA1     file            1         768          768      4,096           3.00
 DATA2     file          769         868          100      3,072           0.29
 DATA3     file          869         878           10      6,144           0.06
 DATA4     file          879         888           10      6,144           0.06

 WORK1     file            1       1,365        1,365      3,072           4.00

-------------------------------------------------------------------------------
                                                                         10.43
                                                                         =====
```

The security information is only displayed if database security has been activated. Otherwise, the information is not displayed.

The encryption information, consisting of encryption algorithm, KMS target (Key Management System) and KEK name (Key Encryption Key), is only displayed if the database is encrypted. Otherwise, the information is not displayed.

The RBAC system file is only displayed if it has been defined. Otherwise, the information is not displayed.

Note: If the database is running in READONLY mode, WORK1 is not displayed.

22 ADASEL (PLOG Selection Tool)

- Functional Overview 344
- Procedure Flow 345
- Checkpoints 346
- Control Parameters 346
- Query Syntax 350
- Examples and Special Cases 364

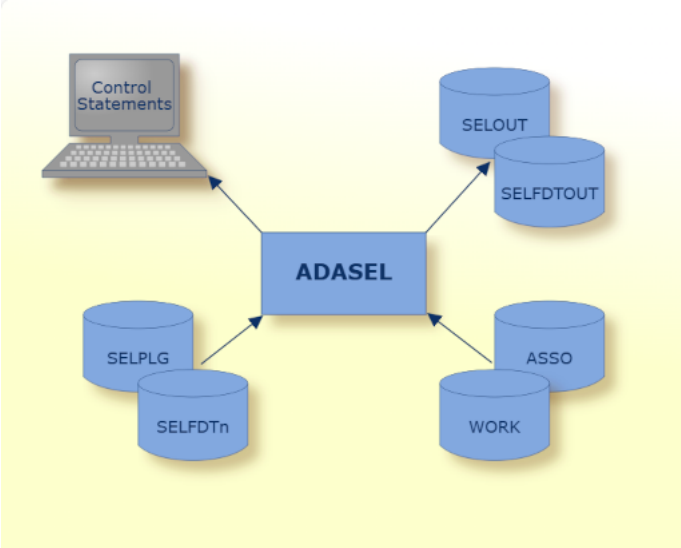
This chapter describes the utility "ADASEL".

The following topics are covered:

Functional Overview

The ADASEL utility selects records from Protection log files and WORK containers.

Procedure Flow



Data Set	Environment Variable / Logical Name	Storage Medium	Additional Information
Associator	ASSOx	Disk	n/a
Alternate FDTs	SELFDTn	Disk	Only required if you process a query with keyword FDTINPUT, to read the FDT from a file (file number 'n'), instead of the database. In case all files are defined as alternate FDT, then no database is required.
Optional output	SELOUT	Disk	The output file location can be defined here. Otherwise, the output is written to stdout.
Optional FDT output	SELFDTOUT	Disk	The FDT output file location can be defined here. The FDTs used in the selection will be exported here, if the FDT parameter is selected.
Protection log	SELPLG	Disk	The sequential input PLOG file is defined here.
Control statements	stdin	-	Utilities Manual
ADASEL report	stdout	-	Messages and Codes
Work storage	WORK1	Disk	Utilities Manual

The sequential file SELPLG can have multiple extents. For information about sequential files with multiple extents, see *Adabas Basics*, *Using Utilities* in the Adabas documentation.

Checkpoints

The utility writes no checkpoints.

Control Parameters

The following control parameters are available:

```
D   DATASET = { PLOG | WORK }

    DBID = number

    [NO]DELTA

    FDT

    FILE = <filepath>

    FORMAT = { CSV | DISPLAY | JSON | JSONPRETTY }

    SELECT = <select query>

    SELECTALL { = FDTINPUT }

    SEPARATOR = character | \character

D   [NO]STAT
```

Note: The query is case insensitive.

DATASET

```
DATASET = { PLOG | WORK }
```

This parameter selects the file containing the protection log information to be processed. The keyword can take the values PLOG or WORK, where PLOG is the default.

Example:

```
adasel: dataset=work
```

The WORK dataset is selected.

DBID

```
DBID = number
```

This parameter behaves differently based on the dataset defined.

- PLOG dataset

  It can be used to select a different database than the one defined in the PLOG file. The default is the database number defined in the PLOG file. If defined, the SELPLG setting in the *DB.INI* file can be used.

- WORK dataset

  It must be used to define the database containing the WORK dataset. There is no default in this case.

Examples:

```
adasel: dbid=200
```

The database currently being used is database 200. The PLOG dataset is selected by default.

```
adasel: dbid=200 dataset=WORK
```

The database currently being used is database 200. The WORK dataset is selected.

[NO]DELTA

```
[NO]DELTA
```

This parameter indicates whether only changed fields are displayed after an update. The produced output remains the same for inserted and deleted records. The delta record will be built for updated records, where only modified field values are generated. The default is NODELTA.

FDT

```
FDT
```

The file FDTs used in the selection will be exported as JSON with this parameter. This will be written to a file, if the environment variable SELFDTOUT is defined. Otherwise, it is printed to stdout.

FILE

```
FILE = <filepath>
```

This function reads a text file containing one or more queries on the dataset.

Example:

```
adasel: file=query.in
```

The content of the file *query.in* will be executed on the dataset.

FORMAT

```
FORMAT = { CSV | DISPLAY | JSON | JSONPRETTY }
```

This parameter selects the output format. The selection CSV creates a “comma-separated values” output. If DISPLAY is selected, then the output format will be similar to the mainframe output. The JSON and JSONPRETTY selections create a JSON file in CDC style. The JSON format is for machine processing, while the JSONPRETTY format is human readable. The default is DISPLAY.

SELECT

```
SELECT = <select query>
```

The select function executes a text string containing one or more queries on the dataset.

Example:

```
adasel: select=select all from file 1 display all end
```

The specified query will be executed on the dataset.

SELECTALL

```
SELECTALL { = FDTINPUT }
```

The 'select all' function creates an output containing the complete content of the dataset. This happens without any library call. If the optional parameter FDTINUT is set, then FDT information will be retrieved from the files defined in the environment variable(s) SELFDTn.

SEPARATOR

```
SEPARATOR = character | \character
```

This parameter defines the separator character for the CSV output. The default is comma.

[NO]STAT

```
[NO]STAT
```

This function enables or disables the display of performance statistics at the end of the execution. The default is NOSTAT.

Example:

```
adasel: stat

-----------------------------------------------------------------------------------------------
Total run time:           84,371,489 us,         84.37 sec
  Library init/next:             814 us
  Processing time:       67,242,964 us,        67.24 sec,   14,581,622 PLOG records
    Process FDT cache:     1,202,276 us,          1.20 sec,     1,199,999 hits,   ↩
 avg:     1 us
                                                                                  100 % hit ↩
rate, 1 file(s)
    Decompress time:       3,707,651 us,         3.70 sec,     1,200,000 records, ↩
avg:     3 us
    Library run:          33,869,021 us,        33.86 sec,     1,200,000 calls,  ↩
 avg:    28 us
    Print time:            8,331,534 us,          8.33 sec,     1,200,000 records
  Library finish:                 36 us
----- Library measurements ------------
Init/Next Total:                 400 us
  Init/Next Set-Up:              147 us
  Compile query:                 238 us
  Stackm. Set-Up:                  2 us
  Criterion handover:             11 us
Run Total:                45,322,926 us
  Run Setup:                  73,677 us
  Parsing PLOG:            6,271,616 us
  Stack machine:          12,986,070 us
  Run Handover:           25,991,563 us

