Ϛ software AG

Adabas

DBA Tasks

Version 7.4.4

September 2009

Adabas

Table of Contents

1 DBA Tasks

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

This documentation describes the complete range of management and control tasks necessary for the successful operation of the database environment. This information is organized in the following parts:

DBA Roles and Responsibilities	This part reviews the role and functions of the database administrator (DBA).
Database Design	This part provides information on database design. It includes information on Adabas file structures, multiple value fields and periodic groups, record design, use of keys (descriptors), disk space usage (compression, null-value suppression, padding factors), security planning, restart and recovery planning, multiclient files, and expanded files.
Database Maintenance Tasks	This part describes tasks involved in defining and maintaining a database, including tuning and error handling.
Adabas Online System Demo Version	This part describes the Adabas Online System (AOS) demo version supplied with Adabas that provides also access to the online services of selected other Adabas products.
AFPLOOK	This part describes the Adabas Fastpath command analysis sampler (AFPLOOK), which can be used to determine where the best results may be expected from Adabas Fastpath by reporting on the command constructs that qualify for Adabas Fastpath.
AVILOOK	This part describes the Adabas Vista command analysis sampler (AVILOOK), which can be used to determine those files which may benefit from Adabas Vista's Partitioning option by reporting on the command constructs used to access the file.

2 DBA Roles and Responsibilities

The success of a database environment depends on central control of database design, implementation, and use. This central control and coordination is the role of the database administrator (DBA).

This part of the DBA documentation describes the roles of the DBA, the authority and responsibility the DBA might have, the skills needed, the procedures, standards, and contacts the DBA may need to create and maintain.

"DBA" in the context of this documentation is a single person; however, large organizations may divide DBA responsibilities among a team of personnel, each with specific skills and areas of responsibility such as database design, tuning, or problem resolution.

This information is organized under the following headings:

- **Central Control and Coordination**
- **The DBA in the IS Organization**
- **Establishing Database Control and Administration**
- **Data Definition and Control**
- **Database Documentation**
- **Education and Training**
- **The DBA and the User**
- **The DBA and Application Selection/Development**
- **The DBA and Computer Operations**

3 Central Control and Coordination

In a database environment such as Adabas, the same data is used by many applications ("users") in many departments of the organization. Ownership of and responsibility for the data is shared by departments with diverse and often conflicting needs. One task of the DBA is to resolve such differences.

Data security and integrity are no longer bound to a single individual or department, but are inherent in systems such as Adabas; in fact, the DBA controls and customized security profiles offered by such systems usually improve security and integrity.

In the past, application development teams have been largely responsible for designing and maintaining application files, usually for their own convenience. Other applications wishing to use the data had to either accept the original file design or convert the information for their own use. This meant inconsistent data integrity, varied recovery procedures, and questionable privacy safeguards. In addition, little attention was given to overall system efficiency; changes introduced in one system could adversely affect the performance of other systems.

With an integrated and shared database, such a lack of central control would soon lead to chaos. Changes to file structure to benefit one project could adversely influence data needs of other projects. Attempts to improve efficiency of one project could be at the expense of another. The use of different security and recovery procedures would, at best, be difficult to manage and at worst, result in confusion and an unstable, insecure database.

Clearly, proper database management means that central control is needed to ensure adherence to common standards and an installation-wide view of hardware and software needs. This central control is the responsibility of the DBA. For these and other reasons, it is important that the DBA function be set up at the very beginning of the database development cycle.

4   The DBA in the IS Organization

The ability of the database administrator (DBA) to work effectively depends on the skill and knowledge the DBA brings to the task, and the role the DBA has on the overall Information Systems (IS) operation. This section describes how best to define the DBA role, discusses the relationship of the DBA to the IS organization, and makes suggestions for taking advantage of that relationship.

This chapter covers the following topics:

Position of the DBA in the Organization

The DBA should be placed high enough in the organization to exercise the necessary degree of control over the use of the database and to communicate at the appropriate level within user departments. However, the DBA should not be remote from the day-to-day processes of monitoring database use, advising on and selecting applications, and maintaining the required level of database integrity.

The appropriate position and reporting structure of the DBA depends solely on the nature and size of the organization.

In most organizations, the DBA is best placed as a functional manager with an status equivalent to the systems, programming, and operations managers. The DBA should have direct responsibility for all aspects of the continued operation of the database. It is also useful to give the DBA at least partial control over the programming and IS operation standards, since the DBA must have the ability to ensure that DBMS-compatible standards are understood and observed.

Necessary Attributes for a DBA

The DBA is an essential resource to the organization: a politician, technician, diplomat, and policeman. The DBA needs to be a fair-minded person who is able to see both sides of database problems (that is, the IS department's side and the user's side) without prejudice in favor of either side. The DBA is expected to resolve problems for the benefit of the organization as a whole.

The DBA also needs

- administrative skill to set up and enforce the standards and procedures for using the database;
- technical ability to understand the factors governing hardware performance, with considerable knowledge both of the operating system software and the DBMS being used;
- a thorough knowledge of existing and future applications; and
- skills to produce an efficient database design that meets the application requirements.

In many medium-to-large installations, DBA functions are performed by a team rather than an individual. In this case, different members of the team specialize in different skills and aspects of managing database resources.

In a small installation, it may be difficult to justify a team, yet impossible to find an individual with all the necessary attributes. In this case, a DBA must rely on assistance from other specialists such as the systems programmer, senior operator, or senior analyst.

Management Support

To be effective, the DBA must be recognized and supported by both IS and user group management. With an in-depth understanding of the database operation and the service it provides to the organization, the DBA needs to be recognized as a center of competence for all matters involving the design or use of the database.

In principle, management should include the DBA in all decisions affecting the database to ensure that the database environment is not disrupted. Additionally, the DBA may often be able to suggest more cost-effective solutions that were known to management.

What Mistakes Are Possible?

When establishing the DBA function, the following mistakes should be avoided:

- Placing the DBA too low in the organization (insufficient authority). To function effectively, the DBA should be given enough authority to match the DBA's responsibilities. Far from being a threat to the established scheme of IS management, the DBA should be seen as a necessary adjunct when working in a DBMS environment. The DBA needs the cooperation, support, and respect of fellow managers, but will not have it if he or she is denied sufficient authority to perform the necessary tasks.

- Placing the DBA too high in the organization (too much authority). The position of the DBA should ensure the smooth operation of the DBMS environment, not bring it to a standstill under mounds of paper, unnecessarily restrictive procedures, or overbearing management. It is accepted that the dividing line between too little and too much authority is narrow, but the line must be recognized and drawn for each organization.

- Failing to define all DBA functions and responsibilities. The DBA should be authorized to perform the necessary functions, as they apply to the DBMS site. These functions need to be defined by participating managers from both the IS and user areas after careful consideration of the organization's requirements. Once the functions are defined, the DBA is responsible for establishing the procedures needed to ensure that they are performed.

■ Failing to select a DBA with sufficient administrative experience. The DBA function is not an appropriate place to teach administration to a junior manager. The DBA function requires considerable management expertise, particularly in the area of human relations.

5 Establishing Database Control and Administration

When establishing the system for controlling and administering the database environment, the general responsibilities of the DBA include

- establishing database procedures and standards;
- assisting in database design;
- educating users;
- selecting applications suitable for the database system;
- maintaining database documentation; and
- administering the database.

