Adabas for Linux and Cloud

Adabas Extended Operation

Version 7.3.0

February 2025

ADABAS & NATURAL

This document applies to Adabas for Linux and Cloud Version 7.3.0 and all subsequent releases.

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

Document ID: ADAOS-EXOP-730-20250228

Table of Contents

Preface

This document describes Adabas Extended Operation (AEO), a mode of operation that greatly simplifies the day-to-day administration of Adabas.

Adabas Extended Operation is intended for experienced DBAs. By using Adabas Extended Operation, many situations which would normally require the intervention of the DBA can be detected and resolved automatically.

This document is organized as follows:

- *Introduction*
- *Configuration Files*
- *Action Templates*
- *Administration Commands*

1 About this Documentation

Document Conventions

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

Online Information and Support

Product Documentation

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

Product Training

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

Tech Community

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

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

A typical example of when to use AEO is when additional disk space has to be allocated because a database is almost full. Without AEO, the DBA must constantly monitor how much disk space is available, and be prepared to allocate more disk space for the database when a certain critical limit is reached. With AEO, the DBA can let the monitoring and the allocation of additional disk space be done automatically. The DBA can specify a default action to be taken when a certain limit is reached (for example, allocate more disk space when the database is 90 percent full), and AEO will automatically allocate the required additional disk space when the specified limit is reached.

This is just one example of the many routine database administration tasks which can be assigned to AEO. As well as freeing the DBA from many such administration tasks, AEO makes the database environment more reliable by ensuring that human error or oversight is reduced to a minimum, and that necessary administration actions are carried out exactly when required.

AEO can be used in the following areas:

- troubleshooting
- logging of response codes
- creating automatic database backups
- performance analysis
- management of database configuration

How Adabas Extended Operation works

A standard Adabas environment is composed of the following components:

- the Adabas nucleus
- the Adabas utilities
- the database container files
- the nucleus startup utility, called Adabas (PC platforms only)
- the database log file adanuc.log
- sequential database files, for example NUCPLG, NUCCLG

AEO extends this environment by adding the following components:

- an event analyser
- a log filter
- an action filter
- a configuration file for each Adabas database

- a configuration file common to all Adabas databases
- a set of administration commands, e.g. for starting and stopping databases

In the AEO environment, the database log file is no longer important, since its contents are also written to the common log file ADABAS.LOG (see below).

The Event Analyser

The event analyser (also called the AEO analyser or simply the analyser) is the component of AEO which receives event messages generated by the Adabas nucleus or Adabas utilities and passes them on to the *log filter* or the *action filter*, depending on the nature of the event.

Database *events* such as creating files and records are reported to the event analyser. Events are passed to the log filter if they are simply status messages which do not require any immediate action to be carried out.

Events are passed to the action filter if they indicate that a change has taken place in the database which requires some immediate database administration action.

The Log Filter

The log filter receives events from the event analyser and writes them to the log file ADABAS.LOG, where they can be analysed later.

Adabas Configuration File (ADABAS.INI)

In order for AEO to work in an Adabas environment, a single configuration file ADABAS.INI must be available. This file contains global information used by AEO, whenever any database is started, in order to set up parameters for AEO operation during the database session. If there is a DBnnn.INI file (see below) for a particular database, the information contained there overrides the information specified in ADABAS.INI. The administration of ADABAS.INI and the description of its contents is described in ***Configuration Files, ADABAS.INI***.

Database Configuration Files (DBnnn.INI)

In order for AEO to work with a database DBnnn in an Adabas environment, where nnn is the database number, a corresponding configuration file DBnnn.INI must be available. This file contains information used by AEO when the database is started, in order to set up parameters for AEO operation during the database session. The information specified in these files overrides any information specified in the Adabas configuration file ADABAS.INI. The administration of the DBnnn.INI files and the description of their contents is described in ***Configuration Files, DBnnn.INI***.

Actions and the Action Filter

Most of the topics and subtopics within the configuration files ADABAS.INI and DBnnn.INI start an associated *action*. Such actions are provided in the form of templates. In general, an action is started if the appropriate event occurs. Such an action could be, for example, to call one of the Adabas utilities automatically to perform necessary reorganization of the database. Another action could be to output a warning message to the DBA's terminal to indicate a situation that requires DBA intervention. An event is initiated either by the Adabas nucleus or by an Adabas utility. Normally, these actions are started under the control of AEO. The exceptions to this rule are the two actions ARCHIVE_LOGFILE and SAVE_DB.

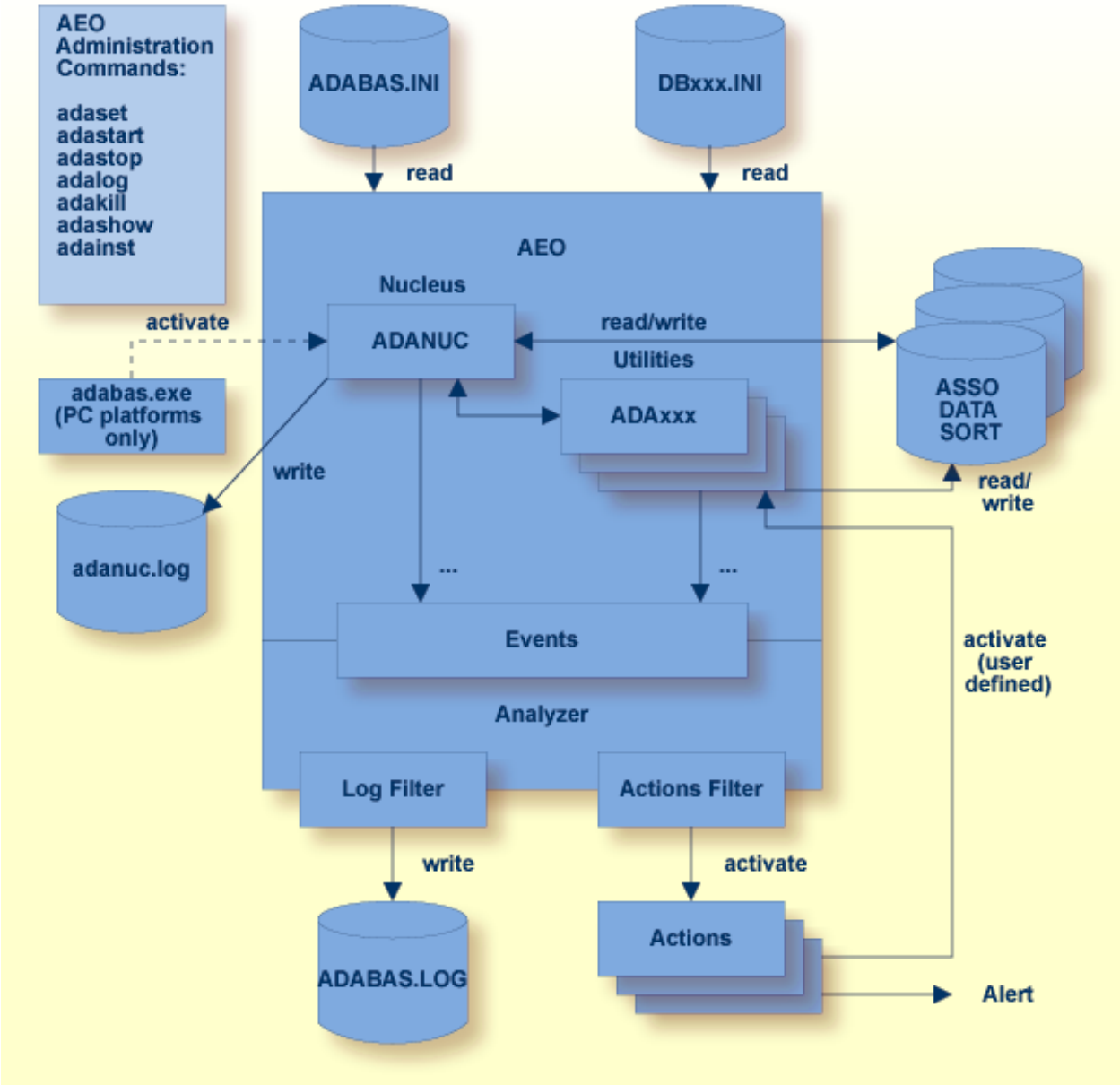
The action filter receives events from the event analyser and, depending on the nature of the event, causes the appropriate predefined action to be activated.