-----------------------------------------------------------------------------------------------
```

Query Syntax

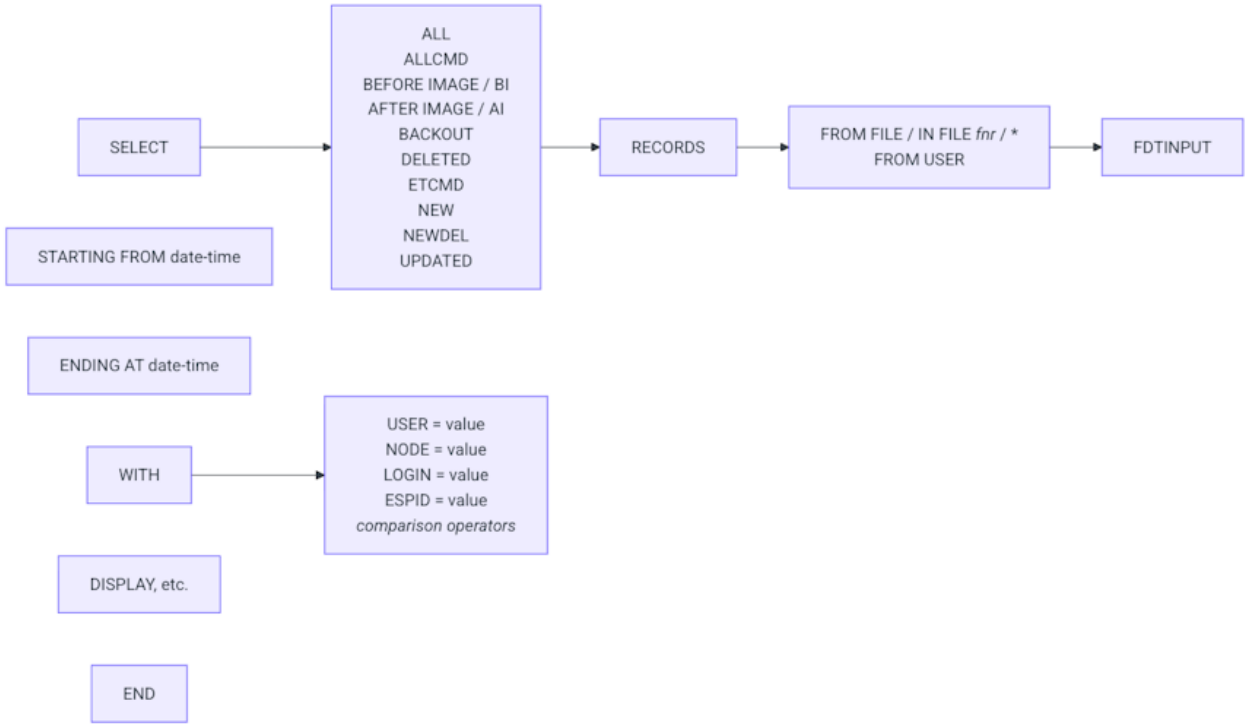
The following paragraphs detail the currently implemented keywords, their control flow, and their function. The keywords are given in capital letters but can be written in any case.

The only exceptions from this rule are field names. These are case sensitive, making AA1 different from Aa1, aA1, and aa1. On top of that, special formatting and naming rules apply, which are explained further down. See *FDT Record Structure* for more naming recommendations.

- Control Flow SELECT
- Control Flow IF
- Comparison Operators
- Control Flow DISPLAY
- Control Flow OUTPUT
- Control Flow NEWPAGE

- Control Flow SKIP

Control Flow SELECT



`SELECT`

- Marks the beginning of a query
- Must be paired with an `END` statement
- Select options:
  - `ALL`

    Is a combination of all options except `BACKOUT` and `ETCMD`
  - `ALLCMD`

    Is a combination of all select options
  - `BEFORE IMAGE / BI`
    - Selects before images derived from A1 and E1 commands
    - Alternatively, the shorter form `SELECT BI` can be used
  - `AFTER IMAGE / AI`
    - Selects after images derived from A1 and N1 commands
    - Alternatively, the shorter form `SELECT AI` can be used

- `BACKOUT`

  Selects before and after images derived from BT commands
- `DELETED`

  Selects before images derived from E1 commands
- `ETCMD`

  Selects ET information
- `NEW`

  Selects after images derived from N1 commands
- `NEWDEL`

  Selects after images derived from N1 and E1 commands
- `UPDATED`

  Selects before and after images derived from A1 commands

- `RECORDS`

  Optional keyword, may be omitted
- Mode options:

  The mode options are mutually exclusive, only one can be used at a time
  - `FROM FILE / IN FILE`
    - Marks the selection of blocks based on file number
    - * (asterisk) can be used for searching all files
    - File number or * have to follow immediately after
    - `FROM FILE` and `IN FILE` have the same function, they act like synonyms
  - `FROM USER`
    - Marks the selection of blocks by user
    - Username can be supplied in `WITH` statement
    - If username is omitted, blocks from all users are displayed

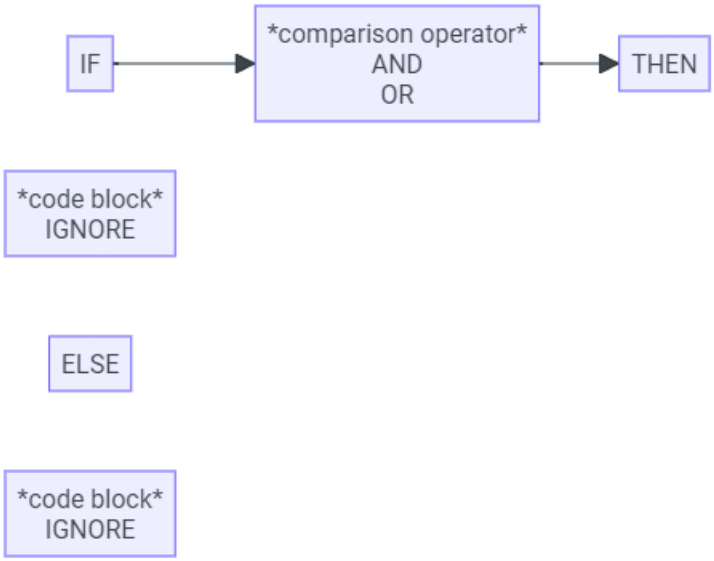
      This can act as an alternative to `SELECT FROM FILE *`
- `FDTINPUT`
  - Marks the use of an external FDT
  - The FDT has to be supplied in the environment variable SELPLGn
- `STARTING FROM`

- `ENDING AT`
  - Sets a timeframe for the selection of a PLOG record
  - The borders are inclusive
  - The date and time can be supplied in the following ways:
    - Gregorian in the format YYYYMMDD/HHMMSS
    - Julian in the format J(YYYYDDD/HHMMSS)
  - The time zone of the supplied date and time will be the one of the local machine, but then converted to a Unix timestamp in UTC
  - If omitted, it defaults to the timeframe
    - Unix timestamp 0 (January 1, 1970 00:00:00 (UTC))
    - Unix timestamp 9223372036854775807 (max_int64, theoretically August 17, 292278994 07:12:55 (UTC))
  - If given, overrides the corresponding default value
- `WITH`
  - Acts like a second stage for the `SELECT` statement
  - For more restrictive and faster processing, less record decompression, and fewer query calls, evaluated PLOG blocks can be limited using the following parameters with =:
    - `USER`, e.g. `USER = 'testuser'`
    - `NODE`, e.g. `NODE = 'testnode'`
    - `LOGIN`, e.g. `LOGIN = 'nodeuser'`
    - `ESPID`, e.g. `ESPID = 123456`
  - To avoid logical inconsistencies, only one of the above parameters is allowed
  - Conditional statements can only be joined with `AND`
  - More elaborate conditional statements can be specified in an `IF` block
- `END`

  Marks the end of a `SELECT` statement or query

Note: The `RECSYNC` keyword is always set to `TRUE` for correct processing of PLOG blocks and no longer needs to be specified by the user. Therefore, it is omitted from the syntax.

Control Flow IF



`IF`

- Provides a conditional logic similar to other programming languages
- Accepts every comparison operator
- Comparison operators can be connected with `OR` as well as `AND`
- `IF`s can also be nested
- When `DISPLAY` clause comes beforehand, use `IFF` to avoid ambiguities in case of a single-line query or when the `DISPLAY` clause is on the same line
- `THEN`
  - Keyword is optional
  - Code block that is executed when the conditional statement is true
- `ELSE`

  Code block that is executed when the conditional statement is false
- `IGNORE`

  Stands in for an empty code block
- `DO / DOEND`
  - Acts similar to curly brackets in other programming languages
  - Can be used to enclose a multi-line code block for `THEN` or `ELSE` part
  - Can be omitted if code is on a single line only

Comparison Operators

- Evaluates a statement to `TRUE` or `FALSE`
- Evaluates fields against literals or literals against literals
  - The evaluation depends on the format of the first literal value
  - Statement will either evaluate numerical comparison or string comparison
    - `AA = 'Test'` => string comparison
    - `AB(1) >= 10` => numerical comparison
    - `USER < '*adatst '` => string comparison
    - `SESSNUM != 2` => numerical comparison
  - If the value formats do not match a runtime error is thrown
  - If two fields are compared, a numerical comparison is attempted first, if it fails a string comparison will occur
  - If a field which is not contained in a record is used in a comparison, the statement will default to `FALSE`
- `EQUAL`

  Can be specified by `EQUALS`, `EQUAL`, `EQ`, or =
- `GREATER THAN`

  Can be specified by `GREATER THAN`, `GT`, or >
- `LESS THAN`

  Can be specified by `LESS THAN`, `LT`, or <
- `GREATER THAN OR EQUAL`

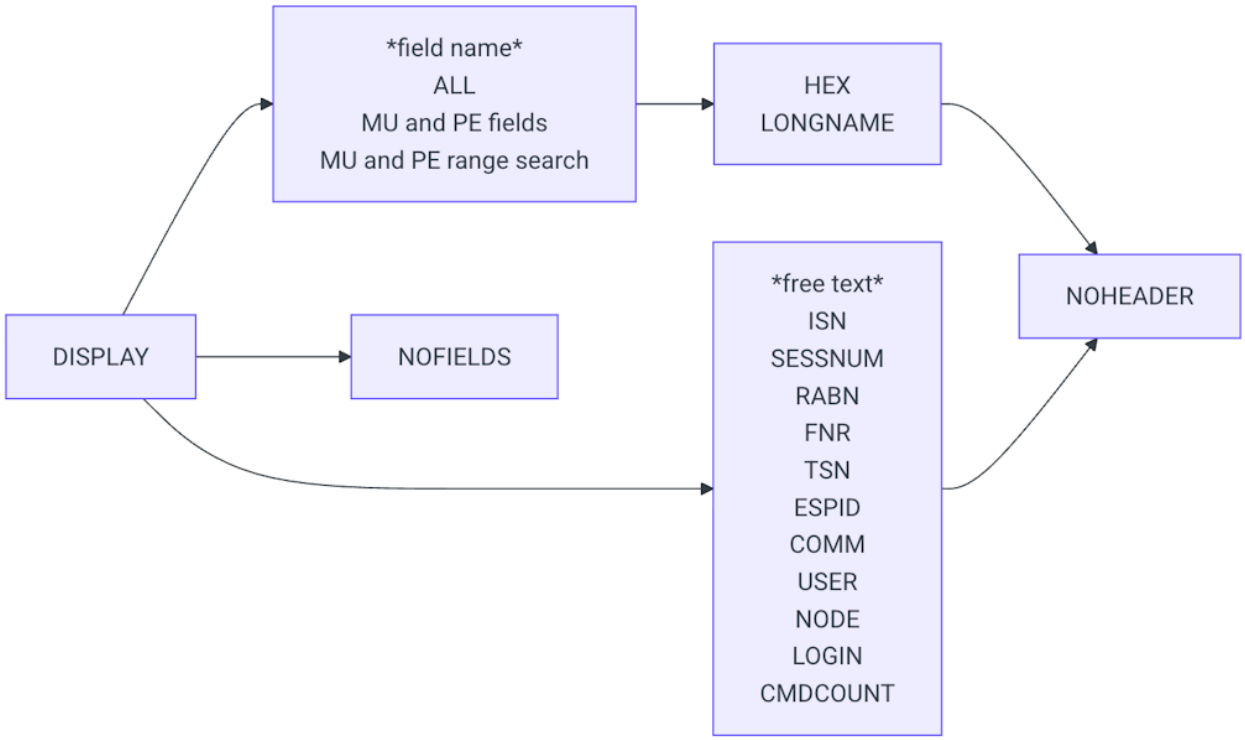
  Can be specified by `GREATER EQUAL`, `GE`, or >=
- `LESS THAN OR EQUAL`

  Can be specified by `LESS EQUAL`, `LE`, or <=
- `NOT EQUAL`

  Can be specified by `NOTEQUAL`, `NOT EQUAL`, `NE`, ¬=, ^=, or !=
- `THRU`
  - Specifies a range of values
  - The borders are inclusive
  - Syntax can only be used with equal operator, e.g.
    - `AA EQ 10 THRU 50`