This chapter covers the following topics:

Establishing Database Procedures and Standards

Standards and procedures are more effectively established as part of the initial planning, rather than later after problems have arisen. This section discusses the general points to consider when defining procedures and standards.

Database Procedures

Procedures for effective control of the database environment should be established at the very beginning of the organization's use of a DBMS.

These procedures are outlined elsewhere in this documentation. Although many installations adopt the use of a DBMS in a short space of time, the planning aspect of the whole process (particularly in the design and implementation of administration and control procedures) must not be sacrificed just to enable startup in a minimum period of time.

Obviously, the implementation of these procedures will involve much discussion, both within IS and with the users (particularly in regard to what is acceptable, cost effective, etc.), and the first application of DBMS technology at a particular site may see a parallel development of the DBA procedures. However, it is essential that the organization's IS and user management be made aware of the need for such procedures, and (after due discussion) accept them.

Data Security Procedures

Who decides just how secure a data item must be? The users are too close and too personally involved in the matter. The analysts may miss the organization-wide implications of such a decision. The final arbitrator must be the DBA. After all, the DBA is the one who must monitor the procedures and correct the results of violations.

Planning Recovery Procedures

The DBA must establish standard recovery procedures for use at the installation. These procedures must be adequate for each application before it is implemented. Different approaches (file save/restore instead of database save/restore, for example) must be chosen; the DBA should participate in any decisions made in this area.

Setting Standards

Much of this documentation attempts to define guidelines for a database environment. Not all of the topics discussed will be relevant to a particular installation. Whether a guideline becomes a standard or not depends mostly on the size and diversity of the user/developer organization. Small, homogeneous user groups usually have good communication and do not require an extensive, rigidly defined set of rules.

On the other hand, larger user groups comprising various areas or geographic locations usually lack the contact necessary for proper controls; such groups may require "rules of the road" to avoid incongruous data structures or program development. No one likes rules, unless they see an obvious benefit in them. And standards are generally rules. So here are some "guidelines" to consider when defining standards:

▪ Keep standards brief, clear, and to a minimum. However, if a standard could be seen as arbitrary, give a brief justification. For example, a standard covering the use of temporary datasets in the Adabas environment might require the following:

```
Using Temporary Datasets
To ensure that your job is recovery-/restart-capable, always catalog temporary
datasets.
The Adabas database uses the Adabas Recovery Aid feature to automatically restore
and restart the nucleus,
rebuild failed job streams, and resubmit the rebuilt job. If temporary datasets
are not cataloged, the Recovery
Aid cannot include them in the rebuilt job stream.
```

▪ Review the standards at least as often as you add to them, and remove or revise outdated ones. Provide an overview of changed/deleted/added standards to users.

If a particular control or administration procedure is deemed to be necessary at a particular site, it should be defined as a standard.

Maintaining Procedures and Standards

The maintenance of database documentation should be treated as a natural part of the application development process. For this reason, the DBA will need to become involved in the development of each new system that uses the DBMS. The *data dictionary* is a major tool in this documentation process.

With a wide knowledge of current database applications, future plans, and responsibility for the integrity of the database, the DBA should be involved in the design of the data validation procedures inherent in each system that inputs data to the database. In this way, the DBA can ensure that the quality of the data in the database is maintained at an acceptable level. The Data Dictionary may also be used to document such validation requirements.

Assisting in Database Design

The assistance that the DBA provides to the project team in this area is best illustrated by the following questions:

- Is the logical design a true statement of the problem? A logical database design should not embody any limitations/features of a particular DBMS;

- Does the physical design cause any processing disadvantages? A project team will work in isolation on a specific problem unless otherwise directed;

- What about the future? Is the design flexible? The DBA and the organization will have to live with this design for some time.

As an independent function, the DBA is the only person who can provide such an objective view of the resulting database design. In some cases, the DBA may even become involved in the design process itself, and at such times will ensure that the right answers can be supplied to these questions.

Educating Users

Users can be oversold on the benefits which are to be derived from DBMS technology, despite the efforts of the system analyst or DP resources manager. Such topics as "flexibility", "program/data independence" and "data availability" can lead to unjustified expectations and give rise to a false sense of "creative freedom".

It is therefore the responsibility of the DBA to ensure that the user appreciates the problems as well as the benefits associated with working in a database environment. The DBA should devise, select and provide introductory training for the user that meets these needs.

Selecting Applications Suitable for the Database System

When an installation acquires a DBMS, it is only natural that analysts and programmers will be eager to use it. Every new application suddenly seems to require DBMS technology. Someone has to take a firm grasp of this situation and control the selection of applications that truly are suitable for DBMS technology. That person is the DBA.

The DBA should produce a list of the "pros and cons" of using the DBMS for each application. This should be done at the feasibility study stage (that is, before too much time and money is spent), before the system is acquired. The analysis of the proposed application should involve such considerations as:

■ Does the application need DBMS?

■ Will the application disturb our existing environment?

■ Will the proposed system be flexible?

■ Will it be cost effective?

■ Has the problem that led to proposal of the application been fully analyzed?

The DBA, as a center of competence in database matters, is thus the ideal person to oversee the selection of new database projects.

DBA Function Summary

The following is a summary of the functions for which the DBA is generally responsible:

Designing	Standard data definitions
	Physical database
	Security, privacy and recovery procedures
	Support software (if not acquired with the DBMS package)
Selecting	Database management system
	Performance measurement tools
	Tuning aids
Predicting	Effect of changing volumes/new applications
Deciding	Search strategies
	Access methods
	Database design
	Record relationships
	Rules of use of database
Training	Analysts and programmers: in database techniques
	Operators: in database operating procedures

Enforcing	Standards for design, documentation, etc.
	Quality control
	Access rules
Organizing/Administering	Data dictionary creation/maintenance
	File conversions
	Integrity, security and recovery benchmarks
	Acceptance tests
	Communication of changes to the users
Measuring	Hardware performance
	Software performance
	Database usage statistics
Tuning	System performance

6   Data Definition and Control

This chapter covers the following topics:

Planned Approach : Central Control of Data

Everyone involved with the database must apply a uniform methodology and standard procedure for data definition (this overlaps with the task of establishing the data dictionary). The DBA must formulate, establish, and maintain a consistent set of controls and standards in this area. These standards must be planned in conjunction with all affected parties: users, IS operations, applications designers, and the DBMS vendor.

Determining Responsibility for Data

Ultimately, one user department must be given the sole responsibility for maintaining a subset of the data in the database, ensuring its currency and integrity relative to the remainder of the database. The decision as to which user this will be is for the DBA to make. It is not only necessary to decide who shall be responsible; this decision must be made known to (and agreed by) all the other affected users. This is where the DBA's diplomacy will be called into play. Although it is natural that user A is responsible now, in a years time, when the use of the database has changed, it may be that user B is now the appropriate person to accept responsibility for some or all of the data.

Selecting Applications : Advising on System Development

The selection of applications for database implementation should be made by a committee, chaired by the DBA. The organization should decide whether or not the users should participate in this process. As a general rule, the user will only be interested in the cost, facilities, feasibility and extensibility of the system, if the database design team has performed it's data gathering and analysis tasks adequately. The DBA must be an impartial judge with the DBA's own independent advisers on the various topics which are likely to be discussed.

As a center of competence, the DBA and the related staff should be in a position to advise on systems development (but only insofar as to whether the DBMS should be used or not)-advice that can only be given if the DBA is serving a full and useful role within the database environment and has the wholehearted support of all interested parties.