Administration Commands

AEO provides a set of commands which simplify the administration of a database. The following commands are available:

adaini	to retrieve, modify and verify information stored in the configuration files
adainst	to create and update the file ADABAS.INI
adakill	to stop a database immediately
adalog	to display the Adabas log file
adaset	to set a default database number required by the other administration scripts
adashow	to display general information about a database
adastart	to start a database
adastop	to stop a database in an orderly manner

Important: The item ACTION in the topic NODE_PARAMETER, subtopic ANALYSER must be set to 'yes' in order to activate AEO. Please refer to the administration command **adaini** in the section *Administration Commands* for further information.



Adabas Extended Operation environment

3 Configuration Files

Configuration parameters are required for each database in an Adabas Extended Operation (AEO) environment. These configuration parameters tell AEO about special criteria which have been defined for the database by the DBA, and therefore about how AEO should react when certain database events occur.

The configuration parameters are stored in configuration files. One configuration file, named ADABAS.INI, contains configuration parameters that apply to all databases in the Adabas environment on that node. In addition, for each database, there must be a file DBnnn.INI, where nnn is the database number.

The files ADABAS.INI and DBnnn.INI contain definitions such as container assignments, nucleus parameters and control parameters for AEO. They are ASCII files, so they can be edited with a standard text editor. However, a text editor does not guarantee that the configuration files are syntactically correct. In order to maintain the AEO-specific entries in the configuration files, we recommend that you use the administration command adaini.

Format Of Configuration Files

Each configuration file is divided into topics, with the following syntax:

```
[topic name]                                    # start of topic "topic name"
  item1 name = item1 value                      # name and value of item 1
  item2 name = item2 value                      # name and value of item 2
  ....
[topic name-END]                                # end of topic
```

Each topic contains item definitions and/or subtopics. There can be several nesting levels, but usually there are not more than 2 levels, i.e. the main topic level and one subtopic level.

Topic names, item names and values are case sensitive. Comment lines start with the character "#".

Empty lines are ignored. White space, i.e. blanks and tabs, surrounding topic names, item names and values are also ignored. If a value contains white space, it must be enclosed by single or double quote delimiters. Delimiters that occur within a value must be typed twice.

Note: You cannot use environment variables as values within the configuration file. They must be substituted by their explicit values.

The following is an example of a configuration file with two nesting levels and one comment line:

```
[topic1 name]                                  # start of topic on nesting level 1
item1 name = item1 value                         # name and value of item 1

# Start of topic2                                # comment line
  [topic2 name]                                # start of topic on nesting level 2
    item2 name = item2 value                     # name and value of item 2
  [topic2 name-END]                              # end of topic 2

[topic1 name-END]                                # end of topic 1
```

ADABAS.INI

The configuration file ADABAS.INI contains information that applies to all databases in an AEO environment. If there is a file DBnnn.INI for a particular database, the information in that file overrides the information in the ADABAS.INI file. ADABAS.INI can be modified with a standard text editor or with the administration command adaini. This can also verify the topic NUCPARMS. See the adaini online help for the utility syntax.

ADABAS.INI contains the following information:

1. Database-independent information, e.g. default nucleus parameters and environment information for new databases being created. Also general information such as the location of ADABAS.LOG (log file for all databases on the local node), what to do if a report utility (adarep, adafin, adapri, adaclp, adaplp, adaerr, adatst) or display function (adaopr/adadbm display=...) is run.
2. Basic database definitions, e.g. location of the individual DBnnn.INI files and the database ID.

Location of ADABAS.INI

On Linux and Windows platforms, ADABAS.INI must be in the subdirectory *etc* of the directory that is pointed to by the environment variable *ADADATADIR*. If there is no ADABAS.INI file in this directory, AEO is automatically disabled and only the standard Adabas functionality is available.

Structure of ADABAS.INI

ADABAS.INI is divided into sections, with one topic per section. Each section of ADABAS.INI starts with a line containing the name of the topic enclosed in square brackets, using the syntax *[topic-name]*. The topics relevant to AEO are:

- DB_DEFAULTS (with various subtopics)
- DB_LIST (with one subtopic per DBID)
- MISCELLANEOUS
- NODE_PARAMETER, with the subtopics

- ALERT
- ANALYSER
- ARCHIVE_LOGFILE
- LOGGING

Important: The item ACTION in the topic NODE_PARAMETER, subtopic ANALYSER must be set to 'yes' in order to activate AEO. Please refer to the administration command **adaini** in the section *Administration Commands* for further information.

These topics are described in the following sections.

Topic: DB_DEFAULTS

The topic DB_DEFAULTS contains default definitions used by ADAFRM or by adainst to create DBnnn.INI. Its contents are copied to DBnnn.INI without the enclosing lines [DB_DEFAULTS] and [DB_DEFAULTS-END]. For further information, refer to the section DBnnn.INI later in this section.

```
# configuration                                  # recommended values

[DB_DEFAULTS]
 ...
 [NUCPARMS]
   ...
 [NUCPARMS-END]
   ...
 [DB_PARAMETER]
   ...
 [DB_PARAMETER-END]
   ...
 [ENVIRONMENT]
   ...
 [ENVIRONMENT-END]
 ...
[DB_DEFAULTS-END]
```

NUCPARMS

The items in the topic NUCPARMS can be verified by the administration command *adaini* with the VER parameter. Potentially unique abbreviations will then be automatically replaced by the correct item names (e.g. the abbreviation LOG will be automatically corrected to LOGGING).

ENVIRONMENT

In this topic, default environment variables can be defined. If the variable is set in the shell environment, the nucleus will take the value, otherwise the setting of the ENVIRONMENT topic will be used.

Topic: DB_LIST

The topic DB_LIST lists the numbers and names of the available databases, as well as the name and location of the associated DBnnn.INI file. The details for each database are given in a separate subtopic DBID_*<dbid>* within the topic DB_LIST.

```
# configuration                                    # recommended values

[DB_LIST]

  [DBID_<dbid>]                                    PC platforms:
    INI_FILE = <configuration file name           %ADADATADIR%\db<dbid>\DB<dbid>.INI
                for this database>                  Linux:
                                                    $ADADATADIR/db<dbid>/DB<dbid>.INI

    NAME = <database name>
  [DBID_<dbid>-END]

[DB_LIST-END]
```


Note: You cannot use environment variables as parameters in the configuration file. You must therefore substitute the ADADATADIR environment variable by its full expanded name in the INI_FILE item.

Topic: MISCELLANEOUS

The topic MISCELLANEOUS contains database-independent information for this node. The item NODE_NAME is used for the logging that is activated by the subtopic LOGGING of the topic NODE_PARAMETERS (see later in this section).

On Linux platforms you can use the Linux command "uname -n" to get the system node name.

The syntax for the topic MISCELLANEOUS is as follows:

```
# configuration                                    # recommended values

[MISCELLANEOUS]
  NODE_NAME = <node name>
[MISCELLANEOUS-END]
```

Topic: NODE_PARAMETER

The topic NODE_PARAMETER contains AEO definitions for all databases on this node.

Subtopic: ALERT

```
# configuration                                    # recommended values

[NODE_PARAMETER]
  [ALERT]
    ACTION = <yes/no>
    ACTION_ROUTINE = <action routine>           ada_alrt
    MINIMUM = <minimum severity required        E
                  to call the action>
  [ALERT-END]

[NODE_PARAMETER-END]
```

The recommended action routine is ada_alrt, which has the following parameters:

Parameter	Description
1	utility name
2	process ID
3	DBID
4	user login name
5	node name
6	severity
7	message header key
8	message header
9	message text

The ALERT action is used to inform the DBA (and possibly other users) when an Adabas event occurs, for example, Adabas could "mail" all messages with severity E (error) and F (fatal) automatically to the DBA.

The subtopic ALERT of the topic NODE_PARAMETER defines the Adabas alert action. If the ALERT action is enabled (ACTION=yes), the severity of each Adabas message is checked, and if it is greater than or equal to the defined MINIMUM severity, the action routine specified by AC-

TION_ROUTINE is started asynchronously. The severity priority is: I (information) < W (warning) < E (error) < F (fatal). Thus, if MINIMUM=E, only messages with severity E and F will call the ALERT action.

The list of recipients of the alert is determined by the contents of the environment variable ADA_ALRT_LIST, which is a concatenation of one or more user specifications that are separated by blanks. The user specifications depend on the platform. On Windows, it is the name of a computer, e.g. pcABC1 if the machine is in the same domain or /DOMAIN:pcABC1 if the computer is in another domain. On Linux, it is the user name or an e-mail address.