  - `USERID EQ 'USER_0 ' THRU 'USER_10'`
- `CHANGES`
  - Tracks if a field changes from before image to after image during an update command, for example:
    - `AA CHANGES`
    - `BB2(3) CHANGES`
  - `TRUE` if field changes, has been added, or removed
  - `FALSE` if field stays the same or does not belong to an update command
  - Only one field can be used per query
  - Select options, like `SELECT BEFORE IMAGE` or `SELECT AFTER IMAGE` can be used to only display changed before or after images
- `OR`
  - Joins two conditional operations together
  - Evaluates to `TRUE` if one of the two or both operations are `TRUE`
- `AND`
  - Joins two conditional operations together
  - Evaluates to `TRUE` if both operations are `TRUE`
  - When used in conjunction, `AND` takes precedence over `OR`, similar to how multiplication is "faster" than addition

Control Flow DISPLAY



`DISPLAY`

- Prints content to `STDOUT` or `SELOUT`, if specified.
- Must be enclosed in a `SELECT` statement.
- Prints a header section for each record.
- For multi-line queries, the `DISPLAY` clause ends after a new line
- Is followed by an output specifier:
  - `free text`

    Prints a free text, text has to be encased in quotation marks.
  - `ISN`

    Prints the ISN of the record.
  - `SESSNUM`

    Prints the session number of the record.
  - `RABN`

    Prints the Relative Adabas Block Number (RABN) of the record.
  - `FNR`

Prints the file number of the record.

- `COMM`

  Prints the communication ID of the record.

- `TSN`

  Prints the transaction number of the record.

- `ESPID`

  Prints the process ID of the record.

- `USER`

  Prints the name of the Adabas user who modified the data in the record.

- `NODE`

  Prints the name of the node (executing computer) on which the record was modified.

- `LOGIN`

  Prints the login ID of the user on the executing computer who modified the record.

- `CMDCOUNT`

  Prints the sequence number of the command that created or modified the PLOG block.

- `field name`

  Prints the content of one field.

  Field names must follow the patterns [A-Za-z][A-Za-z0-9], [A-Za-z][A-Za-z0-9]d+ or [A-Za-z][A-Za-z0-9]d+(d+), for example, AA, zA5, bb5(10).

  If the specified field is missing from a record, it will not be displayed.

- `ALL`

  Prints all data fields in the record.

- MU and PE fields
  - Format `XXI(J)`:
    - `XX`: field specifier
    - `I`: PE number
    - `J`: MU number
  - `XXJ / XXI / XXI(J)`: Prints the specified field.
  - `XX1-I / XX1-J`: Prints all MU or PE fields in the given range.

e.g. `AA1-10`: Prints the fields `AA1` through `AA10`.

- `XX1-N`: Searches and prints MU or PE fields in the given range.
- `XXI(1-n)`: Prints all MU fields in the given range for a given PE field `I`.

  e.g. `AA5(1-10)`: Prints the fields `AA5(1)` through `AA5(10)`.
- `XXI(1-N)`: Searches and prints all MU fields for a given PE field `I`.
- `XX1-n(J)`: Prints the given MU field `J` for a given range of PE fields.

  e.g. `AA1-10(5)`: Prints the fields `AA1(5)` through `AA10(5)`.
- `XX1-N(J)`: Searches and prints all PE fields that have a given MU field `J`.
- `XX1-N(1-N)`: Searches and prints all PE and MU fields.

- `HEX`
  - Can be specified after `ALL` as well as a regular MU or PE field and a MU or PE range search.

    e.g. `DISPLAY AA HEX`, `DISPLAY AN1(2) HEX`, `DISPLAY ALL HEX`, or `DISPLAY AA1-N HEX`.
  - Outputs the field value in hexadecimal representation, with each byte preceded by a `^` character.

    e.g. 'Hello' will be displayed as `^48^65^6c^6c^6f`.
- `LONGNAME`
  - Can be specified after `ALL` as well as a regular MU or PE field and a MU or PE range search.

    e.g. `DISPLAY AA LONGNAME`, `DISPLAY AN1(2) LONGNAME`, `DISPLAY ALL LONGNAME`, or `DISPLAY AA1-N LONGNAME`.
  - Prints a more descriptive long name of the corresponding field.
  - Long names cannot be used in conditional statements and are for informational purposes only.
- `NOHEADER`
  - Switches off the output of the header.
  - The header will be turned off for all output produced.

    Has no effect for the options `format=json` or `format=jsonpretty`.
  - `NOHEADER` only needs to be specified once per `SELECT` statement.
- `NOFIELDS`
  - Suppresses record decompression and output of fields.
  - Cannot be used in conjunction with any other keywords in `DISPLAY`.
  - Since no record decompression happens, fields, e.g. `AA = 10` or `AB(1) != 100` cannot be used in conditional statements within `IF` or `WHEN`.

Only header information such as SESSNUM, ISN, RABN, FNR, COMM, TSN, CMDCOUNT, ESPID, USER, and LOGIN are allowed.

Notes:

1. Field names are case sensitive. This means, as mentioned above, a field AA1 is different from Aa1, aA1, and aa1. When typing a query, this must be kept in mind. A field name cannot start with a number or a special character, making, for example, 1AA or @ab invalid. Field names must follow the patterns [A-Za-z][A-Za-z0-9], [A-Za-z][A-Za-z0-9]d+ or [A-Za-z][A-Za-z0-9]d+(d+). Furthermore, the Adabas documentation recommends under *FDT Record Structure* that names which are English prepositions or articles such as AN, AT, BY, IF, IN, OF, ON etc. should not be used because of a possible conflict with syntactical terms used by Natural. These field names will still be supported by ADASEL, though, should the user choose to go against this recommendation.
2. HEX and LONGNAME can be combined. For example, DISPLAY ALL HEX LONGNAME or DISPLAY AA LONGNAME HEX, DISPLAY AA1-N LONGNAME HEX, or DISPLAY AA1-N(1-N) LONGNAME HEX.
3. Record decompression is a computationally intensive operation, yet it is essential for evaluating and displaying field information. Suppressing it entirely can substantially accelerate execution time. This approach proves beneficial when only header information is needed or during data pre-assessment.
4. Each query is compiled and executed to evaluate the contents of a PLOG block. If evaluation can be simplified, the execution of the compiled query is skipped. This can also speed up execution significantly.

The output of a display command contains a header section and an output body. An example in the main frame format is as follows:

```
SELECT ALL FROM FILE 1
    DISPLAY 'GIVE ME FIVE' USER SESSNUM ISN ALL
END

 SNR   FNR       ISN       COMM    TSN   USER     LOGIN   AFTER IMAGE  2026-02-12 ↩
12:32:42
   1     1       4626      4       1     *adatst  xxxx
      - GIVE ME FIVE
      USER *adatst
      SESSNUM 1
      ISN 4626
      AA 100
      AD USER_0
      AF
      AG
      AN XY
```

For the main frame format, the header section contains the following fields:

- session number
- file number
- ISN
- communication ID
- transaction number
- user ID
- login ID
- record type
- time stamp of the data block

The body of the output lists the desired output fields as described in the query. In the example, e.g. the output specifier `ISN` results in the display of the corresponding number and the specifier `ALL` lists all the data fields contained in the block. The result of each output specifier will always be printed in a separate line. It is possible to combine multiple specifiers in one `DISPLAY` command or have multiple `DISPLAY` commands.

The same example in JSON format is as follows:

```
{
    {
    "PLOG":[
        {
            "action":"insert",
            "after":{
                "sessnum":1,
                "time":1770895962,
                "rabn":138,
                "isn":4626,
                "fnr":1,
                "comm":4,
                "tsn":1,
                "user":"*adatst",
                "login":"xxxx",
                "fields":{
                    "-":"GIVE ME FIVE",
                    "user":"*adatst ",
                    "sessnum":"1",
                    "isn":"4626",
                    "AA":"100",
                    "AD":"USER_0",
                    "AF":" ",
                    "AG":" ",
                    "AN":"XY"
                }
            }
```

```
        }
    ]
}
```

The JSON format lists all blocks as an array structure. This structure is machine readable and can be used to interface with other applications, programs, scripts, etc. The option `format=json` produces a compact output which is more memory efficient. The option `format=jsonpretty` produces the output above with tabs and line breaks for better readability.

The same example in CSV format is as follows:

```
action,session,time,rabn,isn,fnr,comm,tsn,user,login,fields
AI,1,1783952665,138,4626,1,4,1,*adatst ,xxxx    ,AA=100,AD=USER_0,AF= ,AG= ,AN=XY
AI,1,1783952665,151,4627,1,4,1,*adatst ,xxxx    ,AA=101,AF= ,AG= ↩
,AI1=ABC,AI2=DEF,AI3=GHI,AN=ZZ
AI,1,1783952665,151,4628,1,4,1,*adatst ,xxxx    ,AA=102,AF= ,AG= ↩
,AI1=ABC,AN=AA,AT1(1)=5,AZ1=LT
AI,1,1783952665,151,4629,1,4,1,*adatst ,xxxx    ,AA=103,AF= ,AG= ↩
,AR1=4,AT1(1)=5,AR2=6,AZ1=LT
BI,1,1783952665,138,4626,1,4,1,*adatst ,xxxx    ,AA=100,AD=USER_0,AF= ,AG= ,AN=XY
AI,1,1783952665,138,4626,1,4,1,*adatst ,xxxx    ,AA=100,AD=USER_1,AF= ,AG= ↩
,AI1=ZZ,AN=XY
BI,1,1783952665,151,4627,1,4,1,*adatst ,xxxx    ,AA=101,AF= ,AG= ↩
,AI1=ABC,AI2=DEF,AI3=GHI,AN=ZZ
AI,1,1783952665,151,4627,1,4,1,*adatst ,xxxx    ,AA=202,AF= ,AG= ↩
,AI1=ABC,AI2=XYZ,AI3=GHI,AI4=OCH,AN=ZZ
BI,1,1783952665,138,4626,1,4,1,*adatst ,xxxx    ,AA=100,AD=USER_1,AF= ,AG= ↩
,AI1=ZZ,AN=XY
```

The CSV format list all blocks in a table structure and can be produced with the option `format=csv`. The output is either machine readable or can be evaluated with a spreadsheet program, such as Microsoft Excel. The content of the header section is identical to the main frame and json format. Selected records with header and field information follow below.

Control Flow OUTPUT

The `OUTPUT` command on the mainframe version of ADASEL was meant to produce a machine-readable hex dump.

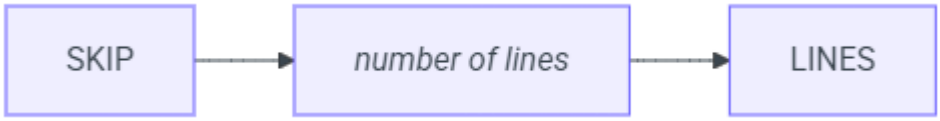
For the Linux version, use the option `format=json` or `format=jsonpretty`.

Control Flow NEWPAGE

```
NEWPAGE
```

- In the mainframe version, it was meant to create a page break or feed paper through to the next top of the page.
- For the Linux version, the keyword is parsed for backwards compatibility, but no new page is generated.

Control Flow SKIP



`SKIP`

- Produces line breaks in the output
- If ADASEL is called with the option `format=json` or `format=jsonpretty`, no line break will be produced
- Number of lines:
  - Specifies the number of lines to be skipped
  - Can be omitted

    If omitted, it will produce 1 line break
- `LINES`
  - Closes `SKIP` instruction
  - Keyword can be omitted
- Examples:
  - `SKIP`: produces one line break
  - `SKIP 2`: produces two line breaks
  - `SKIP 3 LINES`: produces three line breaks

Examples and Special Cases

The following section lists examples and special cases.

Multi-line query with DISPLAY and IF

In this example a `DISPLAY` clause is directly followed by an `IF` clause. The first case is a multi-line query, the second case a single-line query.

