Adabas Utilities

Database services such as loading or deleting files are handled by an integrated set of online and batch-mode utilities. Most utilities can be run in parallel with normal database activity to preclude interruption of daily production. See the *Adabas Utilities Manual* documentation for more information.

Adabas utilities address initial design and load operations, backup/restore/recovery routines, database modification routines, and audit/control/tuning procedures.

Note:
Read *Adabas Online System* for information about this menu-driven, interactive DBA tool.

This chapter covers the following topics:

- Initial Design and Load Operations
- Backup / Restore / Recovery Routines
- Database Modification Routines
- Audit / Control / Tuning/ Reporting Procedures

Initial Design and Load Operations

- ADACMP: Compress / Decompress
- ADALOD: Loader
- ADAULD: Unload

ADACMP: Compress / Decompress

ADACMP COMPRESS is used to edit and compress data records to be loaded into the database using ADALOD; ADACMP DECOMPRESS is used to decompress individual files for data structure or field definition changes, or for use as input to non-Adabas programs.

COMPRESS

Input

ADACMP input data must be in a sequential data set/file. Indexed sequential and VSAM input cannot be used. The records may be fixed, variable, or of undefined length. The maximum input record length permitted depends on the operating system. The maximum compressed record length is restricted by the Data Storage block size in use and the maximum compressed record length set for the file (read about the MAXRECL parameter of the ADALOD utility). The input records can be in either blocked or unblocked format.

It is possible to omit the input data set if the parameter NUMREC=0 is supplied.

The logical structure and characteristics of the data for input to COMPRESS are described with field definitions statements (FNDEF to define fields or groups of fields; SUBFN and SUPFN to define sub- or superfields, respectively; COLDE, HYPDE, PHONDE, SUBDE, and SUPDE to define various types of descriptors). Field definitions are used to create the Adabas field definition table (FDT).

By default, input data records are processed in the order of the field definition statements. The FORMAT parameter allows you to change the order of field processing or skip fields.

To support universal encoding (UES), parameters allow you to specify the data architecture and user encodings of the input and the desired file and user encodings to use during compression.

Processing

ADACMP COMPRESS edits and compresses the data records.

Editing includes checking each field defined with a packed (P) or unpacked (U) format to ensure that the field value is numeric and in the correct format. Any record that contains invalid data is written to the ADACMP error data set and is not written to the compressed data set. Adabas user exit 6 can be used to specify additional editing to be performed during ADACMP COMPRESS processing. Read the *Adabas User, Hyperdescriptor, Collation Descriptor, and SMF Exits Manual* documentation for information about user exits.

Compression includes removing trailing blanks from alphanumeric fields; removing leading zeros from numeric fields; removing trailing zeros in floating-point format fields; and packing numeric unpacked fields. Fields with the fixed (FI) option are not compressed, and empty fields located at the end of the record are neither stored nor compressed. Null value fields are processed differently depending on options being used. SQL null value processing is supported.

If universal encoding support (UES) parameters have been specified, compression includes converting the input to the specified encoding for compressed files.

Output

Processed data records are written out together with the file definition information to a sequential data set with the variable blocked record format. This data set, or several such data sets from multiple ADACMP executions, can be used as input to the ADALOD utility. The data set can be used as input to ADALOD even if it contains no records, meaning that no records were provided on the input data set or all records were rejected during editing.

The ADACMP processing report indicates the approximate amount of space required in Data Storage for the compressed records by device type (specified with the DEVICE parameter) and for Data Storage padding factors between 5 and 30 percent. The compression rate is computed based on the real amount of data used as input to the compression routine.

DECOMPRESS

ADACMP DECOMPRESS accepts as input data records from existing Adabas files, either directly without separate file unloading, or already unloaded with the ADAULD utility. If a file is directly decompressed, it is unloaded without FDT information as part of the decompression process, which can save time when decompressing larger files.

Direct decompression of multiclient files can be limited to records for a specific user only when a valid owner ID (ETID parameter) is specified.

The FORMAT parameter may be used to decompress the record to a format other than that specified by the FDT. This is particularly useful when the FDT of an existing file is to be changed.

If universal encoding support (UES) is used, the encoding characteristics for the decompressed file are passed in the header of the compressed sequential input. Parameters allow you to overwrite these encoding characteristics.