Example

```
set ADA_ALRT_LIST ="pcABC1 /DOM-HQ:pcABC2"   (for Windows)

setenv ADA_ALRT_LIST "abc user.xyz@myCompany.com"   (for C shell)
```

Subtopic: ANALYSER

The subtopic ANALYSER of the topic NODE_PARAMETER enables/disables the AEO analyser. If the analyser is disabled, all Adabas actions, including LOGGING and ALERT, are automatically disabled.

Note: This is the global switch which enables/disables AEO.

```
# configuration                                    # recommended values

[NODE_PARAMETER]

  [ANALYSER]
    ACTION = <yes/no>
  [ANALYSER-END]

[NODE_PARAMETER-END]
```

Caution: If "ACTION=no" is set, all other AEO settings will be ignored, and AEO will be disabled.

Subtopic: ARCHIVE_LOGFILE

```
# configuration                                    # recommended values

[NODE_PARAMETER]

  [ARCHIVE_LOGFILE]
    ACTION = <yes/no>
    ACTION_ROUTINE = <action routine>              ada_svlg
    MAXIMUM = <maximum number of                   7
              archive generations>
  [ARCHIVE_LOGFILE-END]

[NODE_PARAMETER-END]
```

The recommended action routine is ada_svlg, which has no parameters.

Each logging message is appended to the end of the Adabas log file. To prevent uncontrolled growth of this file, you can submit the command specified by the item ACTION_ROUTINE with an external scheduler every day. This routine reads the item ACTION and exits without doing anything if ACTION=no. If ACTION=yes and MAXIMUM=0, the log file is deleted without archiving. If ACTION=yes and MAXIMUM is greater than 0 and less than 100, MAXIMUM number of generations of log files (one generation per execution of ada_svlg) are archived. This is done by appending a generation number in the range 1 ... MAXIMUM for PC platforms or 01 ... MAXIMUM for Linux to the log file name, for example, ADABAS.LOG.1 for PC platforms or ADABAS.LOG.01 for Linux. Log files with a generation equal to MAXIMUM are erased during the next run of ada_svlg.

Note: ada_svlg cannot run while the log file is in use by other tools, for example, adalog -t.

Subtopic: LOGGING

```
# configuration                                    # recommended values

[NODE_PARAMETER]

  [LOGGING]
    ACTION = <yes/no>
    FILTER_MESSAGES_WITHOUT_HEADER = <yes/no>
    FILTER_REPORT_UTILITIES = <yes/no>             PC platforms:
    LOG_FILE = <log file name>                     %ADADATADIR%\etc\ADABAS.LOG
                                                   Linux:
                                                   $ADADATADIR/etc/ADABAS.LOG

  [LOGGING-END]

[NODE_PARAMETER-END]
```

Note: You cannot use environment variables as parameters in the configuration file. You must therefore substitute the ADADATADIR environment variable by an explicit path name in the INI_FILE item.

The subtopic LOGGING of the topic NODE_PARAMETER defines parameters for the logging of Adabas messages. Adabas messages (normally written to standard output exclusively) are analyzed and appended to the AEO log file if the associated logging filter conditions are true. Additionally, AEO logs new Adabas messages (that are not written to standard output), which contain special event and action information.

The name of the log file is given by the item LOG_FILE.

The item ACTION enables/disables AEO logging for all databases on this node. If logging is enabled, each Adabas utility appends its logging information to the sequential file defined with item LOG_FILE. The writing of logging messages is synchronized using an Adabas semaphore. The logging filter recognizes two conditions:

Messages without a header are ignored (i.e. not logged), if:
FILTER_MESSAGES_WITHOUT_HEADER = yes

Messages from Adabas report utilities are ignored (i.e. not logged), if:
FILTER_REPORT_UTILITIES = yes

The Adabas report utilities are: adaclp, adafin, adaplp, adapri, adarep, adatst.

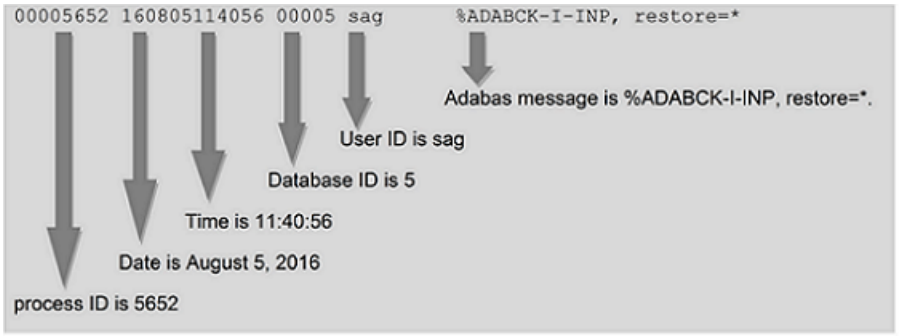
Each entry in the log file has the following format:

```
<process id> <yymmddhhmmss> <dbid> <user ID> <Adabas message>
```

where y`ymmdd` and h`hmmss` are the current date and time, respectively.

The items in the log file entry are separated by blanks.

Example



Note: If an Adabas log message contains 0 as database ID, the database ID has not yet been determined when the message was created.

DBnnn.INI

For each database that is to be administered by AEO, there must be a DBnnn.INI file, where nnn is the database number. The DBnnn.INI file contains information that applies to the database number nnn, and overrides the information in the ADABAS.INI file.

If there is no DBnnn.INI file for a database, AEO does not work with the database, so only the standard Adabas functionality can be used. Also, there must be an INI_FILE entry in the DB_LIST section of ADABAS.INI for the database nnn.

DBnnn.INI contains the following:

1. Database information, e.g. container assignments, nucleus parameters
2. User definitions which describe the permissions to run actions

Location of DBnnn.INI

The location of the DBnnn.INI files is defined by the item INI_FILE in the DB_LIST topic of ADABAS.INI.

When creating a new database with the utility ADAFRM, a DBnnn.INI file is generated. The utility adaini can be used to modify the contents of DBnnn.INI and also to verify the topic NUCPARMS.

Structure of DBnnn.INI

DBnnn.INI is divided into sections, with one topic per section. Each section starts with a line containing the name of the topic enclosed in square brackets, using the syntax *[topic-name]*. The available topics are:

- CONTAINER
- DB_PARAMETER, with the subtopics
  - ACTION_DBA
  - ADANUC_STARTED
  - ADANUC_TERMINATED
  - AUDIT_TRAIL
  - CLUSTER_EVENT (applies only to Adabas Cluster)
  - DELETE_CHECKPOINTS
  - INCREASE_<container>
  - OFFLINE_CHECKPOINTS
  - RECOVER_LOST_BLOCKS
  - REORDER_FILE
  - SAVE_DB
  - SSX_CONFIGURATION
  - TERMINATE_ADANUC
- ENVIRONMENT
- NUCPARMS
- RESERVED_LOCATION
- TEMPORARY_LOCATION

Topic: CONTAINER

```
# configuration                                  # recommended values

[CONTAINER]

  ASSOx = <ASSOx device>                         # n lines for ASSO1 ... ASSOn
  DATAx = <DATAx device>                         # n lines for DATA1 ... DATAn
  WORK1 = <WORK1 device>                         # 1 line  for WORK1
  SORTx = <SORTx device>                         # n lines for SORT1 ... SORTn
  TEMPx = <TEMPx device>                         # n lines for TEMP1 ... TEMPn
  NUCCLGx = <NUCCLG device>
  NUCPLGx = <NUCPLG device>
```

```
[CONTAINER-END]
```

The Adabas containers ASSOx, DATAx, WORK1, SORTx and TEMPx are defined in this topic. Additionally, the Adabas sequential files NUCCLGx and NUCPLGx are defined in this topic.

If explicit external environment definitions for these items exist, their values override the DBnnn.INI values. Therefore, you can use the items in DBnnn.INI to define default values.

Important: If you enable the INCREASE_<container> actions in the DB_PARAMETER topic, AEO must be informed of the disk location to use if a new container is required. You do this by appending at least one *extra* DATAx and ASSOx to your lists, and these define the location that AEO will use if the INCREASE_<container> action is started.

When adding or removing a container with the utility ADADBM, the container definition in the DBnnn.INI file is updated.

Topic: DB_PARAMETER

The topic DB_PARAMETER defines actions and other definitions for AEO.