As far as involvement in systems development is concerned, the DBA should be responsible for the process of defining and describing new data entities and relationships, using uniform data definition procedures. the DBA's is the task of maintaining records of the organization's "logical database" and controlling what part of it is and is not implemented.

Advising on Data Collection and Validation

The DBA should be responsible for establishing and enforcing uniform procedures for describing and defining the attributes of the data entities in the database. The DBA should also introduce standards for editing and validation of the input to the database.

Besides ensuring that the minimum criteria for data quality are met, it is important that the quality of the input be uniform so that the database remains as consistent as is practical.

The data dictionary can serve as a tool for the recording and implementation of these edit and validation rules.

Two types of data should be considered:

Private data	That is, data with a defined, single "owner". Here, the DBA can only insist that certain satisfactory data validation procedures and reasonability checks are performed;
Common data	That is, common-usage data. Unless a particular user can be identified who should have control and is prepared to accept that responsibility, the DBA should accept and exercise the appropriate level of control over the quality of such data.

Defining Database Contents

The majority of the documentation requirements for the database environment are supportable by the data dictionary. The data dictionary is one of the DBA's most important tools. It must be based upon a set of uniform data definition procedures as indicated above. The dictionary should record logical data formats and relationships and be broken down into three main areas:

Conceptual	the data and existing natural relationships
Usage	how it is used now
Implementation	how it is currently stored in the database

Standards are required relating to the use and/or interpretation of specific data entities.

The data dictionary should contain

- logical data structures;
- physical storage structures;
- data attributes;
- a description of data sources:

- Where the data comes from

- How it is obtained

- How it is edited and validated

- accuracy and security requirements:

    - What are the accuracy requirements?

    - What are the security requirements?

    - Who may access each data item?

    - Who may update each data item?

- response requirements: for each application area, what are the retrieval and response requirements?

Database Documentation lists these requirements in more detail. The online capability provided by Predict, the Adabas data dictionary, significantly reduces the effort involved in satisfying documentation requirements.

7 Database Documentation

Database documentation includes the recording of the procedures, standards, guidelines and database descriptions necessary for the proper, efficient and continuing use of the database.

The documentation must be specially prepared for and selectively distributed to

- end users;
- DBA function itself;
- computer operations function;
- applications development function.

The DBA has the responsibility for providing and maintaining adequate documentation for these recipients. This chapter discusses the types of documentation that are required. The list given is not necessarily exhaustive.

This chapter covers the following topics:

Standards

Establish and maintain a consistent set of database controls, particularly in the following areas:

- Data definition: a uniform methodology should be adopted;
- Data usage;
- Ownership of, and responsibility for, identifiable portions of the database;
- Data access and manipulation: Standards for the way data is accessed and updated in the database are crucial to ensure data integrity. Standard procedures for coding requests reduce the possibility of errors. For example, the updating of key values should be strictly controlled;
- Data edit and validation: The DBA must establish procedures to ensure compliance with the rules and maintain consistent levels of data quality in the database. The DBA should, therefore, become involved in systems and acceptance testing of new applications that use the DBMS;
- Computer operations: The DBA is responsible for ensuring that standard procedures are used by computer operations personnel when they deal with the database. This includes standard backup procedures, restart and recovery procedures, and other operations-related activities.

Some personnel will resist the establishment of standards for the database environment. Give careful consideration to the status of a standard and of areas where standards should be established. In general, a new standard will require negotiation, arbitration, and compromise before all the parties concerned will accept it even as a proposed standard. The only way to determine whether a standard is practical is to implement it.

Standards are subject to change. The process of changing an existing standard must be as tightly controlled as that of installing a new one. Changes should be formally proposed and communicated

to all the affected users. After a trial period, review the proposed change with users to decide whether it should become a standard.

Periodically review the database standards to evaluate their effectiveness and to ensure that they are being followed. Corrective action may need to follow such a review. The DBA also has the principal responsibility for ensuring that all personnel who work in the database environment are aware of, and adequately trained in, the use of the standards.

Because each site has its own procedures and requirements, this documentation does not suggest a specific set of standards.

Description of the Database

The database description should cover the following main areas:

- Conceptual database: a formal description of the data to be stored and a definition of its inherent logical data structures. The DBA should explicitly define data structures which reflect the DBA's knowledge of foreseeable developments and include the needs of other known or potential users.

- Use of the database: information should be recorded at both the application level and the individual data item level. This documentation will be of immense value when new systems are being developed. The information that should be recorded is discussed in the next subsection.

- Implementation: how the data is currently stored in the database. This is the documentation of the physical storage structure of the database and it should include (among other items):

  - Implemented data structures (logical data formats and relationships). These are normally a subset of the conceptual database;

  - Storage structure (physical data formats and relationships), in terms of Adabas, this means the contents of the Field Definition Tables, coupling relationships and other inherent relationships;

  - Volume of data: number and average size of records in each file;

  - Anticipated growth: by file and field size (and, therefore, record size);

  - Additions and deletions of records: number in an average maintenance run. It is also useful to keep a note of the distribution of additions and deletions (i.e., are they random across the file or restricted to a small portion of it), as well as the manner (user program or utility) in which these additions and deletions are performed.

In addition, the reasons why this particular implementation has been chosen should be recorded. This information will later prove useful when maintenance or new application design is undertaken.

The DBA is responsible for formally describing the database in the manner discussed above and maintaining this description on a data dictionary (whether this process is automated or not). The

project team will be required to provide the DBA with all the information the DBA needs to perform this task.

Data Dictionary, Function and Use

A data dictionary contains information about the definition, structure and use of data. It does not store the actual data itself, but rather data about data. Simply stated, the data dictionary contains the name of each data type (element), its definition (size and type), where and how it is used and its relationship to other data elements.

A data dictionary enables the DBA to exercise better management and control over the organization's data resources. Advanced users of data dictionaries have also found them to be valuable tools in project management and systems design.

The data dictionary will enable the DBA independently to manage actual data items and the programs that manipulate and access them. This independence of control results in the substantially enhanced usefulness of the data. The data dictionary serves to collect the information needed in order to make the data more useful.

Containing all of the definitions of the data, the dictionary becomes the information repository for the data's attributes, characteristics, sources, use, and interrelationships with other data. The data dictionary should provide the following information:

- The kind of validity tests which have been applied to this data type;
- What modules, programs, systems and reports use this data type?
- The level of security which has been applied; Who is allowed to access the data type? Who is authorized to update this data type?
- By what other names is the data type known in various application environments?
- What is the input source for this data type?
- Textual description of the data type.

Predict : The Adabas Data Dictionary

Predict, the Adabas data dictionary system, is used to establish and maintain an online data dictionary.

Database information may be entered into the dictionary in online or batch mode. The description of the data in the Adabas dictionary includes information about files, the fields defined for each file and the relationship between files. The description of use includes information about the

owners and users of the data in addition to the systems, programs, modules and reports that use the data. Dictionary entries are provided for information about

- network structures
- Adabas databases
- files, fields, and relationships
- owners and users
- systems, programs, modules and reports
- field verification (processing rules)

Standard data dictionary reports may be used to

- display the entire contents of the data dictionary
- print field, file, and relationship information
- print field information by file

In addition, the dictionary data can also be accessed directly from Natural since it is stored in a standard Adabas file.