Processed data records are written to a sequential data set with the variable blocked record format. Rejected data records are written to the error data set.

ADALOD: Loader

The ADALOD LOAD function loads a file into the database. Compressed records produced by the ADACMP or ADAULD utility may be used as input. A parameter specifies whether the file index is loaded in compressed or uncompressed form.

ADALOD loads each compressed record into Data Storage, builds the address converter for the file, and enters the field definitions for the file into the field definition table (FDT). ADALOD also extracts the values for all descriptors in the file together with the ISNs of all records in which the value is present, to an intermediate data set. This data set is then sorted into value/ISN sequence and entered into the Associator inverted lists.

The ADALOD UPDATE function is used to add or delete a large number of records to/from an Adabas file. The UPDATE function requires considerably less processing time than the repetitive execution of the Adabas add/delete record commands. Records to be added may be the compressed records produced by the ADACMP or ADAULD utility. The ISNs of records to be deleted can be provided either in an input data set or by using control statements.

Records may be added and other records deleted during a single execution of ADALOD.

ADAULD: Unload

The ADAULD utility unloads an Adabas file from the database or from a save tape.

Adabas files are unloaded from a database to

- permit the data to be processed by a non-Adabas program. In this case, the file must also be decompressed after unloading using the DECOMPRESS function of the ADACMP utility.
- create one or more test files, all of which contain the same data. This procedure requires that a file be unloaded, and then reloaded as a test file with a different file number.
- change the field definition table (FDT). This requires that the file be unloaded, decompressed, compressed using the modified field definitions, and reloaded. If the ADADBS utility is used to add field definitions to a file, the file does not need to be unloaded first.

The sequence in which the records are unloaded may be

physical	the order in which they are physically positioned within Data Storage.
logical	a sequence controlled by the values of a user-specified descriptor.
ISN	ascending ISN sequence.

The unloaded record output is in compressed format. The output records have the same format as the records produced by the ADACMP utility.

Adabas files may be unloaded from a qualified database or file save tape to

- include a file from a save tape in one or another test environment.
- move a file from a save tape with one block size to a database with another.

Backup / Restore / Recovery Routines

- ADAPLP: Protection Log / Work Print
- ADARAI: Recovery Aid
- ADARES: Restart
- ADASAV: Save / Restore Database or Files
- ADASEL: Select Protection Data

ADAPLP: Protection Log / Work Print

The ADAPLP utility prints data protection records contained on the Adabas Work data set or the Adabas data protection log. You can specify whether to print

ALL	all protection records-the default
ASSO	just Associator protection records
DATA	just Data Storage protection records
C1	records resulting from Adabas C1 commands
C5	records resulting from Adabas C5 commands
EEKZ	records written at completion of a nucleus buffer flush
ET	records resulting from Adabas ET commands
REPR	Work data set records used by autorestart to repair the index
SAVO	records resulting from online SAVE database/file operations
VEKZ	records written at completion of update commands

The number of protection records printed can be reduced even more by specifying a file, ISN, or RABN.

ADARAI: Recovery Aid

The Adabas Recovery Aid utility ADARAI can be used to automate and optimize database recovery. For more information, read *Adabas Restart and Recovery*.

ADARAI supports all Adabas-compatible tape management systems.

The ADARAI utility prepares the recovery log file (RLOG), which records the information about data sets, utility parameters, and protection logs needed to build the recovery job control statements. ADARAI lists the information contained in the RLOG, creates the job control statements to recover the database, and disables ADARAI logging.

Information is stored on the RLOG by generations. A generation includes all activity between consecutive ADASAV SAVE/RESTORE (database) or RESTORE GCB operations. The first generation includes the first ADASAV SAVE/RESTORE (database) or RESTORE GCB operation and extends to (but excludes) the second.

Minimally, the RLOG retains the number of generations specified by the MINGENS parameter during the ADARAI PREPARE step. However, a maximum of 32 generations will be stored on the RLOG if there is enough space available.

Systems using the Recovery Aid feature require a recovery log (RLOG) data set DD/RLOGR1, which must first be formatted with the ADAFRM utility and then defined using the ADARAI utility.