```
SELECT ALLCMD FROM FILE *
 DISPLAY AA
 IF ISN = 1
  DISPLAY BB
END

---

SELECT ALLCMD FROM FILE * DISPLAY AA IFF ISN = 1 DISPLAY BB END
```

For a multi-line query, the `DISPLAY` clause ends at the line break and does not need any further consideration.

For a single-line query, it is unclear where the `DISPLAY` clause ends. Using an `IF` clause would be interpreted as a field identifier and is necessary should a field IF be required. To resolve this situation, an `IFF` clause should be used. This signals an end of the `DISPLAY` clause and allows for correct parsing of the following `IF` clause.

23 ADAULD (File Unloading)

- Functional Overview 368
- Procedure Flow 370
- Checkpoints 372
- Control Parameters 373
- Examples 378
- TEMP Data Set Space Estimation 380
- Restart Considerations 380

This chapter describes the utility "ADAULD".

Functional Overview

The utility ADAULD unloads an Adabas file, i.e. records are retrieved from a database or an Adabas backup copy, and written to a sequential file.

The main reasons for unloading a file are:

- To change the space allocation, to reduce the number of logical extents assigned to the index, Address Converter or Data Storage, and/or to change the padding factor. In this case, the file has to be unloaded, deleted and reloaded. These features are also available with ADAORD;
- To create one or more test files, all containing the same data. This procedure requires a file to be unloaded and then reloaded using a different file number. This feature is also available with ADAORD;
- To extract data from a file for subsequent input to ADAMUP. This is useful for moving records from a production database to an archive database;
- To re-establish a file that has been archived on an Adabas backup copy.

When unloading a file from a database, the records may be unloaded in:

Logical sequence
The records are unloaded in an ascending sequence based on the values of a user-specified descriptor;

ISN sequence
The records are unloaded in ascending ISN sequence;

Physical sequence
The records are unloaded in the order in which they are physically located in Data Storage.

Unloading in logical or ISN sequence requires the nucleus to be active. The nucleus is not required when unloading in physical sequence, provided ADAULD has access to the database container files.

When unloading from an Adabas backup copy, the records are unloaded in the sequence in which they were stored by ADABCK. This is generally in ascending data RABN sequence. However, this sequence cannot be guaranteed when the DRIVES option was used or when the dump was made online (please refer to the DRIVES option of the utility ADABCK for more detailed information).

The unloaded records are output in compressed format and are identical to the records produced by the compression utility ADACMP. Since each data record is preceded by its ISN, these ISNs can be used as user ISNs when reloading the file (please refer to the USERISN option of the utility ADAMUP for more detailed information).

The user can specify that the descriptor values required to recreate the index for the file are omitted during the UNLOAD process (SHORT option). This reduces the unload processing time. This option must not be used if the output is intended as direct input for ADAMUP.

Note: If the file contains collation descriptors, the ICU version is not changed for the unloaded data. You can load the data with ADAMUP to a file with the same fields, but a different ICU version only if the file is empty, and if you use the NEW_FDT option for ADAMUP.

After completion, ADAULD returns one of the following exit status values:

0
Records have been successfully unloaded, and no database corruption was detected.

12
The unload was successful, but corrupted data records were detected, which were not unloaded. It is recommended that you run ADAVFY in order to obtain more information about the database corruptions.

15
The unload was successful, but no records were unloaded. In scripts, you can check for this status value if further activities are required only after unloading at least one record.

255
Unload was not successful.

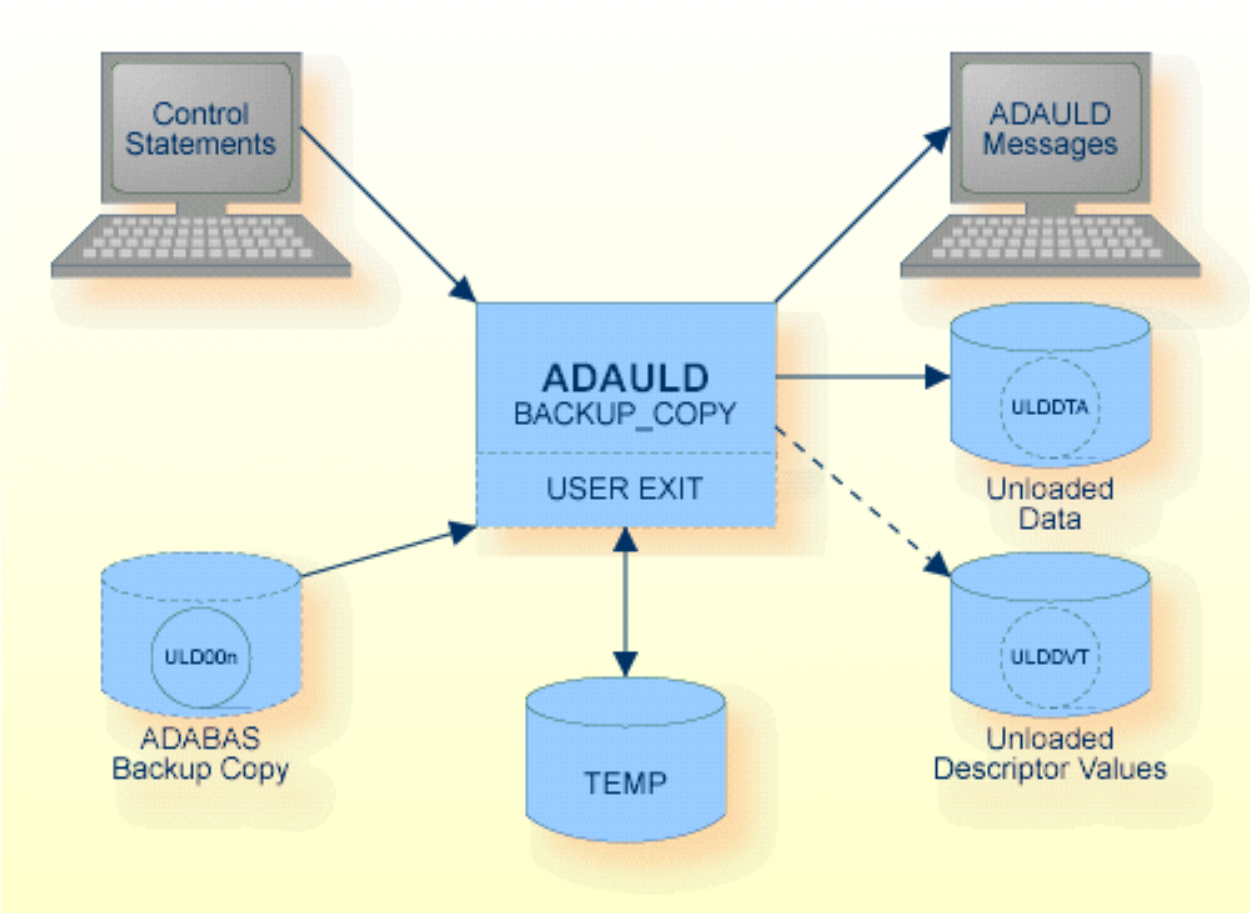
This utility is a single-function utility. For more information about single- and multi-function utilities, see *Adabas Basics*, *Using Utilities* in the Adabas documentation.

Encryption:

When using ADAULD on encrypted data, the corresponding encryption context is derived in the following way:

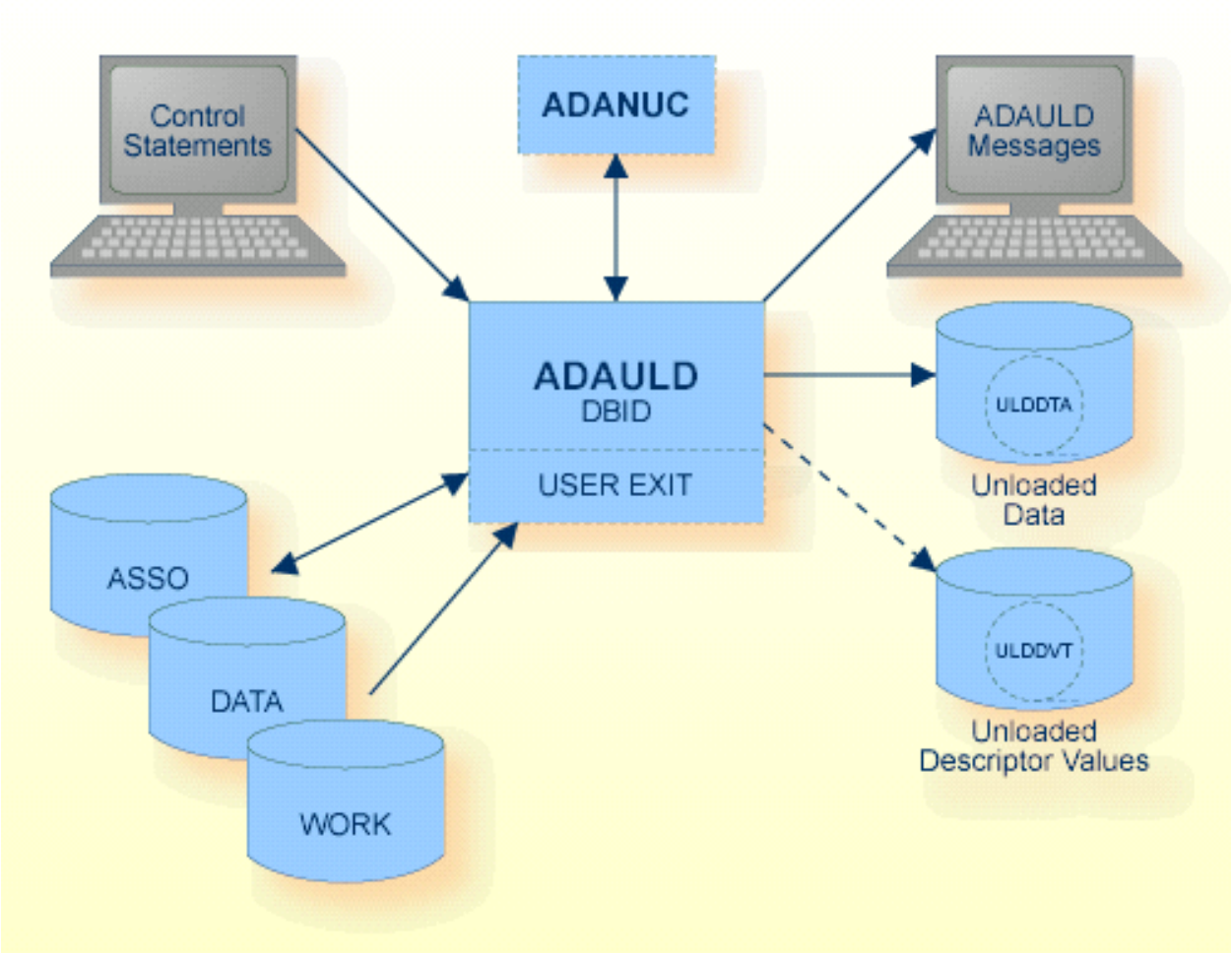
- When unloading from an encrypted database, the encryption information stored in ASSO1 is used.
- When unloading from one or multiple BCK-files, the encryption information in BCK001 is used.

Procedure Flow



BACKUP_COPY Function

The sequential files ULD00n, ULDDTA, ULDDVT can have multiple extents. For detailed information about files with multiple extents, see *Adabas Basics*, *Using Utilities* in the Adabas documentation.



DBID Function

The sequential files ULDDTA, ULDDVT can have multiple extents. For detailed information about files with multiple extents, see *Adabas Basics*, *Using Utilities* in the Adabas documentation.

Data Set	Environment Variable / Logical Name	Storage Medium	Additional Information
Associator	ASSOx	Disk	see note 2
Data storage	DATAx	Disk	see note 2
Backup copy	ULD00n	Disk	Output of ADABCK's DUMP function, input for ADAULD
Unloaded data	ULDDTA	Disk (see note 1)	
Unloaded descriptor values	ULDDVT	Disk (see note 1)	
Control statements	stdin		Utilities Manual
ADAULD messages	stdout		Messages and Codes
Temporary storage	TEMPx	Disk	see note 3 and 4

Data Set	Environment Variable / Logical Name	Storage Medium	Additional Information
Work storage	WORK1	Disk	see note 2

Notes:

1. A named pipe can be used for this sequential file.
2. Required by offline unload. Will also increase the speed of online unload using physical sequence.
3. Only required if unloading from a backup copy with the online option being used. If the utility is executed offline, WORK may be used as TEMP if there is no Autorestart pending, by setting the environment variable TEMP1 to the same value as WORK1.
4. The ADAULD BACKUP_COPY function does not read the DBxxx.INI file to find TEMP, therefore you must specify TEMP via environment variables.

Checkpoints

The following table shows the nucleus requirements for each function and the checkpoint written:

Function	Nucleus must be active	Nucleus must NOT be active	Nucleus is NOT required	Checkpoint written
BACKUP_COPY			X	-
DBID	X(see note 1)	X(see note 3)	X(see note2)	SYNX

Notes:

1. When unloading in logical sequence or ISN sequence or when the database container file cannot be accessed by ADAULD (e.g. when unloading from a remote node). Also applies if a file contains LOB data, because LOB data must be unloaded in logical sequence. Also applies if the search buffer and value buffer are provided.
2. When unloading in physical sequence and ADAULD has access to the database container files.
3. When unloading an Adabas system file.

Control Parameters

The following control parameters are available:

```
     BACKUP_COPY = number, FILE = number
                   [,FDT]
                   [,NUMREC = number]
D                  [,[NO]ONLINE]
D                  [, [NO]SHORT | [NO]SINGLE_FILE ]
                   [,SKIPREC = number]
D                  [,[NO]USEREXIT]

     DBID = number , FILE = number
            [,FDT]
D           [,[NO]LITERAL]
            [,NUMREC = number]
            [,SEARCH_BUFFER = string, VALUE_BUFFER = string]
D           [, [NO]SHORT | [NO]SINGLE_FILE  ]
            [,SKIPREC = number]
            [,SORTSEQ = { string | ISN } ]
            [,STARTISN = number]
D           [,[NO]USEREXIT]
```

BACKUP_COPY

```
BACKUP_COPY = number
              ,FILE = number
              [,FDT]
              [,NUMREC = number]
              [,[NO]ONLINE]
              [,[NO]SHORT | [NO]SINGLE_FILE ]
              [,SKIPREC = number]
              [,[NO]USEREXIT]
```

This function unloads records from an Adabas backup copy. You are not allowed to specify a LOB file. "BACKUP_COPY=number" specifies the ID of the database from which the backup copy was derived, and "FILE=number" specifies the file number. Both offline and online backup copies can be used. If a LOB file is assigned to the file specified, a partial reload using the ADAMUP parameters NUMREC, SKIPREC is not possible.

FDT

This parameter displays the FDT of the file to be unloaded.

FILE = number

This parameter specifies the file to be unloaded.

NUMREC = number

This parameter limits the number of data records retrieved from the file when unloading. All records are unloaded if NUMREC is omitted and SKIPREC is not specified. You cannot use NUMREC if a LOB file is assigned to the file to be reloaded.

[NO]ONLINE

This option indicates whether the backup copy might contain online data storage blocks for the file to be unloaded.

If the backup copy is expected to contain online data storage blocks, two passes are made when processing the backup copy. This is because the most recent version of each data storage block has to be found. Setting this option to NOONLINE unloads in one pass and saves a considerable amount of processing time, at the risk of ADAULD terminating with an error message if an online data storage block is detected.

The default used depends on whether or not the Adabas nucleus was active when the backup was made.

[NO]SHORT

This option indicates whether the descriptor values used to build up the index should be included in the output or omitted.

If SHORT is specified, no descriptor values are unloaded.

If the output is intended as direct input for the mass update utility, the file must be unloaded in NOSHORT mode.

SHORT and SINGLE_FILE are mutually exclusive.

NOSHORT is the default.

[NO]SINGLE_FILE

If this option is set to SINGLE_FILE, ADAULD writes the DVT and DATA information to a single data set (ULDDTA).

SINGLE_FILE and SHORT are mutually exclusive.

The default is NOSINGLE_FILE.

SKIPREC = number

This parameter specifies the number of records to be skipped before unloading is started. You cannot use SKIPREC if a LOB file is assigned to the file to be reloaded.

[NO]USEREXIT

A user-written routine is dynamically loaded. A pointer to an input parameter block and a pointer to an output parameter are passed with each call (please see the include file adauex.h for more information). For each record retrieved from the database, the decision can be made whether to unload the record (write it to the unload file), skip it or terminate execution immediately.

The environment variable/logical name ADAUEX_7 must point to a user-written routine.

NOUSEREXIT is the default.

DBID