Refer to the Predict documentation for more detailed information about the Adabas data dictionary system.

Applications Using the Database

For each application using the database, the following information should be recorded:

- Application characteristics:
    - Description of the function of the application;
    - Mode of use: batch/online, single or multiuser;
    - Frequency of use: standard scheduled times;
    - Type and volume of transactions;
    - Performance considerations: minimum acceptable response time;
- File requirements:
    - Which files are accessed;
    - How files are accessed (the use of descriptors);
    - Specific data items (fields, subfields, superfields, and so on) used;
- Security requirements:

- Ciphering and/or password usage (that is, cipher keys and/or passwords are not to be made generally available);

- Authorized users of this system/program/enquiry;

- Back-up requirements: frequency and content of file backups;

- Restart requirements.

Since any database is only a partial implementation of the conceptual database (see previous section) and user's requirements change with time, new applications of the database will be found as time passes. Some of these applications may be developed into new systems or additions to existing systems, but they first arise as a simple user requirement .

Establish procedures for recording unplanned applications of the database; if one becomes relatively frequent or important, you can often gain a processing advantage by redesigning or reorganizing the database or files within it. For example, assume that a file was initially loaded in Customer Number order. Subsequently, applications that process the file by Salesman Number assume greater significance. You can unload the file and reload it in Salesman Number sequence without affecting the logical operation of most existing applications, thus achieving an overall reduction in the processing time needed by all the applications that use the file.

Records of this unplanned use of the database or shift of emphasis in processing priorities, can be made when a user makes an interactive request for information, whether the user does this in his or her own department or through the DBA. These records should be regularly reviewed by the DBA and discussed with the affected users.

Description of Data Sources

For any new application, the data dictionary is the first reference document for determining the potential sources of information. The description of data sources will be derived during the systems analysis and design phases of a new project.

Record and retain the following information about each system:

- The present form and location of data: forms, files, computer storage media;

- Access techniques to be used to acquire the data;

- The intended use of the data in relation to its present accuracy, completeness, and timeliness, including necessary validation or editing;

- The need for modification of the data before it is stored on the database;

- The authorized agent for the use of the data;

- The cost of acquiring the data.

Data Access and Manipulation Procedures

The DBA must have administrative control over all access to and updating of the data in the database. Unless this is so, there can be little meaningful control or protection exercised over it. The lack of such control can result in serious security and integrity problems.

Because authority and responsibility for the database cross organizational boundaries, a corporate policy covering database usage by and among operating units should be published. Such policy statements can enhance the administrative control of the DBA and help to promote clear understanding of database procedures among users and data processing personnel.

Part of this policy will include statements on

- the use of DBMS commands-who can and (more important) cannot use the various facilities provided by the DBMS?

- database usage-is this to be achieved by user programs? Are standard interfaces such as Natural or SQL to be used? What error handling procedures should be observed? To whom should difficulties be reported, and how (for example, a trouble report)?

- maintenance and update procedures-who will be responsible?

This policy statement should be proposed and drafted by the DBA, and then reviewed and agreed upon by all the affected parties.

Passwords and User Identification

User ID and password information needs to be stored securely by the DBA, as only the DBA and the affected users should have access to it. This documentation will include

- assignment procedures for cipher keys, passwords and user identification;

- actual assignments: cipher keys (where it is necessary for the DBA to know this), passwords and user identifications;

- terminal and data access procedures;

- access authorities must be established for each data entity. These should define:

  - Who has the right and/or need to know the content of the data, as well as of its existence;

  - Who can read the data from the database, add new occurrences of the data, update existing values of the data, and/or delete the data from the database.

Once this authority has been established, it is important to set up proper control procedures in order to ensure that violations of database security do not occur.

- database security procedures. The physical protection of the data in the database should be recorded, detailing

  - positive human control over the database (communications rooms, access to the computer and terminals, storage of database backup and log tapes);

  - physical separation of data entities (separate files, separate databases, use of partial mount and file cluster facilities);

  - secure areas for terminals (lockable terminals, keyholders, leased or dial-up lines, taken offline when not in use);

  - persons authorized to receive published information about the database.

The DBA uses the Adabas Security utility to implement and control password security (see the Adabas Security documentation for more information). The DBA must be the only person at an installation who is permitted to use this utility.

The DBA must implement procedures to physically secure the security utility itself, and all documentation concerning security.

Back-Up Procedures

The content of back-up files, as either database or file copies (taken by the Adabas ADASAV utility) should be recorded together with the following information:

- What state (or point in time) the backup data refers to;
- Identification of the data which can be backed up;
- The volume of data that is involved;
- The backup facilities that should be used in order to reestablish the database to that state;
- The frequency and schedule of database backup operations.

The DBA should help computer operations personnel to develop procedures for carrying out the database backup task. Database backup is an essential step in ensuring that the database can be restored to its proper state in the event of destruction or damage. The decision will have to be taken (for every application) as to whether the entire database is to be backed up or whether dumping and restoring of specific files is more appropriate.

Information about developing backup procedures for a particular application is included in the Adabas Operations documentation.

Restart and Recovery Procedures

The DBA is responsible for formulating and supervising procedures for

- restarting the DBMS after failure;
- recovering the database to a recent checkpoint (if necessary), thus removing the need to repeat database maintenance work;
- controlling the priority and sequence of database restoration.

Restart and recovery is an important database protection consideration. The DBA must develop standards, procedures and rules to provide such a capability. The DBA must be certain that the standards and rules are being adhered to and enforced. Restart and recovery must be planned for and designed in conjunction with the implementation of the DBMS. It should not be added as an afterthought.

Detailed information about Adabas restart and recovery is included in the Adabas Operations documentation.

DBMS Performance and Measurement

The DBA has a continuing role in maintaining and improving the performance of the database system.

To do this, the DBA must monitor the performance of the system and try alternative design strategies to improve it. As work patterns change in the company, both the volume and relative proportion of types of transactions may change. This may affect performance and design changes may be necessary to counteract it.

In the longer term it may be possible to predict changes in workload, and plan how to meet them by redesign or equipment enhancement.

The effect of new hardware or software should also be evaluated, and possible changes should be cost-justified and incorporated into the long term strategy.

Keeping track of (and measuring) the performance of the DBMS is therefore an important part of the DBA's function. The DBA should establish and maintain records of

- the computing resources used, including frequency of use, by each application area;
- the users who are serviced by a particular application; and
- DBMS effectiveness with respect to response time and cost.

The DBA will also need to establish and document procedures for

- monitoring the frequency of DBMS usage; and

- DBMS performance management.

It is the responsibility of the DBA to monitor the database environment on a continuing basis, in order to ensure that an efficient level of service is provided while database integrity is maintained. This responsibility for monitoring takes the form of a variety of activities and procedures, of which performance management is but one.

The Adabas Online System provides the DBA with a powerful tool for monitoring the database. See the Adabas Online Systems documentation for more information.

8   Education and Training

This chapter briefly discusses the main features of a training program suitable for the database environment.

The following topics are covered:

Overview

The DBA is responsible for education and training in database concepts and the procedures and techniques involved in operating in the database environment. The DBA develops the training curriculum and selects the content of the training materials to be used. Information systems personnel must be trained to implement, operate, and maintain the database environment. Users external to the data processing area should receive training in database concepts, data availability, data entry, report generation, and the use of query facilities.