ADARES: Restart

The ADARES utility performs functions related to database recovery:

- BACKOUT removes all the updates applied between two checkpoints. The checkpoints used are normally the result of a non-synchronized checkpoint command (C1) but may also be synchronized checkpoints. The complete database may be included in the back-out process, or backout may be limited to selected files.
- CLCOPY copies a command log data set from disk to a sequential data set. This function is necessary only if dual or multiple command logging is in effect for an Adabas session.
- COPY copies a sequential Adabas protection log data set. This function should be executed if the Adabas session in which the sequential protection log data set was created was terminated abnormally.
- MERGE CLOG manually merges command log data sets resulting from individual nucleus CLCOPY executions into a single command log for a cluster of nuclei.
- PLCOPY copies a protection log data set from disk to a sequential data set. This function is necessary only if dual or multiple protection logging is in effect for an Adabas session.
- REGENERATE reapplies all the updates performed between two user-specified checkpoints. The checkpoints specified may be the result of a non-synchronized checkpoint command (C1) but may also be synchronized checkpoints. The REGENERATE function may process all files or be limited to one or more files. It is most often used after the database (or one or more files) has been restored to a previous status with the RESTORE or RESTONL function of the ADASAV utility.

- REPAIR repairs one or more blocks in Data Storage that, for any reason, have become unusable. The most recent save tape of the database and any protection log tapes created thereafter are used as input to this function.

To minimize the time required to recover after a system failure, the BACKOUT, BACKOUT DPLOG or MPLOG, and REGENERATE functions of ADARES can be executed in multiple threads that simulate the original update environment with multiple commands active at one time.

ADASAV: Save / Restore Database or Files

The ADASAV utility saves and restores the contents of the database, or one or more files, to or from a sequential data set. ADASAV should be run as often as required for the number and size of the files contained in the database, and the amount and type of updating. For large databases, ADASAV functions may be run in parallel for the various disk packs on which the database is contained.

Special ADASAV functions are available for use with the Adabas Delta Save Facility. For more information, read the *Adabas Delta Save Facility Facility* documentation.

RESTONL functions restore from one or more SAVE data sets created while the Adabas nucleus was *active* (that is, online); RESTORE functions restore from one or more SAVE data sets created while the Adabas nucleus was *inactive* (that is, offline).

RESTONL and RESTORE have the subfunctions GCB, FILES, and FMOVE:

- Without a subfunction, RESTONL and RESTORE restore entire databases.
- With the GCB subfunction, they restore the general control blocks, Associator RABNs 3-30 of the database, and specified files.
- With the FILES subfunction, they restore one or more files into an existing database to their original RABNs.
- With the FMOVE subfunction, they restore one or more files into an existing database to any free space, allowing changes to extent sizes.

If changes occurred during an online SAVE, the RESTONL function is followed automatically by the RESTPLOG function. RESTPLOG applies the updates that occurred during, and therefore were not included in, the online SAVE.

RESTPLOG is also executed following a RESTONL or RESTONL FILES function that ended before the protection log (PLOG) updates were completely restored. RESTPLOG applies the database updates not applied by the unsuccessful RESTONL function.

The SAVE function to save a database or one or more files may be executed while the Adabas nucleus is active (online) or inactive (offline). If the Recovery Aid option is active, a SAVE database operation begins a new RLOG generation.

ADASEL: Select Protection Data

The ADASEL utility selects information in the Adabas sequential (SIBA), dual, or multiple (PLOG) protection log. ADASEL decompresses the information and writes it to a print data set (DD/DRUCK) or to a user-specified output data set.

The protection log contains information on all updates applied to the database during a given Adabas session. Information selected by ADASEL can be used for auditing or as input to a Natural or non-Adabas program.

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

Database Modification Routines

- ADACDC: Changed-Data Capture
- ADACNV: Database Conversion
- ADADBS: Database Services
- ADADEF: Define a Database
- ADAFRM: Format Data Sets
- ADAINV: Invert
- ADAORD: Reorder
- ADAZAP: Modify Physical Database Blocks

ADACDC: Changed-Data Capture

ADACDC is an interval-driven, asynchronous mass update feature to generate a sequential file containing all database modifications. This feature is important for open systems and data warehousing solutions.