Subtopic: ACTION_DBA

The topic ACTION_DBA contains the names of all users for whom AEO will start actions if required. Adabas and all utilities will also work for other users, but all actions defined in the topic DB_PARAMETER are only initiated for users defined as ACTION_DBA.

Linux: The user name is case-sensitive.

PC platforms: If no user is defined and AEO is enabled, the respective actions will be initiated for any user.

```
# configuration                                  # recommended values

[DB_PARAMETER]

  [ACTION_DBA]
    <user name 1>                              (see platform-specific notes given ↩
above)
    <user name 2>
    ...
  [ACTION_DBA-END]

[DB_PARAMETER-END]
```

Subtopic: ADANUC_STARTED

```
# configuration                                    # recommended values

[DB_PARAMETER]

  [ADANUC_STARTED]
    ACTION=<yes/no>
    ACTION_ROUTINE=<action routine>                ada_nsta
  [ADANUC_STARTED-END]

[DB_PARAMETER-END]
```

The recommended action routine is ada_nsta, which has the following input parameters:

Parameter	Description
1	utility name = ADANUC
2	process ID
3	DBID
4	DB status = ONLINE
5	user login name
6	node name
7	session number

The action routine is submitted by the Adabas utility ADANUC only if the nucleus started successfully.

The ACTION_ROUTINE will only be submitted if ACTION=yes.

Subtopic: ADANUC_TERMINATED

```
# configuration                                    # recommended values

[DB_PARAMETER]

  [ADANUC_TERMINATED]
    ACTION=<yes/no>
    ACTION_ROUTINE=<action routine>                ada_nsto
  [ADANUC_TERMINATED-END]

[DB_PARAMETER-END]
```

The recommended action routine is ada_nsto, which has the following input parameters:

Parameter	Description
1	utility name = ADANUC
2	process ID
3	DBID
4	DB status = ONLINE
5	user login name
6	node name
7	termination status (= TERMINATED or ABORTED)

The action routine is submitted by the Adabas utility ADANUC when the nucleus terminates.

The ACTION_ROUTINE will only be submitted if ACTION=yes.

Subtopic: AUDIT_TRAIL

```
# configuration                                    # recommended values

[DB_PARAMETER]

  [AUDIT_TRAIL]
    ACTION=<yes/no>
    FILTER=<all/rejected>
    MAXFILESIZE=<max file size to switch to next sequential audit file>
  [AUDIT_TRAIL-END]

[DB_PARAMETER-END]
```

To write the audit trail set ACTION to YES. The default value is NO.

To only log rejected authentications and authorizations for Adabas direct call interface, set FILTER to REJECTED. The default value is ALL.

Subtopic: CLUSTER_EVENT

CLUSTER_EVENT is only available in a cluster setup.

When CLUSTER_EVENT is set, the entries under the topic CLUSTER_EVENT in the DBnnn.INI file are used in the case of a lost quorum or if the primary node changes.

```
[DB_PARAMETER]
   [CLUSTER_EVENT]
       ACTION = <YES/NO>
       ACTION_ROUTINE = <action routine>   ada_clualrt
       LOST_QUORUM = <YES/NO>
       PRIMARY_NODE_SWITCH = <YES/NO>
   [CLUSTER_EVENT-END]
[DB_PARAMETER-END]
```

When the nucleus detects a lost quorum or a change of primary node, the Adabas utility ADANUC submits the action routine. The ACTION_ROUTINE is only submitted if ACTION=YES and LOST_QUORUM=YES or PRIMARY_NODE_SWITCH=YES.

For more information about the cluster parameters, see *Adabas Cluster* > *From Project to Production* > *How to Set Up the Adabas Cluster* > *Environment-Specific Parameters*.

Subtopic: DELETE_CHECKPOINTS

```
# configuration                                              # recommended values

[DB_PARAMETER]

  [DELETE_CHECKPOINTS]
    ACTION=<yes/no>
    ACTION_ROUTINE=<action routine>                          ada_dlcp
    MINIMUM=<minimum age (in days) for checkpoints           100
               to be deleted>
  [DELETE_CHECKPOINTS-END]

[DB_PARAMETER-END]
```

The recommended action routine is ada_dlcp, which has the following input parameters:

Parameter	Description
1	utility name = ADANUC
2	process ID
3	DBID
4	DB status = ONLINE
5	user login name
6	node name
7	date string for ADADBM command DELCP

The action routine is submitted by the Adabas utility ADANUC when

- the nucleus starts

- the nucleus writes a checkpoint and the last time a checkpoint was written is at least one day ago (this case will occur for databases which are usually always online)

The ACTION_ROUTINE will only be submitted if ACTION=yes.

MINIMUM must be greater than 0. This value defines the minimum age (in number of days) for checkpoints to be deleted.

Subtopic: INCREASE_<container>

```
# configuration                                              # recommended values

[DB_PARAMETER]

  [INCREASE_<container>]                                     ASSO or DATA
    ACTION=<yes/no>
    ACTION_ROUTINE=<action routine>                          ada_iass or ada_idat
    MINIMUM=<minimum free space (in percent)>                15
    EXTEND_RATE=<minimum extend rate (in percent)>           10
    MESSAGE = (I=<i>,W=<w>,E=<e>,F=<f>)                      (I=40,W=20,E=10,F=5)
  [INCREASE_<container>-END]

[DB_PARAMETER-END]
```

The recommended action routines are ada_iass (for ASSO) and ada_idat (for DATA), each of which has the following input parameters:

Parameter	Description
1	utility name (upper case)
2	process ID
3	DBID
4	DB status (ONLINE or OFFLINE)
5	user login name
6	node name
7	current number of ASSO/DATA extents (e.g. 7)
8	new extent name (e.g. ASSO8)
9	extent size in MB

The action routine is submitted by the following Adabas utilities:

- ADAREP submits the action asynchronously when executing "adarep dbid=<dbid> free"
- ADANUC submits the action asynchronously after space allocation
- ADAFDU, ADABCK, ADAFRM, ADAREC, ADAORD submit the action asynchronously after space allocation when ADANUC is offline

- ADAMUP, ADAINV submit the action synchronously during space allocation when adanuc is offline

The message DBFREE is logged by adarep and by every Adabas utility which performs space allocation.

The ACTION_ROUTINE will only be submitted if ACTION=yes.

The Analyser compares the current free space (in percent) with the configured MINIMUM (valid range: 5-50 percent). If the amount of free space is less than the specified minimum value, the Analyser computes how much additional space is required and starts the action to allocate the additional space.

The calculation of additional space is done in two steps:

1. Additional space equal to the current size of the database multiplied by the value of EXTEND_RATE (valid range: 5-100 percent) is calculated. This definition ensures that the additional space is relative to the current size. The additional space calculated here is added to the value resulting from step 2.
2. It may be that even with the additional space calculated in step 1, the ratio of the free space to the total database space would be less than the percentage specified by MINIMUM. This can happen if a large block was allocated, thus causing the limit for the required minimum amount of free space to be not only reached but also greatly fallen short of. In this case, a second allocation of space is required, so that the ratio specified by MINIMUM is reached.

The amount of additional space to be allocated is therefore the result of adding the values from steps 1 and 2.

Adabas is able to increase ASSO/DATA while the nucleus is active. When the nucleus is active, space allocation (even for utilities) is done from ADANUC. The action routine takes the new container definition from the topic CONTAINER, so be sure to define one or more CONTAINER items for this action. If such a definition is missing, the action stops without increasing the size of the database.

The Analyser also compares the computed free space ratio with the defined percentage values i, w, e, f in the item MESSAGE. These values must be in the range 1-100 and in descending order, i.e. i > w > e > f. The message DBFREE is logged if the free space is equal to or smaller than one of these values.

Subtopic: OFFLINE_CHECKPOINTS

```
# configuration                                                  # recommended values

[DB_PARAMETER]

  [OFFLINE_CHECKPOINTS]
    MESSAGE = (I=<i>,W=<w>,E=<e>,F=<f>)                          (I=50,W=20,E=5,F=2)
  [OFFLINE_CHECKPOINTS-END]

[DB_PARAMETER-END]
```

The item MESSAGE defines the severity for the Adabas message CPBFREE. If Adabas is offline, the AEO Analyser is called every time an Adabas checkpoint is written into the checkpoint block. It evaluates the number of free entries in the checkpoint block and compares it with the numeric values (which must be greater than 0) of the MESSAGE parameters i, w, e and f. These values must be defined in decreasing order, i.e.: i > w > e > f. Thus, if there are 10 free entries and MESSAGE=(I=50,W=20,E=5,F=2), an Adabas message CPBFREE with severity W is written.