It is wise to produce a general training program for each type of person who will come into contact with the database environment. In this program, input (knowledge) expectations and output (performance) expectations should be recorded together with the training that is to be given, to ensure that the person meets the output expectations. A person requiring training can then be readily evaluated with the input criteria and remedial training, or pre-course reading can be prescribed before attending the appropriate training course. This approach will ensure the effectiveness of training.

The training given should correspond with the work requirements of the individual. The DBA's training should be carefully planned; it should be timely (i.e., not several months before or after the DBA is called upon to use it); and it should be immediately followed by a period of "reinforcement" (i.e., practical use of what the DBA has been taught.

When the DBMS is initially installed, a significant number of people will require training. The same is true when a new project starts or a new system is installed. Apart from these major requirements, ongoing training will be needed (for example, for new employees). For this reason, "packaged" training (for example, tape cassettes and workbooks) is recommended for the small numbers of staff and full courses for the large numbers.

Database Concepts

All personnel who interact with the database environment should have an understanding of the concepts of database management systems that includes

- why DBMSs evolved;

- similarities and differences between conventional data processing systems and DBMSs;

- advantages and disadvantages of the reduction of data duplication;

- flexibility inherent in the DBMS;

- ease of use and accessibility;

- the end user; differing "views" of the same data; the logical structuring capability of the DBMS;

- functions provided by the DBMS;

- importance of security, data integrity and recovery procedures in the database environment;

- need for database standards and controls.

Database Design

Database designers need training in the design methodology preferred at the site so that they can quickly become productive.

A large portion of training time should be spent on practical exercises that teach and give practice in the use of the site's standards, particularly for documentation. In a public course provided by Software AG, this may not be possible. In that case, the student should receive training in site standards immediately after returning to work.

The subjects taught should include

- a high-level understanding of Adabas facilities, their control and operation;

- loading files and file definitions; Adabas direct access method (ADAM) files; estimating disk space requirements;

- transaction processing; ET/BT logic;

- integrity; restart/recovery; autobackout; autorestart;

- security; passwords; ciphering;

- an overview of the Adabas utilities;

- program design and efficiency.

Programming

Training for computer programmers should be based on installation procedures and standards. The training must be as practical as is possible with a large portion of the time spent on exercises.

During the course, students should be expected to write an application program which will actually be run on a computer. To provide some measure of continuity and reinforcement, provision should be made for them to complete this exercise after the course has ended.

The subjects taught in this training should include

- an overview of the Adabas facilities applicable to the applications programmer;

- Adabas direct mode commands and/or high level programming interfaces (SQL, Natural) facilities available to the programmer;

- designing an Adabas program for efficiency and ease of maintenance.

Operating Procedures and Techniques

Training provided for computer operations personnel should be based on installation procedures and standards. It should also be as practical as possible (for example, running application systems, executing recovery and restart procedures).

The subjects taught should include

- operating procedures; starting an Adabas session; shutting down an Adabas session; normal operation; exceptions; problem recovery and restart;

- running utilities: what they do and what to expect;

- scheduling computer time; communication with the DBA;

- performance management;

- controls and audit trails;

- error reporting and follow-up.

Obviously, these topics are heavily installation-dependent and as such, the training provided in this area will need to be given by the installation's own staff.

Data Entry

This form of training will be an essential part of that given to personnel in the user department when a new application system is installed. As such, it is heavily application system-dependent. However, it is possible to give some general guidelines.

Training should include

- input procedures, whether at a terminal or by input document; application rules;

- standards and control; auditing;

- what the system does with the input;

- input errors and their correction.

Database Query and Report Generation

The content of this type of training will depend largely upon whether it is being given to data processing or user personnel. The former will require training in the commands and facilities of the query facilities to be used (for example, Natural) together with details of how to construct and run a request.

The user, on the other hand, will require much more specialized training. It will need to be geared much more closely to the application system that the DBA is to use.

The subjects that should be covered include

- how the query facility works (an overview only); for example, Natural or SQL;

- the standard reports produced by the application system-their contents and adaptability;

- the query facility commands, functions and use with an emphasis being given to the standards in force.

9   The DBA and the User

Before considering the normal database application development cycle and the DBA's role within it, the DBA must understand what the user requires from the database. The word "user" in its widest sense embraces user management and personnel, data processing management, computer operations, programming, systems, software support and the DBA's section.

The relationship between the DBA and the user community can be delicate, especially if a particular user actually or apparently must expend more effort or accept a lower level of service than would be the case outside the database environment.

The users should feel that the DBA is an impartial and unbiased authority whose decisions will enhance the welfare (and support the policies) of the organization as a whole.

The DBA must be aware of both corporate long-range plans and long-range user needs. The DBA must reconcile any conflicts that arise between users or between a particular user and any corporate plans that are affected.

Note that during the development of a new application, the DBA should involve the project group in any interaction with the end user.

This chapter covers the following topics:

Liaison with the User

Liaison with the user (whether programmer or end user) is the most important and sensitive part of the DBA's job.

In responding to an end-user request (which will normally be made in terms of retrieval requirements), the DBA should check the documentation describing the production database, particularly the logical data structure, to see whether the request can be readily handled.

Four basic outcomes are possible:

1. **The request cannot be fulfilled at present**
   In this case, the DBA should note the request, and review it at regular intervals to determine whether the situation (or need) has changed;

2. **Preparing a one-time request**
   Using Natural, it may be possible to satisfy the request from the existing database with minimum effort. Irrespective of whether the DBA's section actually writes the application to fulfill the request-indeed, the user department may have this capability-or if the application is written by a programmer, the point is that the DBA should define the solution to this one-time requirement for database information;

3. **Creating a new application program**

   Here the application is to be run regularly. The DBA will specify the program and negotiate with the programming manager for it to be written and tested;

4. **Changing an existing application**

   This may, of course, involve negotiation with the "prime owner" of the application system, if that person is different from the requestor, because any such change affect the performance or flexibility of the system.

Any end-user request should be fully documented, whatever the outcome. Such requests form one of the most useful information sources of information for the DBA as to whether or not the training supplied to the end user has been effective.

Requests from data processing personnel will normally be for

Training   The DBA should decide with the requestor what training is required and when. The DBA will need to cultivate an awareness in data processing management, that such training cannot be provided at a moment's notice. Such training should be properly planned in advance.

Information   The DBA will need to be satisfied that the requestor needs to know and is authorized to have the information. If the answer to either of these questions deviates from the normal situation, some temporary or permanent adjustment to existing standards and/or practices may need to be made.

Problems   The DBA will either know the answer or may need to refer the problem to Software AG. In the latter case, the DBA will need to assemble as much documentation on the problem as possible.

Assistance   This could take a variety of courses. The DBA should ensure that it is indeed the DBA's responsibility to provide the assistance that has been requested.

Access Requirements

The DBA must have administrative control over any access to and updating of the database. The DBA should establish with the users, the rights of access for each item in the database. Most of these access and update authorities will be evident from the design of the application systems which use the data and the data items will be secured with this in mind. When an unplanned request arises, the user should discuss this with the DBA. The latter, by reference to the database documentation, will be able to advise the user on the best way to satisfy the request. In addition, the very existence of the request is in itself useful input to the DBA's monitoring of the use of the database.

The requirement for access to the database, whether as a part of an application system or as an unplanned requirement, can be thought of as a "user view" or subschema of the database implementation. Its content, security, mode of access and manipulation should all be discussed and recorded.