ADACDC then processes the raw data in the sequential file to isolate the latest status of the data. The ADACDC utility

- takes as input one or more sequential protection logs; and
- produces as output a *delta* of all changes made to the database over the period covered by the input protection logs.

A *delta of changes* means that the last change to each ISN in a file that was altered during this period appears on the primary output file.

This output may be used on a regular basis as input for data warehousing population procedures so that what is applied to the data warehouse database is the delta of changes to a database rather than a copy of the entire database. This affords more frequent and less time consuming updates to the data warehouse, ensuring greater accuracy of the information stored there.

ADACNV: Database Conversion

The utility ADACNV *must* be used to perform all necessary conversions of both operating system-dependent and -independent database system structures when moving in either direction between Adabas versions.

The ADACNV utility converts (CONVERT) an Adabas database from lower versions to higher versions; it also reverts (REVERT) an Adabas database from a higher version to a lower one. Some restrictions may apply these functions.

To ensure database integrity, ADACNV writes changed blocks first to intermediate storage; that is, to the sequential data set DD/FILEA. After all changed blocks have been written out to DD/FILEA, a point of no return is reached and the changed blocks are written to the database. If ADACNV terminates abnormally after the point of no return, the RESTART parameter can be used to begin the ADACNV run by reading the contents of DD/FILEA and writing them out to the database.

The TEST parameter is provided to check the feasibility of a conversion or reversion without writing any changes to the database.

ADADBS: Database Services

All ADADBS functions can also be performed using Adabas Online System (AOS). When the Adabas Recovery Aid is active, using AOS is preferable for file change operations because it writes checkpoints that are necessary for recovery operation.

ADADBS offers a variety of functions, any number of which may be performed during a single execution of the utility.

Database Functions

The ADD function adds a new data set to the Associator or Data Storage to a maximum of 99 data sets for each. However, your actual real maximum will be less because the maximum derived from the block size of the first Associator data set (DDASSOR1).

The DECREASE function reduces the size of the last data set currently being used for Associator or Data Storage. The space to be released must be available in the free space table (FST).

The DECREASE function does *not* deallocate any of the specified physical extent space. To deallocate space, you must decrease the database with the DECREASE function; save it with ADASAV SAVE; reformat the data sets with ADAFRM; and restore the database with ADASAV.

The INCREASE function increases the size of the last data set currently being used for the Associator or Data Storage. This function may be executed any number of times for the Associator. When the maximum (99) number of Data Storage Space Tables (DSSTs) has been reached, all Data Storage extents must be combined into a single extent with either the REORASSO or REORDB function of the ADAORD utility.

The RENAME function changes the name assigned to a file or database. If a file is not specified or is specified with file number zero, the database is renamed.

The TRANSACTIONS function suspends and resumes update transaction processing; that is, it creates a quiesced state for the database that could be a recoverable starting point.

File Functions

The ALLOCATE/DEALLOCATE functions are used to allocate/deallocate, respectively, a logical extent (an address converter, Data Storage, normal or upper index) of a specific size. Only one extent may be allocated or deallocated per ADADBS execution.

The CHANGE function changes the standard length of an Adabas field but does not modify records in Data Storage. The user is, therefore, responsible for preventing references to the field that would cause invalid results because of an inconsistency between the new standard length as defined to Adabas and the actual number of bytes contained in the record.

The DELETE function deletes an Adabas file from the database. The file may not be coupled. If an Adabas expanded file is specified, the complete expanded file (the anchor and all component files) is deleted. The deletion process deallocates all logical extents assigned to the file, releasing space that may be used for a new file or for a new extent of an existing file.

The DSREUSE function determines, for a specified file, whether Data Storage blocks that become free as a result of record deletion are reused. Block reuse is originally determined when a file is loaded into the database with the ADALOD FILE function, or when the system file is defined with the ADADEF DEFINE function. In both cases, block reuse defaults to YES.

To support universal encoding (UES), the ENCODEF function can be used to define encodings for fields in a file that is already loaded:

- an EBCDIC file encoding for alphanumeric fields; or
- a user encoding for the wide-character fields. The file encoding of wide-character fields cannot be changed using this function.