Note: The message CPBFREE is logged for every Adabas utility that writes checkpoints while the nucleus is offline.

Subtopic: RECOVER_LOST_BLOCKS

```
# configuration                                                  # recommended values

[DB_PARAMETER]

  [RECOVER_LOST_BLOCKS]
    ACTION=<yes/no>
    ACTION_ROUTINE=<action routine>                             ada_rlst
  [RECOVER_LOST_BLOCKS-END]

[DB_PARAMETER-END]
```

The recommended action routine is ada_rlst, which has the following input parameters:

Parameter	Description
1	utility name (upper case)
2	process ID
3	DBID
4	DB status (ONLINE or OFFLINE)
5	user login name
6	node name

The action routine is submitted by

- the Adabas utility ADANUC when Adabas return code 77 occurs and the associated command is an Adabas utility command
- the Adabas utility ADAREP: "adarep dbid=<dbid> layout"
- the Adabas utility ADAVFY: "adavfy dbid=<dbid> lost"

The ACTION_ROUTINE will only be submitted if ACTION=yes.

Lost blocks can only occur only in a utility context. If AEO was not active in this situation and lost blocks already exists, you may use ADAREP or ADAVFY to find lost blocks and to submit this action.

Note: RECOVER_LOST_BLOCKS does an implicit reset of UCB entries for the given database.

Subtopic: REORDER_FILE

```
# configuration                                                  # recommended values

[DB_PARAMETER]

  [REORDER_FILE]
    ACTION=<yes/no>
    ACTION_ROUTINE=<action routine>                              ada_reor
    MAXIMUM=<maximum number of AC/NI/UI/DS extents>              12
    MESSAGE =(I=<i>,W=<w>,E=<e>,F=<f>)                           (I=5,W=8,E=12,F=14)
  [REORDER_FILE-END]

[DB_PARAMETER-END]
```

The recommended action routine is ada_reor, which has the following input parameters:

Parameter	Description
1	utility name = ADAREP
2	process ID
3	DBID
4	DB status (ONLINE or OFFLINE)
5	user login name
6	node name)
7 - 106	Adabas file number list

The action routine is submitted within the context of the SAVE_DB action by a call of the Adabas utility ADAREP with the following parameters: adarep dbid=<dbid> content

The message FIFREE is logged by every Adabas utility which performs space allocation.

The ACTION_ROUTINE will only be submitted if ACTION=yes.

The ADAREP utility calls the AEO Analyser, which checks all database user files (Adabas system files are ignored) for their number of AC/NI/UI/DS extents. If one extent number is greater than or equal to the defined MAXIMUM (valid range: 3 - 32), this file will be reordered. The action is started once to reorder all such files (see action parameters 7 - 106). So the reorder task is performed one file after the other sequentially.

Important: This ACTION_ROUTINE is not directly submitted when Adabas allocates an extent, because at this time Adabas is still using the file, and this would result in an access conflict with the reorder action. This action may only be started after the database has been successfully saved. Therefore the action REORDER_FILE checks if the action SAVE_DB (see section Subtopic: SAVE_DB) has successfully finished. This is done by comparing the value of environment variable ADA_SAVE_DB with input parameter 3. If they match, the reorder is started. In every other context the action terminates without reordering a file.

Every time Adabas allocates file extents, the Analyser compares the current extent number with the defined message values i, w, e and f. These values must be greater than 1 and in ascending order, i.e. i < w < e < f. The Adabas operations message FIFREE is logged for any file for which one extent type (AC,NI,UI,DS) is equal to or greater than one of the defined message values i, w, e or f.

Subtopic: SAVE_DB

```
# configuration                                              # recommended values

[DB_PARAMETER]

  [SAVE_DB]
    ACTION=<yes/no>
    ACTION_ROUTINE=<action routine>                          ada_svdb
  [SAVE_DB-END]

[DB_PARAMETER-END]
```

The recommended action routine is ada_svdb, which has the following input parameter:

Parameter	Description
1	database ID = dbid

The action routine should be started with an external scheduler via ada_actn. It takes the following steps:

- checks the action definitions (ACTION must be yes)
- expands the backup file name with extension <dbid>.<nn> to

  (PC platforms:) BCK001=%ADADATADIR%\db<dbid>\BCK001.<dbid>.<nn>where <nn> is in the range 01 ... 99 and one higher than the last backup file found in that directory

(Linux:) BCK001=$ADADATADIR/db<dbid>/BCK001.<dbid>.<date>.<time>where <date> is in the form yymmdd (years, months, days) and <time> is in the form hhmm (hours, minutes).

- starts backup with the DUMP option
- (after successful backup:) sets the environment variable ADA_SAVE_DB=<dbid>
- starts "adarep dbid=<dbid> content" to check the conditions for the action REORDER_FILE.

Note: If you want to modify a backup option or the location of the backup, customize ada_svdb to your specific requirements.

Subtopic: SSX_CONFIGURATION

```
# configuration                                    # recommended values

[DB_PARAMETER]

  [SSX_CONFIGURATION]
    <option>=<value>
  [SSX_CONFIGURATION-END]

[DB_PARAMETER-END]
```

In the section *Infrastucture Security Library* of the section *Adabas Role-Based Security (ADARBA)* you can find examples and example templates for the different authorization types. These templates are not complete as some of the settings are customer-specific and must be modified where necessary.

Subtopic: TERMINATE_ADANUC

```
# configuration                                          # recommended values

[DB_PARAMETER]

  [TERMINATE_ADANUC]
    SHUTDOWN=<n1>                                          0
    CANCEL=<n2>                                            12
    ABORT=<n3>                                             12
  [TERMINATE_ADANUC-END]

[DB_PARAMETER-END]
```

The topic TERMINATE_ADANUC contains information for adastop. The item values n1, n2 and n3 must be numeric (greater than or equal to 0).

adastop executes the utility ADAOPR to stop the database. ADAOPR has 3 options to stop a database:

1. ADAOPR SHUTDOWN waits until all transactions are terminated, and then shuts down the database.
2. ADAOPR CANCEL rolls back all open transactions immediately and then shuts down the database.
3. ADAOPR ABORT stops the database without doing a clean shutdown and writes a dump.

adastop does the following:

1. If n1 > 0, adastop first performs ADAOPR SHUTDOWN and then waits for up to n1 * 5 seconds for the termination of the database. If n1 = 0, ADAOPR SHUTDOWN is not performed.
2. If the database has not yet terminated after step 1, and n2 > 0, adastop performs ADAOPR CANCEL and then waits for up to n2 * 5 seconds for the termination of the database. If n2 = 0, ADAOPR CANCEL is not performed.
3. If the database has not yet terminated after step 2, and n3 > 0, adastop performs ADAOPR ABORT and then waits for up to n3 * 5 seconds for the termination of the database. If n3 = 0, ADAOPR ABORT is not performed.

If the database has not yet terminated after step 3, adastop aborts with an error message.

Example:

With the recommended values, adastop does not perform ADAOPR SHUTDOWN, and starts directly with ADAOPR CANCEL. If the database is not down after 12 * 5 seconds = 1 minute, ADAOPR ABORT is performed. If then the database is not down after 12 * 5 seconds = 1 minute, adastop aborts with an error message.

Topic: ENVIRONMENT

You may define all Adabas environment variables in this topic (with exception of the nucleus container and SAM files). If explicit external environment definitions for these items exist, the external values override the DBnnn.INI values. Therefore, you can use these items in DBnnn.INI to define default values.

```
# configuration                       # recommended values

[ENVIRONMENT]                         # Any Adabas environment variable may
                                      # be defined here.

  ADAUEX_n = ...                      # Adabas user exit

  ADANUCLOG = ...                     # Location or path for nucleus log
                                      # upon adastart.

  ADANUCLOGOLD = <COPY|APPEND>        # selection to copy or append
                                      # the previous session of nucleus log to
                                      # <ADANUCLOG>.old
```

```
  ADATCP_DNSLOOKUP = <YES|NO>       # enables/disables a reverse
                                    # lookup for incoming adatcp
                                    # connections

  ADATRT = ....                     # file name of the shared library (Linux)
                                    # or dynamic link library (PC) containing
                                    # the Adabas translation table

  ADAHYX_n                          # Adabas hyperexit

[ENVIRONMENT-END]
```

If the environment variable ADANUCLOG is neither defined nor valid, the default path for the *adanuc.log* will be in the same directory as the location of the DBnnn.INI file. The path for each DBnnn.INI is specified in `$ADADATADIR/etc/ADABAS.INI` for Linux platforms, or in `%ADADATADIR%\etc\ADABAS.INI` file for Windows platform under the topic INI_FILE.