Occasionally, the access requirement will cross application system boundaries. In this case, the DBA will need to discuss the right to access the data item with the item's "owner".

Application Interface

The documentation standards should define the normal interface for an application program to interact with the DBMS. One of two approaches may be used:

1. Direct calls to Adabas from a host programming language;

2. Calls for service to an access module.

Whichever of these two approaches is used, there will be cases where it is not appropriate or even possible to adhere to the standard interface policy. Before deviating from the standard interface technique, however, the DBA should be consulted and approval obtained.

When dealing with unplanned requirements, the DBA should advise the user on the interface approach to be adopted. This may be an application program, with or without an SQL access module; Natural; or something else.

Complying with Standards and Controls

The DBA should carefully explain to the user the benefits which are to be derived from conforming to database standards and control and the problems that can arise if any particular user decides to ignore them.

A feeling of mutual trust between the user and the DBA must be developed. The users should feel that the DBA is an impartial and unbiased authority, whose decisions will enhance the welfare and support the policies of the organization as a whole.

If the user is allowed to access the database using Natural, the DBA should be encouraged to record any unplanned requests and inform the DBA at regular intervals of these requirements. This is a part of the feedback and monitoring information that helps the DBA to ensure the continued effectiveness of the database environment.

10   The DBA and Application Selection/Development

This chapter covers the following topics:

Configuration and Applications Planning

From the DBA's knowledge of the use of the database and the monitoring of its performance, the DBA can contribute valuable data-processing expertise for making management decisions in the area of configuration and applications planning. The DBA is aware of the user's short- and long-term needs, as well as day-to-day problems and difficulties. The DBA's contacts with Software AG enable the DBA to keep abreast of the developments intended for the DBMS. The DBA should, therefore, be brought into this type of discussion.

The DBA should be involved in any application development project from the beginning. The DBA will be able to help in the initial survey in order to decide whether a database approach is justified in view of the organization's planned data processing developments.

The DBA will continue to be involved in project development after the initial implementation of the database project(s). The addition of new projects creates special problems which the DBA must resolve carefully.

The addition of new data and a changing use of existing data may change the performance characteristics of an existing system. Careful redesign of the physical structure and placement of data may be needed in order to give a reasonable service to all users.

Database Organization

As mentioned elsewhere, the DBA is responsible for the formulation and definition of the data relationships for the purpose of defining logical data structures. These data structures should reflect the DBA's knowledge of foreseeable developments and include the needs of other related users.

There are two major aspects:

- the definition and organizing of existing data; and
- the addition of new data.

Efficient physical structuring demands considerable expertise in translating and implementing logical relationships. Space, performance and cost must be balanced, taking into account

- data structure (logical data formats and relationships);
- storage structure (physical data formats and relationships);
- access methods (available and to be used);
- frequency of access;

- physical storage media requirements;

- timing considerations; and

- search strategies.

The solution (and the reasoning behind it) should be fully documented.

Understanding Current and Future User Requirements

The DBA is in an ideal position to help the members of a project team to appreciate and be aware of the user's current and future requirements. "User" is to be interpreted in its broadest sense. It does not only mean the section or department for which the application system is being designed and developed. User also means other potential users, not forgetting the organization as a whole. Known future developments must be taken into account. At times, this may mean that the DBA will need to exercise control over the development of a new application, in order that these developments may be readily included in the database operation when completed.

Coordinating Database Activities

Providing that the contents of this documentation are put into practice, the DBA will be able to coordinate all database activities. The DBA's advice should be sought on all developments planned for the database and the DBA should aim to ensure a steady, controlled progression to an integrated information system, which will serve the organization as a whole.

In general, the DBA should be involved in all stages of a new project from feasibility study onwards, both in order to advise on the practical uses of the database and also to carry out the DBA's quality control function.

The DBA will, therefore, provide lines of communication between different project teams, as well as with present and future users. The aim should be to cultivate the attitude of designing the database for the greatest benefit of all users.

Analyzing Access Requirements

This is an important part of the design of the database. When new projects reach the data analysis and file design stage, it is important for the DBA to ensure that the project team does not take too parochial a view of the requirements for access to the data to be used by the new application system.

The analysis of access requirements is also an ongoing task. As the requirements of the organization change (as they are bound to do with time), the DBA will be receiving feedback on these changing

requirements. However, the DBA should be careful not to overreact to a new requirement; it may only be required this one time. Rather, the DBA should respond to gradual and perceptible changes of emphasis in the access and/or processing requirements of the organization and even then, only after full discussion with all the affected parties.

Establishing Data Availability

The DBA should assist the project team (possibly by using the data dictionary) to plan a suitable data acquisition program, ensuring that the following aspects are taken into account:

- The present form and location of the data;

- How it is to be collected;

- How accurate and complete it is at present;

- What modifications are required to be made to the data before its inclusion in the database;

- At what time should the data be collected in relation to the implementation of the application system. How is the intervening period to be handled?

This process will result in a data collection program, which will include the necessary specifications for any special editing or validation that may be required, as well as providing information to the DBA for recording in the data dictionary.

Performance Versus Flexibility

The design of a (part of the) database will naturally involve consideration of performance (in the sense of disk space utilization and computer processing time), as opposed to flexibility (the ease of adapting to future unknown needs).

The DBA should ensure that the project team does not opt for performance at the expense of flexibility, and vice versa. The DBA is in a good position to advise the project team on which areas of the application need to be flexible (i.e., a planned system will also use this data) and which should be designed for good performance. The ultimate aim of such considerations of "performance" versus "flexibility" is to avoid making decisions where only one of these aspects is considered at the possible expense of the other.

Advising on Application/Program/Database Design

From the DBA's contact with project teams, with other Adabas users, and with Software AG, the DBA gradually acquires considerable knowledge about application program and data structure design. The DBA circulates appropriate information throughout the organization. In addition, the DBA must be available to advise on application design.

Although the DBA may not actually design the database, he/she must be able to advise team members on file/record design, descriptor selection, and other matters. This provides the DBA an opportunity to represent other users in the database design.

The DBA must ensure that the design of the physical database will efficiently support the logical requirements of the first application without prejudicing the success of later projects. The DBA will advise on how Adabas should be used in order to fulfill the security, integrity and recovery needs of the application system, design rules and procedures for these. In some cases, additional software may be needed and the DBA will help to design this. The DBA will consider whether additional utilities are needed for saving/restoring the database, measuring performance, analyzing the actual content of the database, and so on.

During this phase, the DBA may also be advising application project teams on the best approach to data analysis, the use of Adabas, how to design the logical data structure and which design options are likely to prove to be the most efficient.

Determining Physical Storage Requirements

It is the DBA's responsibility to provide assistance to the project team in determining the physical storage requirements for a particular application system.

The following parameters should be considered:

- The volume of data to be stored;
- The anticipated growth of data;
- The average size of records;
- The number of additions and deletions of records over a given time period;
- Data relationships (data structure);
- Data representation (internal formats);
- The effect of compression;
- The access methods which are to be used on the data.

Evaluate carefully the tradeoffs between minimizing the use of storage media and processing costs while maximizing service (measured in terms of speed or throughput). Also consider the need for flexibility in the implementation of the application system; requirements may be subject to change, and other application systems may need access to this application's data. If, minimizing physical storage requirements means a loss of flexibility, it should not be done without careful consideration of the problems that may arise in the future. The DBA is, of course, ideally placed to provide the project team with this type of information.