The ISNREUSE function determines, for a specified file, whether Adabas reuses the ISN of a deleted record for a new record. If not, each new record is assigned the next higher unused ISN.

For a specified Adabas file that is not a system file, the MODFCB function modifies parameters such as file padding factors for the Associator or Data Storage; maximum size of secondary logical extent allocations for Data Storage, normal index, and upper index; maximum compressed record length permitted; and whether a user program is allowed to perform a file refresh operation by issuing a special E1 command.

The NEWFIELD function adds one or more fields to a specified Adabas file that is not a system file. The new field definition is added to the end of the field definition table (FDT). NEWFIELD cannot be used to specify actual Data Storage data for the new field; the data can be specified later using Adabas add or update commands, or Natural commands.

The ONLINVERT function allows you to invert files when online applications are active, ensuring continuous access to the files. You can add one descriptor per file per run.

The ONLREORFASSO (reorder Associator), ONLREORFDATA (reorder Data Storage), and ONLREORFILE (reorder both Associator and Data Storage) functions allow you to reorder a list of files when online applications are active, ensuring continuous access to the files. Files are reordered within their existing extents, thus increasing I/O performance as free space is recovered and the sort sequence of data records is changed according to processing needs.

The REFRESH function sets the file to "0" records loaded; sets the first extent for the address converter, Data Storage, normal index, and upper index to *empty* status; and deallocates other extents.

The RELEASE function releases a descriptor from descriptor status. All space currently occupied in the Associator inverted list for this descriptor is released. The space can then be reused for this file by reordering or by ADALOD UPDATE. No changes are made to Data Storage.

The RENAME function changes the name assigned to a file or database. If a file is not specified or is specified with file number zero, the database is renamed.

The RENUMBER function changes the number of an Adabas file that is not a system file. If the new number specified is already assigned to another file, the RENUMBER function will not execute.

The UNCOUPLE function eliminates the coupling relationship between two files.

Other Functions

The CVOLSER function prints the Adabas file extents that are contained on a disk volume specified by its volume serial number.

The DELCP function deletes checkpoint information recorded up to and including a specified date; checkpoint information recorded after the date specified is not deleted. After running ADADBS DELCP, the remaining records are reassigned ISNs to include those ISNs made available when the checkpoint records were deleted. The lower ISNs are assigned but the chronological order of checkpoints is maintained.

The OPERCOM function issues operator commands to the Adabas nucleus. Adabas issues a message to the operator, confirming command execution. In cluster environments, OPERCOM commands can often be directed to another nucleus in the cluster or to all nuclei in the cluster for execution.

The PRIORITY function sets or changes the Adabas priority of a user. A user's priority can range from 0 (the lowest) to 255 (the highest, and the default). The priority value is added to the operating system priority by the interregion communications mechanism. The user for which a priority is to be set or changed is identified by the same user ID provided in the Adabas control block (OP command, Additions 1 field).

The RECOVER function recovers space allocated by rebuilding the free space table (FST). RECOVER subtracts file and DSST extents from the total available space.

The REFRESHSTATS function resets statistical values maintained by the Adabas nucleus for its current session. Parameters may be used to restrict the function to particular groups of statistical values:

- ALL (the default) resets values for the combination of CMDUSAGE, COUNTERS, FILEUSAGE, POOLUSAGE, and THREADUSAGE.
- CMDUSAGE resets the counters for Adabas direct call commands such as L*x*, S*x*, or A1.
- COUNTERS resets the counter fields for local or remote, physical or logical calls, format translations, format overwrites, autorestarts, protection log switches, buffer flushes, and command throw-backs.
- FILEUSAGE resets the count of commands for each file.
- POOLUSAGE resets the high-water marks for the nucleus pools such as the Work pool, the command queue, or the user queue.
- THREADUSAGE resets the count of commands for each Adabas thread.

Adabas maintains a list of the files used by each Adabas utility in the data integrity block (DIB). The DDIB operator command (or Adabas Online System) displays this block to determine which jobs are using which files. A utility removes its entry from the DIB when it terminates normally. If a utility terminates abnormally (for example, the job is canceled by the operator), the files used by that utility remain in use. The RESETDIB function releases any such files and resets the DIB entries for a specified job and/or a particular utility execution.