If the value of INI_FILE has not been modified, the default path will be *$ADADATADIR/db<dbid>/adanuc.log* for Linux platforms, or *%ADADATADIR%\db<dbid>\adanuc.log* for Windows platform

ADANUCLOG and ADANUCLOGOLD are defined in one of two ways: as environment variable, or by using the adaini command to enable it in DBnnn.INI.

If ADANUCLOG is simultaneously defined in DBnnn.INI and as environment variable, the valid nucleus log is taken from environment variable setting.

If the environment variable ADANUCLOGOLD is defined as

- `COPY`, the previous session of the nucleus log is copied to the environment value <*ADANUCLOG*>*.old*, or the default location *$ADADATADIR/db<dbid>/adanuc.log.old* for Linux platforms or *%ADADATADIR%\db<dbid>\adanuc.log* for Windows platforms.
- `APPEND`, the previous sesson of the nucleus log is appended to the environment value <*ADANUCLOG*>*.old*, or the default location *$ADADATADIR/db<dbid>/adanuc.log.old* for Linux platforms, or *%ADADATADIR%\db<dbid>\adanuc.log* for Windows platforms.

These environment settings are used by the Adabas utilities when the corresponding environment variables are not defined.

The ADATCP_DNSLOOKUP environment variable enables or disables the reverse lookup of incoming adatcp connections. The default value is 'NO'. This prevents performance issues because a DNS (Domain Name Service) lookup can be very time consuming. The command 'ADAOPR DB=nnn DISPLAY=TCPCONNECTIONS' will then show the host names that are delivered by the client and a local port number 0. The host's name given by the client, also known as 'Node Id' in the user queue, might not be the real one (see also the client function lnk_set_adabas_id() in the *Command Reference, Calling Adabas*). If the 'real' host's names are wanted, set ADATCP_DNSLOOKUP to 'YES'.

Topic: NUCPARMS

```
# configuration                      # recommended values

[NUCPARMS]

  PLOG                               # Any nucleus parameter may be defined here.
  NU=100
  ....

[NUCPARMS-END]
```

The topic NUCPARMS defines any nucleus parameter with the exception of DBID. These values are used by adastart when starting the database. The DBID is passed to the nucleus by adastart, which itself takes it as a parameter.

These parameters can be verified by the administration command *adaini* with the VER parameter. Potentially unique abbreviations will then be automatically replaced by the correct item names (e.g. the abbreviation PORT will be automatically corrected to PORTNUMBER).

Topic: RESERVED_LOCATION

```
# configuration                      # recommended values

[RESERVED_LOCATION]
  LOCATIONx = <path name>            # n lines for disk locations with free space
[RESERVED_LOCATION-END]
```

When an auto expand of the database is necessary and a new container is to be created, Adabas searches for free space in the locations specified.

Topic: TEMPORARY_LOCATION

```
# configuration                      # recommended values

[TEMPORARY_LOCATION]
  TEMPLOCx = <path name>            # n lines for disk locations with free space
[TEMPORARY_LOCATION-END]
```

When Adabas needs temporary disk space for the nucleus or utilities, Adabas searches for free space in the locations specified.

4 Action Templates

The following table lists the templates of the AEO actions that are delivered with the distribution kit. The templates are in the directory *%ADAPROGDIR%\tools* (Windows) and *$ADAPROG-DIR/tools* (Linux).

Note: It may be necessary to customize these templates to suit your environment.

It is strongly recommended to start only the subtopics *ARCHIVE_LOGFILE* and *SAVE_DB* via ada_actn. All other actions should be used under the control of AEO itself.

Most action templates are implemented as batch files. They may, however, be replaced by binary executables.

Action Template	Description
ada_actn	Start an Adabas action; read the action routine name and start this routine (see for example the description of the subtopics SAVE_DB and ARCHIVE_LOGFILE).
ada_alrt	Notify all users listed in the environment variable ADA_ALRT_LIST.
ada_clualrt	Available only in a cluster environment. When the nucleus detects lost quorum or a change of primary node, this action template writes a message to $ADADATADIR/etc/ADABAS.LOG (default).
ada_dlcp	Use "adadbm dbid=<nnn> delcp=" to delete old checkpoints
ada_iass	Check if the new container is defined in the topic CONTAINER; if so, call "adadbm dbid=<nnn> add_container" to add a container
ada_idat	Check if the new container is defined in the topic CONTAINER; if so, call "adadbm dbid=<nnn> add_container" to add a container

Action Template	Description
ada_inuc	Compute new values for nucleus parameters using the following formula: new value = (current value) * 1.1 + (default value) / 10; (For a list of the default values of the nucleus parameters, see the chapter adanuc in the Adabas Utility Manual .) Modify the nucleus parameter in DB<nnn>.INI; For the nucleus parameters TNAA, TNAE, TNAX and TT use adaopr to increase the parameter for the current nucleus session. The calculation of the new parameter value is done in the program adainuc.exe, which is included in source (adainuc.c in the EXAMPLE\AEO directory) to be modified to any other value computation.
ada_nsta	empty
ada_nsto	empty
ada_reor	Check if this action is running after a successful database backup action SAVE_DB by checking the environment variable ADA_SAVE_DB (otherwise abort this action) For every file in the file list (parameters 7 - 106) do the following: -/tab/use "adaord dbid=<dbid> export=<fnr>" to save the file -/tab/use "adadbm dbid=<dbid> delete=<fnr>" to delete the file -/tab/use "adaord dbid=<dbid> import=<fnr>" to load the file
ada_rlst	Check if adanuc is online (DB Status must be ONLINE); call "adadbm dbid=<nnn> recover".
ada_svdb	Check definitions in SAVE_DB; expand the backup file name to BCK001=%ADADATADIR%\db<nnn>\BCK001.<dbid>.<xx> where <xx> is a number in the range 01 ... 99; perform the backup; xx is a two-digit number, and ada.svdb will search for the lowest value where the file %BCK001 does not exist if the backup is successful, set environment variable ADA_SAVE_DB and start "adarep dbid=<nnn> content" to check the action conditions for REORDER_FILE (adarep will start the action REORDER_FILE if this action is enabled and the conditions are true)
ada_svlg	Check the definitions in the topic ARCHIVE_LOGFILE; read the item LOG_FILE in the topic LOGGING and check the write permission; If the log file is empty, do nothing; Delete the oldest log file (filename is <LOG_FILE>.<MAXIMUM>); Rename all the other log files in the range 1 to <MAXIMUM-1> so that the log number is increased by 1, e.g. <LOG_FILE>.05 becomes <LOG_FILE>.06; Rename the latest log file from <LOG_FILE> to <LOG_FILE>.01

5 Administration Commands

This chapter describes the general purpose commands of Adabas Extended Operation (AEO).

Retrieve and Modify Information Stored in the Configuration Files: adaini

Usage: *adaini [DBID=<dbid>] {<add> | <del> | <show>> | <file> | <ver>}*

<dbid> is a numeric value between 0 and 255. If the DBID parameter is not specified, or if DBID=0 has been specified, the ADABAS.INI file is processed; otherwise the DB<dbid>.INI file is processed.

Adding or Modifying Information in a Configuration File

<add> is used to add or modify one or more items in a configuration file, and has the following syntax:

{ADD | MOD[IFY] } <topic_list> <item_value_list>

Note: ADD and MOD[IFY] are equivalent; you can also use ADD to modify items and MOD[IFY] to add items.

<topic_list> has the following syntax:

{ TOPIC=<topic> } ...

where <topic> is the name of a topic. If the item(s) to be processed belong to a subtopic, the complete hierarchy of topics to which the item(s) belong must be specified.

Note: Topic names are converted to upper case.

<item_value_list> has the following syntax:

{ITEM=<item>[=<value>] } ...

where <item> is the name of an item and <value> the value of the item.

Notes:

1. Items can be defined either with a value or without a value.
2. Unlike topic names, item names and item values are not converted to upper case.
3. adaini verifies the item names in the topic NUCPARMS only, potentially unique abbreviations will be automatically replaced by the correct item names, or an error message will be displayed. It does not check whether the topics or items specified are really used by Adabas, and whether the item values specified are valid; adaini only guarantees the syntactical correctness of the configuration files.