The Test Database and Testing Strategy

The DBA should advise the project team on the type of test database to be used for the new application. Assist the team by setting up the test database. During system testing, test with your own monitoring, audit, error correction, and control procedures before the system goes "live". These procedures should not be designed after the system has gone into production; they should be developed by the DBA in parallel with the development of the application system.

It is best to keep test databases separate from the production database by loading them onto separate disk packs, or even databases. This, in itself, poses two problems:

■ Tests cannot take place in parallel with production work in multiuser mode (in single-user mode, this problem does not exist);

■ Production data (or some fields in a production file) may be required to test the new system.

The first of these problems could be solved (if storage permits) by running two copies of the Adabas nucleus in parallel-one for production work, one for test work.

The second can be solved by using the Adabas ADASAV utility to copy across the required data from the production database to the test packs. In this case, the access authorization for the test data will have to be agreed before testing begins.

The main advantages of having a separate test database are that files can be loaded with the file numbers they will have when the system goes live and testing can in no way corrupt the production database. This is a particularly important consideration when fields are to be added to an existing file or new descriptors are to be established for the new application system.

Before systems testing starts, the DBA should decide how file conversion and database initialization is to be accomplished and ensure the preparation of any necessary special conversion or set-up programs. The strategy for parallel running will need careful consideration and here, too, special programs may be needed to assist in the comparison of outputs from the existing system and the new system or to carry out validity checks. Special inquiry facilities (e.g., Natural) may be needed to help testing and parallel running (these have sometimes also been found useful in subsequent live running).

Before the new database is finally implemented, acceptance tests should be run to demonstrate that all aspects of the system, including performance and resilience, are satisfactory. These may or may not be additional to the parallel runs.

Close control of the way in which the new project accesses data will be necessary in order to ensure that there is not loss of data integrity for existing users of the database. For systems testing, a special "testing mode" may be needed in order to ensure that "test changes" to the database do not actually affect the operational database.

11 The DBA and Computer Operations

This chapter covers the following topics:

Influence of the Database Administrator

The DBA carries the responsibility for ensuring that the computer operations function performs its duties with regard to the database environment. This responsibility is in terms of assisting the operations function to establish database related operating procedures, restart and recovery procedures, special database utilities and schedules for computer time for database related work.

The DBA also has a role in actually carrying out the day-to-day administration of the procedures and safeguards associated with the use of the database.

The DBA will ensure that the operational procedures are correctly adhered to, that dumps and logs are correctly taken and the DBA may also carry out periodic tests of the recovery systems.

In any emergency situation, the DBA may be involved in controlling recovery, discussing problems with users and generally working out ways of minimizing the disruption.

Scheduling Computer Time

The DBA should exercise some degree of control over the scheduling of the computer, in order to facilitate "scheduling around a problem" and to provide for priority use of the database in emergency situations.

While direct control over the computer schedule will reside with the computer operations personnel, it is, nevertheless, advisable to allow the DBA some degree of discretion in determining the schedule of events as they relate to database processing. In doing so, (for example) problems involving currency of update can be avoided and response time requirements during relatively infrequent peak load times can be satisfied without undue effort.

Operating Procedures

The DBA is responsible for working with computer operations personnel in order to develop formal and documented procedures for operating database-related jobs on the computer.

Among the areas that should be considered are

- loading a new database;
- running database utilities;
- maintaining the data dictionary;

- maintaining the database;
- backup procedures;
- restart/recovery procedures;
- production and testing requirements.

Restart and Recovery Procedures

The DBA must ensure that the database can be restored to its proper state in the event of destruction or damage. Restart and recovery is thus an important protection consideration and the DBA must develop standards, procedures, and rules to provide such a capability.

Computer operations personnel must be educated in and adhere to these standards and procedures in order to ensure that the recovery and restart of the database can be accomplished without loss of data integrity.

Any variations to standard practice (for example, a particular sequence of programs to be run after restart for a particular application system) should be recorded in the computer operations run book for that application.

Database Utilities

The DBA is responsible for controlling the use of Adabas utilities and for developing or acquiring specialized utilities to facilitate certain functions involving the database. These utilities may include

- creation of test databases of suitable size which include all the features of real-life databases (ADALOD utility);
- save/restore individual files or the entire database (ADASAV utility);
- provision of automated reports reflecting the integrity of the data in the database (ADAREP utility);
- provision of automatic reporting of security violations (ADALOG facility).

The DBA should retain control over when the utilities are run, including who is authorized to use them. The DBA's permission should be sought before a utility is used (except, of course, in the case of well-documented and tested recovery/restart procedures).

Working with Software AG

The DBA should be the primary contact between the organization and Software AG. The DBA's involvement with Software AG includes

- obtaining education and training for the organization's staff;
- receiving and installing new releases and system changes to the Adabas nucleus and utilities;
- receiving and distributing electronic documentation, manuals and other literature;
- obtaining advice;
- reporting problems;
- suggesting improvements to the system.

This section discusses these interfaces in detail.

Training and Education

Software AG supplies two types of education and training courses:

In-house Tailored to the particular requirements of an individual user site.

Open   General information; any user may participate.

In-house training is normally given when the Adabas system is first installed, although the DBA may from time to time have sufficient need for additional courses of this type. Such courses can be tailored to meet specific customer requirements and training objectives.

An open course is more general, and although thorough, it may not meet all of the DBA-defined specific requirements. As a result, the DBA may need to arrange supplementary training to meet objectives.

Training is offered by Software AG in the following areas:

- Application programming with Adabas;
- Database design;
- Query facilities (for example, Natural);
- Internals of the Adabas system.

Detailed descriptions of training, including recommended sequences, prerequisites, schedules, and enrollment information are available from your Software AG representative.

New Releases

When a new release has been thoroughly checked out, it will automatically be distributed to all Adabas user sites together with instructions which cover the means of effecting a transfer to the new release.

The new release should be thoroughly checked out by the DBA before production work is transferred to it. If this is the case, the DBA may find that a standard set of test programs, in the form of a prepared job stream, may be the best way of checking that the functions previously available still operate correctly. Such a test job stream will grow with each new function provided by Adabas.

Distribution of Documentation and Updates

As the sole recipient of new literature from Software AG, the DBA should keep a record of the copies distributed to ensure that the literature is kept as up-to-date as possible. A register of authorized document holders is easily maintained and is perhaps the easiest way to perform this part of the DBA's responsibilities.

Advice or Consultancy from Software AG

During the initial installation of the Adabas system, assistance is provided to install Adabas into the user's system library, generate a test database, and perform checkout tests.

Beyond this initial period, there may be occasions when the DBA feels the need for advice or consultancy from Software AG. Such a request should always come from the DBA.

Software AG will keep the DBA informed of any planned extensions to the Adabas package. As a general rule, such extensions will be included in the training courses as soon as they have been firmly defined by Software AG. The DBA, however, may need to pass on such information to existing projects in order that advantage can be taken of the new facilities as soon as they become available, thus eliminating the need for later redesign or reprogramming.

Problem Reporting

If a problem arises in the database, the DBA will most often be able to solve them without contacting Software AG. Nonetheless, Software AG offers comprehensive support to help restore operations as quickly as possible. The DBA can add to the effectiveness of this support by ensuring that the problem is defined accurately and succinctly to Software AG's technical support team. All available output should first be noted and/or collected for eventual reference and, if necessary and requested, should be sent to Software AG.