ADADEF: Define a Database

The ADADEF utility is used to

- define a new database (DEFINE functions), including the checkpoint file,
- set database encoding defaults for a new database or modify them (MODIFY function) for an existing database
- define a new Work file (NEWWORK function) for an existing database.

Databases are defined with name, ID, components (Associator, Data Storage, and Work) with device type and size, and default encodings.

Adabas uses certain files to store system information. The checkpoint file is used to store checkpoint data as well as user data provided with the Adabas CL and ET commands. It is required and must be specified in the ADADEF DEFINE (database) function.

Before database components (Associator, Data Storage, and Work) can be defined with ADADEF, each must be formatted by the ADAFRM utility.

ADAFRM: Format Data Sets

The ADAFRM utility formats the Adabas direct access (DASD) data sets; that is, the Associator, Data Storage, and Work data sets as well as the intermediate storage (temp, sort, recovery log, and dual or multiple command/protection log) data sets.

Formatting with ADAFRM comprises two basic operations: creating blocks (that is, RABNs) on the specified tracks/cylinders; and filling the created blocks with binary zeros (nulls).

Any new data set must be formatted before it can be used by the Adabas nucleus or an Adabas utility. After increasing a data set with the ADADBS INCREASE or ADD function, new RABNs must also be formatted.

ADAFRM also provides functions to reset existing Associator, Data Storage, or Work blocks to binary zeros (nulls).

More than one ADAFRM function (ASSOFRM, DATAFRM, RLOGFRM, and so on) can be performed in the same job. However, each function must be specified on separate statements.

ADAINV: Invert

The ADAINV utility is used to

- create a descriptor (INVERT function); or
- couple two files (COUPLE function).

The INVERT function

- modifies the field definition table (FDT) to indicate that the specified field is a descriptor; and
- adds all values and corresponding ISN lists for the field to the inverted list.

The newly defined descriptor may then be used in the same manner as any other descriptor. This function may also be used to create a subdescriptor, superdescriptor, phonetic descriptor, hyperdescriptor, or collation descriptor.

The COUPLE function adds a common descriptor to two files (updates their inverted lists). Any two files may be coupled provided that a common descriptor with identical format and length definitions is present in both files. A single file may be coupled with up to 18 other files, but only one coupling relationship may exist between any two files at any one time. A file may not be coupled to itself.

Note:
Only files with numbers 255 or lower can be coupled.

Changes affecting a coupled file's inverted lists are automatically made to the other file. The DBA should consider the additional overhead required to update the coupling lists when the descriptor used as the basis for coupling is updated, or when records are added to or deleted from either file. For example, if a field used as the basis for coupling contains a large number of null values and is not defined with the NU (null suppression) option, the result may be a significant increase in execution time and required disk space to store the coupling lists.

An interrupted ADAINV operation can be restarted without first having to restore the file.

ADAORD: Reorder

Three types of functions are available within the ADAORD utility; only one function may be executed during a given execution of ADAORD.

Reorder Functions

The REORASSO function physically reorders all Associator blocks for all files; REORFASSO reorders the Associator for a single file. This eliminates Associator space fragmentation, and combines multiple address converter, normal and upper index, and Data Storage Space Table (DSST) component extents into a single logical extent for each component.

The REORDATA function reorders Data Storage for all files in the database; REORFDATA reorders Data Storage for a single file. This condenses extents containing only empty blocks, and also eliminates any Data Storage fragmentation caused by file deletion.

The REORDB function performs both the REORASSO and REORDATA functions in a single ADAORD execution; the REORFILE function performs both the REORFASSO and REORFDATA functions in a single ADAORD execution. The records may be reordered in the logical sequence by a descriptor, by ISN, or in the current sequence.

Restructure Functions

The RESTRUCTURE functions are used to relocate a database or specified files to a different physical device.

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

Store Function

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

ADAZAP: Modify Physical Database Blocks

The ADAZAP utility is used to modify physical database blocks. It can be used to

- write a checkpoint for each VER and REP it processes providing an audit trail of database modifications. SYNP 3F checkpoints are printed by both Adabas Online System and ADAREP and are ignored by ADARES.
- handle errors according to standard Adabas utility conventions.