Example

```
adaini mod topic=node_parameter topic=analyser item=ACTION=no
```

This command sets the item ACTION in the topic NODE_PARAMETER, subtopic ANALYSER to no, and hence deactivates AEO.

Deleting Information from a Configuration File

<del> is used to delete one or more items from a configuration file and has the following syntax:

DEL[ETE] <topic_list> <item_list>

<topic_list> is used in the same way as for <add>.

<item_list> has the following syntax:

*{ITEM= *} | {ITEM = <item>} ...*

where <item> is the name of an item. The specified items are deleted; if you specify '*', all items and the topic to which they belong are deleted.

Notes:

1. In Linux shells, '*' is a special character, therefore you must precede it by a backslash or put it in quotes or double quotes.
2. If you specify all items that belong to a topic explicitly, only the items are deleted, but the topic to which the items belong remains as an empty topic in the configuration file. In order to delete the topic as well, you must specify ITEM=*.

Example

```
adaini dbid=36 del topic=environment item=ADAHYX_4
```

This command deletes the environment setting for ADAHYX_4 and hence deactivates hyperexit 4.

Showing Information from a Configuration File

<show> is used to show one or more items stored in a configuration file and has the following syntax:

show [<format>] <topic_list> [<item_list>]

<format> has the following syntax:

FORMAT={ BAT | BSH | CMD | CSH}

If you specify FORMAT, statements for the specified shell are generated, which create an environment variable with the item names to be processed as name and the item values as values.

<topic_list> is used in the same way as for <add>.

<item_list> has the same syntax as for <del>. If ITEM=* has been specified, all items belonging to the topic specified are displayed; if <format> has not been specified, they are displayed in the format <item>=<value>, followed by a line feed. If items are specified explicitly by their name, these items are displayed; if <format> has not been specified, only the values of the items are displayed, followed by a line feed.

If <item_list> has not been specified, all items belonging to the topic are displayed; if format has not been specified, they are displayed in the format <item>=<value>. Subtopics are also displayed; the layout is shown in the following example.

Examples

The command

```
adaini show topic=db_list
```

might generate the following output:

```
[DBID_001]
  INI_FILE=C:\Program Files\Software AG\Adabas\db001\DB001.INI
  NAME=V33-DATABASE
  STRLVL=12
[DBID_001-END]

[DBID_002]
  AUTOSTART=V616
  INI_FILE=C:\Program Files\Software AG\Adabas\db002\DB002.INI
  NAME=P289591
  STRLVL=12
[DBID_002-END]

[DBID_003]
  INI_FILE=C:\Program Files\Software AG\Adabas\db003\DB003.INI
```

```
  NAME=DEFAULT-DATABASE
  STRLVL=15
[DBID_003-END]

[DBID_004]
  INI_FILE=C:\Program Files\Software AG\Adabas\db004\DB004.INI
  NAME=TEST-ICU
  STRLVL=15
[DBID_004-END]

[DBID_005]
  INI_FILE=C:\Program Files\Software AG\Adabas\db005\DB005.INI
  NAME=ADA618-DB
  STRLVL=15
[DBID_005-END]

[DBID_012]
  INI_FILE=C:\Program Files\Software AG\Adabas\db012\DB012.INI
  NAME=GENERAL_DATABASE
  STRLVL=15
[DBID_012-END]

[DBID_036]
  AUTOSTART=NO
  INI_FILE=C:\Program Files\Software AG\Adabas\db036\DB036.INI
  NAME=GENERAL_DATABASE
  STRLVL=15
[DBID_036-END]

[DBID_062]
  AUTOSTART=NO
  INI_FILE=C:\Program Files\Software AG\Adabas\db062\DB062.INI
  NAME=GENERAL_DATABASE
  STRLVL=16
[DBID_062-END]
```

The command

```
adaini dbid=36 show format=bat topic=backup item=BCK001 item=BCK002 item=BCK003
```

might generate the following output:

```
set BCK001=C:\Program Files\Software AG\Adabas\db036\BCK001.036
set BCK002=C:\Program Files\Software AG\Adabas\db036\BCK002.036
set BCK003=C:\Program Files\Software AG\Adabas\db036\BCK003.036
```

Displaying Config File Name and Path of a Configuration File

<file> is used to display the path and name of the config file for Adabas or a selected DBID. It has the following syntax:

FILE

Verifying Information in a Configuration File

<ver> is used to verify items in the topic NUCPARMS of a configuration file. Potentially unique abbreviations will then be automatically replaced by the correct item names (e.g. the abbreviation PORT will be automatically corrected to PORTNUMBER). It has the following syntax:

VER[IFY]

Install Configuration Files: adainst

On Windows:

Usage: *adainst* `<dbid>`

If `<dbid>` is missing and ADABAS.INI does not exist in `%ADADATADIR%\etc`, *adainst* creates `%ADADATADIR%\etc\ADABAS.INI`.

The following steps are done by this command:

- create directory `%ADADATADIR%\etc`
- copy the template file `%ADAPROGDIR%\ADABAS.INI` to `%ADADATADIR%\etc\ADABAS.INI` if this does not yet exist.
- if `%ADADATADIR%\etc\ADABAS.INI` did already exist, check whether it already contains the topic NODE_PARAMETER and the DB_PARAMETER subtopic of the DB_DEFAULTS topic. If not, copy from the template.
- substitute the following values if required:
  NODE_NAME in topic MISCELLANEOUS
  LOG_FILE in subtopic LOGGING within topic NODE_PARAMETERS.

If `<dbid>` is specified and the topic DBID_`<dbid>` does not exist in ADABAS.INI, *adainst* creates `%ADADATADIR%\db<dbid>\DB<dbid>.INI`. The following steps are done by this command:

- read the topic DB_DEFAULTS from ADABAS.INI and copy it to DB`<dbid>`.INI.
- search for *assign.*sh* and *adanuc.*sh* files in the directory `%ADADATADIR%\db<dbid>` and copy container definitions into the topic CONTAINER

- search for *adanuc.*sh* files in directory %ADADATADIR%\db<*dbid*> and copy nucleus parameters into the topic NUCPARMS
- ask for user names in the topic ACTION_DBA
- display enabled/disabled actions
- use adarep dbid=<*dbid*>summary to get the database name and insert the item NAME into the topic DBID_<*dbid*> of ADABAS.INI

On Linux:

Usage: *adainst* <*dbid*>

If <*dbid*> is missing and ADABAS.INI does not exist in $ADADATADIR/etc, *adainst* creates $ADADATADIR/etc/ADABAS.INI.

The following steps are done by this command:

- create directory $ADADATADIR/etc
- copy the template file $ADAPROGDIR/ADABAS.INI to $ADADATADIR/etc/ADABAS.INI if this does not yet exist.
- if $ADADATADIR/etc/ADABAS.INI did already exist, check whether it already contains the topic NODE_PARAMETER and the DB_PARAMETER subtopic of the DB_DEFAULTS topic. If not, copy from the template.
- substitute the following values in the copied file:
  NODE_NAME in topic MISCELLANEOUS
  LOG_FILE in topic LOGGING (NODE_PARAMETER).

If <*dbid*> is specified and the topic DBID_<*dbid*> does not exist in ADABAS.INI, *adainst* creates $ADADATADIR/db<*dbid*>/DB<*dbid*>.INI. The following steps are done by this command:

- read the topic DB_DEFAULTS from ADABAS.INI and copy it to DB<*dbid*>.INI.
- search for *assign.*sh* and *adanuc.*sh* in the directory $ADADATADIR/db<*dbid*> and copy container definitions into the topic CONTAINER
- search for *adanuc.*sh* files in directory $ADADATADIR/db<*dbid*> and copy nucleus parameters into the topic NUCPARMS
- ask for user names in the topic ACTION_DBA
- display enabled/disabled actions
- use adarep dbid=<*dbid*>summary to get the database name and insert the item NAME into the topic DBID_<*dbid*> of ADABAS.INI

Kill Database: adakill

Usage: *adakill <dbid>*

adakill stops the Adabas nucleus for the database <dbid> as follows:

- (PC platforms:) by sending an interrupt (CTRL/BREAK). The parameter <dbid> must be specified.
- (Linux:) with Linux signal 15. The parameter <dbid> must be specified.