```
DBID = number
       ,FILE = number
       [,FDT]
       [,[NO]LITERAL
       [,NUMREC = number]
       [,SEARCH_BUFFER = string]
       [,[NO]SHORT | [NO]SINGLE_FILE ]
       [,SKIPREC = number]
       [,SORTSEQ = { string | ISN }]
       [,STARTISN = number]
       [,[NO]USEREXIT]
       [,VALUE_BUFFER = string]
```

This function unloads records from the specified database.

FDT

This parameter displays the FDT of the file to be unloaded.

FILE = number

This parameter specifies the file to be unloaded. You are not allowed to specify a LOB file.

[NO]LITERAL

If this option is set to LITERAL, leading blanks and lower case characters can be specified in the value buffer and remain relevant in the string, i.e. they are not removed or converted to upper case. If NOLITERAL is set, lower case characters will be transformed to upper case, and leading blanks will be suppressed except when the value is specified as a hexadecimal value.

NOLITERAL is the default.

NUMREC = number

This parameter limits the number of data records retrieved from the file when unloading. All records of the file are unloaded if NUMREC is omitted and SKIPREC or STARTISN are not specified.

SEARCH_BUFFER = string

This parameter is used to restrict the unloaded records to those which meet the selection criterion provided. The selection criterion must be provided according to the syntax for search buffer entries as described in the Command Reference Manual.

The maximum length of this parameter is 200 bytes. For complex entries, use the following method:

```
adauld: search_buffer=aa,20,a,d,\
> ab,10,a.
```

ADAULD will concatenate this to:

```
aa,20,a,d,ab,10,a.
```

The values which correspond to the selection criterion are provided by the VALUE_BUFFER parameter.

[NO]SHORT

This option indicates whether the descriptor values used to build up the index should be included in the output or omitted.

If SHORT is specified, no descriptor values are unloaded.

If the output is intended as direct input for the mass update utility, the file must be unloaded in NOSHORT mode.

SHORT and SINGLE_FILE are mutually exclusive.

NOSHORT is the default.

[NO]SINGLE_FILE

If this option is set to SINGLE_FILE, ADAULD writes the DVT and DTA information to a single data set (ULDDTA).

SINGLE_FILE and SHORT are mutually exclusive.

The default is NOSINGLE_FILE.

SKIPREC = number

This parameter specifies the number of data records to be skipped before unloading is started.

When used together with the STARTISN parameter, positioning is carried out before skipping.

SORTSEQ = string

This parameter controls the sequence in which the file is unloaded. If specified, it may either contain the field name of a descriptor, sub- or superdescriptor (1) or the keyword 'ISN' (2). The default is physical sequence (3).

1. Logical sequence

   If a string specifies a field name of a descriptor or sub/superdescriptor, the records are unloaded in ascending logical sequence of the descriptor values to which the field name refers. The field name must not refer to a descriptor contained within a periodic group.

   If the field name refers to a descriptor which is a multiple-value field, the same record may be unloaded more than once (once for each different descriptor value in the record). Therefore, it is not recommended to use this type of descriptor to control the unload sequence.

   If the field name refers to a descriptor defined with the NU or NC option, the records with a null value for the descriptor are not unloaded.

2. ISN sequence

   If 'ISN' is specified, the records are unloaded in ascending ISN sequence.

3. Physical sequence

   If the SORTSEQ parameter is omitted, the records are unloaded in the physical sequence in which they are stored in the Data Storage.

   If a search buffer has been specified and the SORTSEQ parameter has been omitted, the records are unloaded in ascending ISN sequence.

STARTISN = number

If the SORTSEQ = ISN option is used or a search buffer is provided, the STARTISN parameter may be specified to start unloading at a given ISN rather than from the lowest ISN in the file. If the specified ISN does not exist, unloading starts at the next highest ISN found.

[NO]USEREXIT

NOUSEREXIT is the default.

VALUE_BUFFER = string

If a selection criterion is specified with the SEARCH_BUFFER parameter, this parameter is used to supply the values which correspond to the selection criterion. The maximum length of this parameter is 2000 bytes.

Note: See also [NO]LITERAL, which controls the conversion of the value buffer to upper case.

Examples

Example 1

```
adauld: backup_copy = 3, file = 6
```

File 6 on the backup copy of database 3 is unloaded. A TEMP data set and two passes through the backup copy may be required, depending on the default setting of the [NO]ONLINE option.

Example 2

```
adauld: backup_copy = 3, file = 6
adauld: single, noonline
```

The same file is unloaded. Both data records and descriptor value table entries are written to the same output file. The backup copy is processed in one pass as no online blocks are expected. No TEMP data set is required.

Example 3

```
adauld: dbid = 3, file = 6, skiprec = 100
```

File 6 in database 3 is unloaded. The records are unloaded in the physical sequence in which they are stored in the Data Storage. The first 100 records found are not written to the output files.

Example 4

```
adauld: dbid = 3, file = 6
adauld: numrec = 10
adauld: sortseq = ab
adauld: short
```

Ten records from file 6 in database 3 are unloaded. The values of the descriptor AB are used to control the sequence in which the records are retrieved. The values required to re-create the inverted list when reloading are omitted.

Example 5

```
adauld: dbid = 3, file = 6, sortseq = isn, startisn = 123
```

File 6 in database 3 is unloaded. The records are unloaded in ascending ISN sequence starting at ISN 123.

TEMP Data Set Space Estimation

When unloading from an Adabas backup copy without the NOONLINE option set, the TEMP data set is required to accumulate information about online block occurrences.

The formula TRH=DRH/1000 can be used as a rough estimate with the default TEMP block size (4 kilobytes).

The following formula may be used to calculate the exact requirements:

```
X = ENTIRE ((DRH / BSTD) * 4)