Because caution is necessary when running ADAZAP:

- Software AG recommends that you have a current save tape available before running ADAZAP. If an error is encountered while running ADAZAP, it may be necessary to restore the affected file or database.
- a mastercode available only to authorized personnel controls its use. The mastercode is distributed by Software AG on written request.

Audit / Control / Tuning/ Reporting Procedures

- ADAACK: Check Address Converter
- ADADCK: Check Data Storage
- ADAICK: Check Index and Address Converter
- ADAMER: ADAM Estimation
- ADAPRI: Print Selected Adabas Blocks
- ADAREP: Report
- ADAVAL: Validate the Database
- ADAWRK Utility: Work Area Recovery Reports

- ADAZIN: Print Database Information

ADAACK: Check Address Converter

ADAACK should only be used for diagnostic purposes. It checks

- the address converter for a specified file(s) and ISN range. It is used in conjunction with ADAICK.
- each address converter element to determine whether the Data Storage RABN is within the used portion of the Data Storage extents specified in the file control block.
- the ISN for each record in each Data Storage block within the specified ISN range to ensure that the address converter element for that ISN contains the correct Data Storage RABN.

ADADCK: Check Data Storage

ADADCK should only be used for diagnostic purposes. It checks the Data Storage and the Data Storage Space Table (DSST) of a specific file (or files) in the database.

ADADCK reads each used Data Storage block (according to the Data Storage extents in the file control block) and checks whether:

- the block length is within the permitted range (4 block length physical block size).
- the sum of the lengths of all records in the Data Storage block plus 4 equals the block length.
- any record exists with a record length greater than the maximum compressed record length for the file or with a length 0.
- any duplicate ISNs exist within one block.
- the associated DSST element contains the correct value. If not, the DSST must be repaired (read about the ADADCK REPAIR parameter).

ADAICK: Check Index and Address Converter

ADAICK should only be used for diagnostic purposes. It checks the physical structure of the Associator. This includes validating the index based upon the descriptor value structures and the Associator extents defined by the general control blocks (GCBs) and file control blocks (FCBs).

ADAICK can

- check index and address converter for specific files;
- print/dump the contents of any Associator or Data Storage block in the database; or
- produce a formatted print/dump of the contents of the GCBs, FCBs, and FDTs.

ADAMER: ADAM Estimation

The ADAMER utility produces statistics that indicate the number of Data Storage accesses required to find and read a record when using an ADAM descriptor. This information is used to determine

- whether the number of accesses required to retrieve a record using an ADAM descriptor would be less than the standard Adabas accessing method;
- the amount of Data Storage space required to produce an optimum distribution of records based on the randomization of the ADAM descriptor.

The input data for ADAMER is a data set containing the compressed records of a file produced by the ADACMP or ADAULD utility.

The field to be used as the ADAM descriptor is specified with the ADAMDE parameter. A multiple value field or a field contained within a periodic group may not be used. The ISN assigned to the record may be used instead of a descriptor as the basis for randomization (ADAMDE=ISN).

The ADAM descriptor must contain a different value in each record, since the file cannot be successfully loaded with the ADAM option of the ADALOD utility if duplicate values are present for the ADAM descriptor. The ADAMER utility requires a descriptor field defined as unique (UQ), but does not check for unique values; checking for unique descriptor values is done by the ADALOD utility when loading the file as an ADAM file.

The BITRANGE parameter may be used to specify that a given number of bits are to be truncated from each ADAM descriptor value before the value is used as input to the randomization algorithm. This permits records containing ADAM descriptor values beginning with the same value (for example, 40643210, 40643220, 40643344) to be loaded into the same physical block in Data Storage. This technique can be used to optimize sequential reading of the file when using the ADAM descriptor to control the read sequence, or to remove insignificant information such as a check digit.

ADAPRI: Print Selected Adabas Blocks

The ADAPRI utility prints the contents of a block (or range of blocks) contained in the Associator, Data Storage, Work, temp, sort, dual or multiple command log (CLOG), dual or multiple data protection log (PLOG), the recovery log (RLOG), or the Delta Save images (DSIM) data set.

ADAREP: Report

The ADAREP utility produces a status report that provides information concerning the current physical layout and logical contents of the database or a qualified save tape.