Caution: This command should only be used if adastop with the option ABORT is not able to stop the nucleus. Adabas will write a memory dump and will perform an AUTORESTART at the next startup.

The following steps are done by this command:

- get the process ID for adanuc
- send interrupt

Show Log File: adalog

Usage: *adalog [<dbid>] [-t]*

adalog displays the Adabas log file.

If <dbid> is specified, all entries for this database are displayed. If <dbid> is not specified, entries of all databases are displayed. If the option -t is used, adalog displays the end of the log file and continuously appends new lines from the log file to the display.

Write A Message To The Log File: adamsg

Usage: *adamsg DBID=<dbid> PID=<process ID> UTILITY=<utility name> MESSAGE=<message ID> TEXT=<message text>*

The following message IDs are supported:

- ABORTED

  For this message ID, TEXT contains the abort reason.
- INCNUCP

For this message ID, TEXT=nucleus parameter=<parameter>, current size=<current size>, new size=<new size>. This option is used in the action ada_inuc (increase nucleus parameter).

- INP

  For this message ID, TEXT=parameter assignment.

- STARTED

  For this message ID, TEXT is empty.

- TERMINATED

  For this message ID, TEXT is empty.

adamsg is the interface from batch files to the Adabas log file. Every batch file (as well as AEO actions) may use this interface to log messages. The order of the parameters is free, except that the TEXT parameter must be last in the parameter list. All parameter values except the TEXT parameter will be converted to upper case.

Example:

```
adamsg DBID=77 PID=4711 UTILITY=my_uti MESSAGE=ABORTED TEXT=file abc is empty
```

will generate following message line at the end of the log file:

```
004711 <date + time> 00077 <user name> %my_uti-F-ABORTED, file abc is empty
```

Furthermore, the ID of the current user (if available) is inserted into the ACTION_DBA topic.

Define Default Database: adaset

Usage (PC platforms): *[CALL] adaset <dbid>*

Usage (Linux): *adaset <dbid>*

adaset defines the following environment variables:

- ADADBID=<dbid>This is the default database ID.
- (PC platforms:) ADADBDIR=%ADADATADIR%\db<dbid>
  (Linux:) ADADBDIR=$ADADATADIR/db<dbid>This is the database working directory of the default database.

On Windows

In addition, adaset expands the PATH variable as follows:

```
PATH=%ADAPROGDIR%;%ADAPROGDIR%\tools;%PATH%
```

Note that the CALL command must be used when adaset is executed from a batch file rather than from a command prompt.

On Linux

In addition, adaset expands the PATH variable as follows:

```
PATH=$ADAPROGDIR:$ADAPROGDIR/tools:$PATH
```

In a C shell context, adaset is an alias that is created by a call of adaset.csh. In a Bourne shell context, adaset is a function that is created by a call of adaset.bsh. Before you use adaset, you must issue one of the following commands:

```
. $ADATOOLS/adaset.bsh  (Bourne shell)
source $ADATOOLS/adaset.csh (C shell)
```

These statements are already executed by adaenv.bsh (Bourne shell) or adaenv.csh (C shell).

Show Database(s): adashow

Usage: *adashow [<dbid>] [-a]*

adashow displays the following information for the database <dbid>:

- Database ID : the value specified by ADADBID
- Name : NAME in the topic DBID_<value in ADADBID>
- Version : Database version
- Config. File : INI_FILE in the topic DBID_<value in ADADBID>
- Status : either active or inactive
- TCP-Port : displays the TCP port number, if configured. The default port number is 49152. If the TCP parameter has not been configured, "not configured" is displayed. If the TCP parameter has been configured, but either ADATCP is not set or NOADATCP is set, "not enabled" is displayed.

- Adastart-Port: This is a Linux only option. Displays the adastart Port number, if configured. If the ADASTARTPORTNUMBER parameter is not configured or is set to value 0, "not configured" is displayed. There is no default port number for this parameter.

If <dbid> is missing, adashow displays the information for the default database specified by the environment variable ADADBID.

If the option -a is used, adashow displays the database ID, name, version and status for all configured Adabas databases on this node which are found in section DB_LIST in Adabas.INI.

Note: If adashow does not show all the expected information, the reason might be that abbreviated item names have been configured, possibly with a text editor. These item names can be verified and automatically corrected with the usage of *adaini verify*.

Start Database: adastart

Usage: *adastart [<dbid>]*

adastart starts the Adabas database <dbid>. The first time it is called, it creates the nucleus log file *adanuc.log*, on subsequent calls the nucleus log is saved with a time stamp, i.e. *adanuc.log.timestamp*.

On Windows:

The following steps are done by this command:

- Check if the nucleus is already online
- The environment variable ADANUCLOG is defined as the full path with the file name for the nucleus log output.
- If ADANUCLOG is not defined or valid, the default path for the adanuc.log will be in the same directory as the location of the DBnnn.INI file. The path for each DBnnn.INI is specified in *%ADADATADIR%\etc\ADABAS.INI* file under the topic INI_FILE. If the value of INI_FILE has not been modified, the default path will be *%ADADATADIR%\db<dbid>\adanuc.log*
- If the environment variable ADANUCLOGOLD is defined as "COPY", and the nucleus log of the previous session exists (file name *%ADADATADIR%\db<dbid>\adanuc.log*), the nucleus log is copied to *%ADADATADIR%\db<dbid>\adanuc.log.old*
- If the environment variable ADANUCLOGOLD is defined as "APPEND", and the nucleus log of the previous session exists (file name *%ADADATADIR%\db<dbid>\adanuc.log*), the nucleus log is appended to *%ADADATADIR%\db<dbid>\adanuc.log.old*
- Start the nucleus using the utility named Adabas which reads the nucleus parameters from DB<dbid>.INI.
- Wait until the nucleus is online or an Adabas error occurs.

On Linux:

The following steps are done by this command:

- Read the nucleus parameters from DB<dbid>.INI and write them into the file $ADADBDIR/nucparms.<dbid>.
- Check if the nucleus is already online.
- The environment variable ADANUCLOG is defined as the full path with the file name for the nucleus log output.
- If ADANUCLOG is not defined or valid, the default path for the adanuc.log will be in the same directory as the location of the DBnnn.INI file. The path for each DBnnn.INI is specified in *$ADADATADIR/etc/ADABAS.INI* file under the topic INI_FILE. If the value of INI_FILE has not been modified, the default path will be *$ADADATADIR/db<dbid>/adanuc.log*
- If the environment variable ADANUCLOGOLD is defined as "COPY", and the nucleus log of the previous session exists (file name *$ADADATADIR/db<dbid>/adanuc.log*), the nucleus log is copied to *$ADADATADIR/db<dbid>/adanuc.log.old*.
- If the environment variable ADANUCLOGOLD is defined as "APPEND", and the nucleus log of the previous session exists (file name *$ADADATADIR/db<dbid>/adanuc.log*), the nucleus log is appended to *$ADADATADIR/db<dbid>/adanuc.log.old*.
- Start the nucleus using the parameter file *$ADADBDIR/nucparms.<dbid>*.
- Wait until the nucleus is online or an Adabas error occurs.

Aditionally, adastart communicates with the nucleus and displays the nucleus messages. By default, adastart displays any error messages written into the nucleus.log file. If you want to display all the messages issued from the nucleus, you must provide an extra parameter while issuing the adastart call. For example:

```
adastart [<dbid>] [-v | --verbose]
```

where the `-v` or `--verbose` option prints the messages received from the nucleus.

For communicating with the nucleus, adastart needs a dedicated port. You can assign a port by providing an `ADASTARTPORTNUMBER` in the DB<dbid>.INI file under the `NUCPARMS` item. You must assign `ADASTARTPORTNUMBER` when calling adastart for the first time, otherwise then operating system assigns a free port number communicating.

By default, adastart waits 30 seconds for the next available nucleus message. If you want to increase this timeout value, you must add the `ADASTARTTIMEOUT` parameter in the `ENVIRONMENT` item. The value you provide must be in seconds.

If adastart fails to receive any adanuc messages within timeout, the script ends with an error and you must check the adanuc.log file.

Stop Database: adastop

Usage: *adastop [<dbid>]*

adastop stops the database <dbid>. If <dbid> is missing, adastop stops the default database specified by the environment variable ADADBID.

The following steps are done by this command:

- check if the nucleus is online or offline
- read the topic definition TERMINATE_ADANUC from DB<dbid>.INI
- take the defined shutdown options as defined in the topic and wait for the defined time intervals for the nucleus to stop