TRH = X + ENTIRE (X / BSTD / 8) + 1
```

where:

ENTIRE
the next highest integer

BSTD
TEMP block size in bytes.

DRH
highest Data Storage RABN in the database on the backup copy. The SUMMARY function of the ADABCK utility can be used to obtain this number.

TRH
highest RABN required on TEMP.

Restart Considerations

ADAULD has no restart capability. An interrupted ADAULD run must be re-executed from the beginning.

24 ADAVFY (Database Consistency Check)

- Functional Overview 382
- Procedure Flow 383
- Checkpoints 384
- Control Parameters 384
- Examples 388

The following topics are covered:

Functional Overview

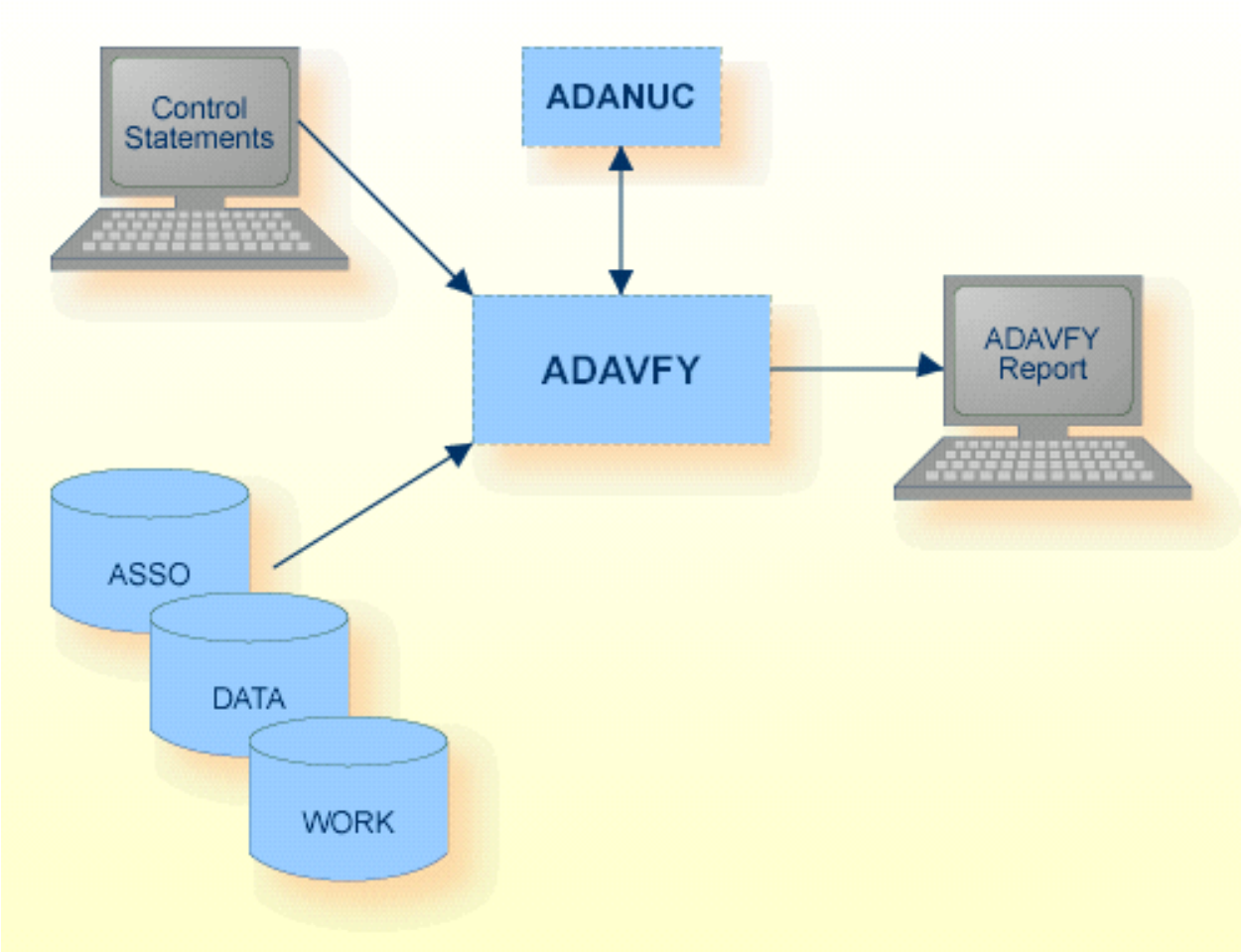
The ADAVFY utility checks the consistency of the database. The General Control Block (GCB) is validated together with each File Control Block (FCB) and each Field Definition Table (FDT) of the loaded files. The index structure and Data Storage are validated. ADAVFY can also search for lost RABNs.

Running ADAVFY against an active nucleus, or running in parallel with utilities that perform database updates, may result in errors being reported. This is because further updates can be made before the utility terminates and some of these updates are only reflected in the nucleus buffer pool. ADAVFY does not require the Adabas nucleus to be active; it processes the database offline.

In general, ADAVFY only displays consistency errors that it detects and it does not modify the database. However, there are some errors in FCB and FDT that will be corrected by ADAVFY in offline mode when ADAVFY finds them, for example, an invalid value of the record counter in the FCB for the number of records in the file.

This utility is a multi-function utility. For more information about single- and multi-function utilities, see *Adabas Basics*, *Using Utilities* in the Adabas documentation.

Procedure Flow



Data Set	Environment Variable / Logical Name	Storage Medium	Additional Information
Associator	ASSOx	Disk	
Data storage	DATAx	Disk	
Control statements	stdin		Utilities Manual
ADAVFY messages	stdout		Messages and Codes
Work	WORK1	Disk	

Checkpoints

The utility writes no checkpoints.

Control Parameters

The following control parameters are available:

```
     AC

     DATA

M    DBID = number

D    ERRORS = number

     FCB

     FIELD

D    FILES = { * | (number [-number][,number[-number]]...)  }

     FROM = number - number

     INDEX

D    LEVEL = number

     LOB_REFERENCES

     LOST

     RECORD
```

The parameters AC, DATA, FCB, FIELD, INDEX, LOST and RECORD immediately invoke the corresponding verification function. The remaining parameters are only evaluated if they have been specified before such a parameter.

AC

```
AC
```

This function validates from the Address Converter to the Data Storage and checks that records can be found in the specified Data Storage for the files specified with the FILES parameter (see also DATA).

DATA

```
DATA
```

This function verifies Data Storage for the specified file number(s). This function validates from the Address Converter to the Data Storage and from the Data Storage to the Address Converter for the files specified with the FILES parameter. The ADAVFY DATA function corrects the following error in offline mode if it is detected: the FCB contains a record counter for the number of records in the file, and if this counter has an incorrect value, it will be corrected.

DBID

```
DBID = number
```

This parameter selects the database to be verified.

ERRORS

```
ERRORS = number
```

This parameter specifies the number of errors to be reported before the verification of a single file terminates. The minimum number allowed is 1. The default value is 20.

FCB

```
FCB
```

This function validates the file control block together with the Field Definition Table for the files specified with the FILES parameter (see also INDEX).

FIELD

```
FIELD
```

This function validates the Data Storage. It checks the record structure and validates the contents of unpacked, packed and floating point values for the specified files.

FILES

```
FILES = { * | (number[-number][,number[-number]]...)  }
```

This parameter specifies the files to be verified. If an asterisk '*' is entered, all files will be verified. The FILES parameter is required for all functions except the LOST function.

The default is no files.

FROM

```
FROM = number - number
```

The values specified are used in conjunction with the LEVEL option to print various structures. Please refer to the LEVEL parameter in this section for more detailed information.

INDEX

```
INDEX
```

This function verifies the complete index to level 1 (Normal Index). This includes verification of the FCB and FDT.

ADAVFY also counts the number of used, free, reusable and lost NI (Normal Index, index level 1), MI (Main Index, index level 2) and UI (Upper Index, index level 3 or greater) blocks.

Example:

```
2024-12-13 05:37:17 %ADAVFY-I-INDSTR, Index verification
2024-12-13 05:37:17 %ADAVFY-I-INDCNT, NI: used: 26, free: 64, reusage: 0, lost: 0
2024-12-13 05:37:17 %ADAVFY-I-INDCNT, UI: used: 1, free: 2, reusage: 0, lost: 0
2024-12-13 05:37:17 %ADAVFY-I-INDCNT, MI: used: 12, free: 2, reusage: 0, lost: 0
2024-12-13 05:37:17 %ADAVFY-I-INDEND, Index verification completed
```

Notes:

1. Used index blocks are index blocks that are currently in use.
2. Free index blocks are index blocks that have not yet been used.
3. Reusable index blocks are index blocks that already have been used, but that have become empty again and were included in the reusage queue. These blocks can be used again.

4. Lost index blocks are index blocks that are not currently used and that are missing in the reusage queue, and therefore cannot be used again. A value of 1 lost block is normal - this can happen after running ADAINV REINVERT.
5. The number of free, reusable and lost MI and UI blocks is the same, because these blocks are taken from the same logical extent. Please note that the numbers displayed are the numbers for MI and UI together - if you use additional space for MI blocks, this also reduces the number of space available for UI blocks.

LEVEL

```
LEVEL = number
```

This parameter specifies how much information ADAVFY should output concerning the internal structures. Specifying this parameter does not affect the degree of verification performed. If this parameter is used, it must be specified before the function in question.

The default value is the highest possible index level plus 1.

with INDEX function

Level n	prints information about index level n and higher
Level 0	prints more detailed structure of the index blocks

The FROM option is used to specify an index RABN range. Only the RABNs specified will be dumped.

with AC/DATA/RECORD/FIELD functions

Level 2	prints which RABNs processed
Level 1	prints record structure (when RECORD or FIELD is used), or where each ISN points (when DATA or AC is used), or which ISN with LOB ID is verified (when LOB_REFERENCES is used)
Level 0	dumps fields within records

with LOB_REFERENCES function

Level 0	dumps the ISN with LOB ID list found from the base file, and ISN with LOB record from the LOB file

with LOST function

Level 0	dumps the physical structure of the database

LOB_REFERENCES

This function verifies LOB references between the LOB file and the base file.

LOST

```
LOST
```

If this option is specified, ADAVFY searches for lost RABNs in the database. If any lost RABNs are found, the space can be recovered by using the RECOVER function of ADADBM.

RECORD

```
RECORD
```

This function validates the Data Storage and checks the structure of each record for the specified files (see also FIELD).

Examples

Example 1

```
adavfy: dbid=3,file=*,data,field,index
```

All files of database 3 are validated using the functions DATA, FIELD and INDEX. This combination of functions gives the maximum degree of validation.

Example 2

```
adavfy: dbid=3, file=7, level=1, field
```

File 7 of database 3 is validated. The record structure in Data Storage is validated, as well as the contents of unpacked, packed and floating point fields. ADAVFY prints a list of the RABNs which have been processed and, for each record processed, its offset in the corresponding RABN, its length and its ISN.