The information provided in this report includes:

- A database overview: the database name, number, creation date/time, file status, and current log number.
- Current space resources for Associator, Data Storage, and Work: amount and locations of currently used space, and allocated but unused space.
- Summary and detailed file information: summary by file of ISN, extent, padding factor, used/unused Associator and Data Storage space, and file options; and detailed, optionally by file, that includes all summary information plus MINISN/MAXISN settings, detailed space information, creation and last use date/time, field definition table (FDT) contents, and general or extended checkpoint file information.

- Checkpoint information: general and extended checkpoint file information.
- Physical structure: Associator/Data Storage RABN information including device type, VOLSER number, file number (if appropriate), and usage (AC, NI/UI, Data Storage, DSST, or unused).

The purpose of the save tape report is to determine what the save tape contains.

ADAVAL: Validate the Database

The ADAVAL utility validates any or all files within an Adabas database except the checkpoint and security files.

ADAVAL compares the actual descriptor values contained in the records in Data Storage with the corresponding values stored in the Associator to ensure that the Associator and Data Storage are synchronized, and that there are no values missing from the Associator.

Before running ADAVAL, the consistency of the inverted lists should be checked with the ADAICK utility.

ADAWRK Utility: Work Area Recovery Reports

The ADAWRK utility can be used to produce reports from records in the autorestart area of Work part 1. This information can be used when the database autostart fails and the database will not come up. In such a situation, you need to know what can be done to get the database back up and running quickly, with a minimum amount of lost data and with enough information to retrieve any lost updates.

The data on the ADAWRK reports can help you determine whether:

- You should run a restore/regenerate (ADASAV RESTORE utility function followed by the ADARES REGENERATE utility function) of the database, which can be time-consuming.
- Excluding specific files from the autorestart using AREXCLUDE and then restoring/regenerating only these single files would be beneficial.
- The database can be quickly repaired so it can be started and functional more quickly.

ADAWRK can produce the following reports:

- The Environment report is always produced, regardless of the ADAWRK parameters specified. It identifies the ADAWRK parameters used to produce the report as well as the Work data sets used for the report.
- The Summary report is produced by default and provides an overview of the data in the autorestart area of Work part 1.
- The File report is an optional report that provides a breakdown of the data in the autorestart area of Work part 1 by file.
- The Transaction report is an optional report that provides a breakdown of the data in the autorestart area of Work part 1 by transaction.

- The Checkpoint record report is an optional report that lists the checkpoint records and associated data within the autostart area of Work part 1.
- The Replication-related reports are optional reports that report on data that may need to be replicated.

The ADAWRK utility will only report on transactions that may need to be corrected as part of the autorestart processing logic. You can filter all of the Work part 1 autorestart area records processed and reports produced in an ADAWRK run by communication ID, ETID, user ID, and file number.

ADAZIN: Print Database Information

The ADAZIN utility can be used to print maintenance information about Adabas load modules and status information about the Adabas SVC on the system in which ADAZIN is run. Names of target load modules and SVC numbers can be specified to the utility to limit the range of the printed report. Module information in the ADAZIN report includes:

- The load module name;
- CSECT names (if appropriate);
- The date the module was last compiled;
- The Adabas version and release of the module;
- The number of the library from which the module was loaded within the concatenation list;
- A list of zap numbers applied to the module for the zap base level.
- The load module name CSECT names (if appropriate) The date the module was last compiled The Adabas version and release of the module The number of the library from which the module was loaded within the concatenation list A list of zap numbers applied to the module for the zap base level.

ADAZIN processing varies by operating system. In z/OS environments, the Adabas modules that ADAZIN reviews for the report reside in a load library (or concatenation of load libraries) defined through either the DDZIN or STEPLIB job control statements in the ADAZIN batch job. In BS2000 environments, ADAZIN uses the BLSLIB chain. Loading modules from the DDZIN link name is not supported. Finally, in z/VSE environments, ADAZIN uses the LIBDEF PHASE search chain to identify the libraries from which modules will be loaded. Loading modules from a library associated with DLBL DDZIN is not supported. There is no support for providing SVC status information on z/VSE. The SVC and SVCRANGE parameters are ignored in z/VSE environments.