DBMS Improvement

Potential areas for system improvement logically occur as a result of the monitoring, auditing and operations activities. The DBA will have the responsibility for evaluating these potential enhancements and initiating any improvement activities. Software AG encourages and supports User Groups for its systems, which are an excellent forum for discussing such enhancements. Users can start the process by submitting a change/enhancement request to the appropriate User Group representative.

12 Database Design

This part of the DBA task description contains information about and guidelines for database design. Topics discussed include performance, file structure, record design, efficient use of descriptors, use of the Adabas direct access method (ADAM), disk storage space techniques, database recovery and restart procedures, and security.

This information is organized under the following headings:

- **Performance Control During System Design**
- **File and Record Design**
- **Data Access Strategies**
- **Disk Space Usage**
- **Adabas Security**
- **Recovery/Restart Design**
- **The Adabas Recovery Aid**
- **Multiclient Support**
- **Expanded Files**

13   Performance Control During System Design

The performance of a system is measured by the time and computer resources required to run it. These may be important for the following reasons:

- Some system functions may have to be completed within a specified time frame;

- The system may compete for computer resources with other systems that have more stringent time constraints.

Performance may not be the most important objective. Trade-offs often need to be made between performance and

- flexibility;

- data independence;

- accessibility of information;

- audit and security considerations;

- currency of information;

- ease of scheduling and impact on concurrent users of the database; or

- disk space.

In some cases, performance may be a constraint to be met rather than an objective to be optimized. If the system meets its time and volume requirements, attention may be turned from performance to other areas.

This chapter covers the following topics:

Methodology for Performance Control in System Design

The need to achieve satisfactory performance may affect

- the design of the database;

- the options defined when first loading the database;

- the logic of application functions (for example, whether to use direct access or a combination of sequential accesses and sorts); and/or

- operation procedures and scheduling.

Performance requirements must be considered early in the system design process. The following procedure may be used as a basis for controlling performance:

1. Obtain from the users the *time* constraints for each major system function. These requirements are likely to be absolute; that is, the system must meet them.

2. Obtain the computer *resource* constraints from both the users and operations personnel and use the most stringent set.

3. Describe each function in terms of the logical design model specifying the

- manner in which each record type is processed;

- access path and the sequence in which records are required;

- frequency and volume of the run;

- time available.

4. *Decide* which programs are most "performance critical". The choice may involve volumes, frequency, deadlines, and the effect on the performance or scheduling of other systems. Other programs may also have minimum performance requirements which may constrain the extent to which critical functions can be optimized.

5. *Optimize* the performance of critical functions by shortening their access paths, improving their logic, eliminating database features that increase overhead, and so on. On the first pass, an attempt should be made to optimize performance without sacrificing flexibility, accessibility of information, or other functional requirements of the system.

6. *Estimate* the performance of each critical function. If this does not give a satisfactory result, either a compromise between the time constraints and the functional requirements must be found or a hardware upgrade must be considered.

7. *Estimate* the performance of other system functions. Calculate the total cost and compare the cost and peak period resource requirements with the economic constraints. If the estimates do not meet the constraints, then a solution must be negotiated with the user, operations, or senior management.

8. If possible, *validate* the estimates by loading a test database and measuring the actual performance of various functions. The test database should be similar to the planned one in terms of the number of records contained in each file and the number of values for descriptors. In the test database, the size of each record is relatively unimportant except when testing sequential processing, and then only if records are to be processed in physical sequence.

14 File and Record Design

It is possible to design an Adabas database with one file for each record type as identified during the conceptual design stage. Although such a structure would support any application functions required of it and is the easiest to manipulate for interactive queries, it may not be the best from the performance point of view, for the following reasons:

- As the number of Adabas files increases, the number of Adabas calls increases. Each Adabas call requires interpretation, validation and, in multiuser mode, supervisor call (SVC) and queueing overhead.

- In addition to the I/O operations necessary for accessing at least one index, address converter, and Data Storage block from each file, the "one file per record type" structure requires buffer pool space and therefore can result in the overwriting of blocks needed for a later request.

For the above reasons, it may be advisable to reduce the number of Adabas files used by critical programs. The following techniques may be used for this procedure:

- Using multiple-value fields and periodic groups;

- Including more than one type of record in an Adabas file;

- Linking physical files into a single logical (expanded) file;

- Controlling data duplication (and the resulting high resource usage).

Each of these techniques is described in the following sections.

This chapter covers the following topics:

Multiple-Value Fields and Periodic Groups

The simple example below shows the practical use of a periodic group:

Order Number	Order Date	Date Filled	Customer	Date Required	Item Code	Quantity
A1234E	29MAR	--	UK432M	10JUN	24801K	200
		--		15APR	30419T	100
		--		01JUN	273952	300

ORDERS File Record

Order_No	Ord_Date	Cust_No	Req_Date	Fill_Date	Item_Code	Quantity
			1006		24801K	200
			0106		27395R	300
A1234E	2903	UK432M	1504		30419T	100

Order_Item Periodic Group
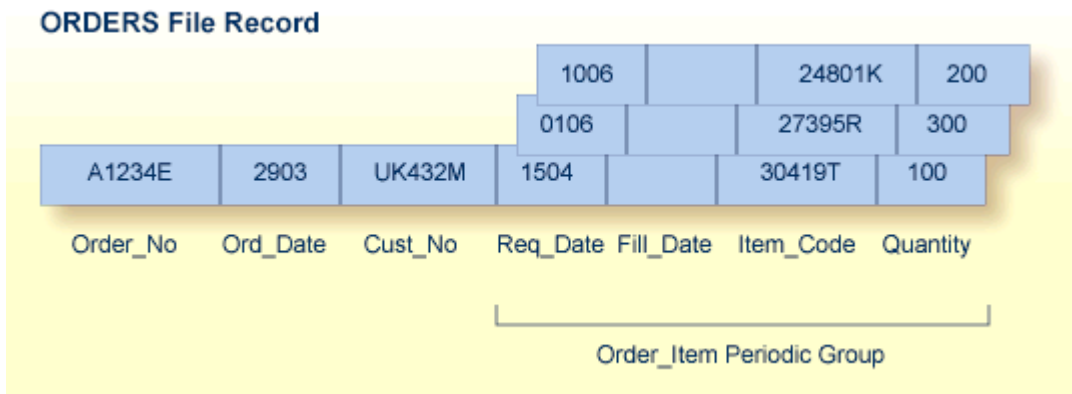
Example of a Periodic Group

In the example shown in the table above, the order information in the table is shown converted to a record format in an Adabas file called ORDERS. Each order record contains a periodic group to permit a variable number of order items. In this case, the periodic group ORDER_ITEM, comprising the ITEM_CODE field and the related fields QUANTITY, REQ_DATE, and FILL_DATE, can specify up to 191 different items, quantity desired, and the date needed as well as when the order is actually filled. Each item/quantity/date needed/date filled group is called an "occurrence"; up to 191 occurrences per periodic group are possible.

The unique characteristic of the periodic group-the ability to maintain the order of occurrences-is the reason for choosing the periodic group structure. If a periodic group originally contained three occurrences and the first or second occurrence is later deleted, those occurrences are set to nulls; the third occurrence remains in the third position. This contrasts with the way leading null entries are handled in multiple-value fields, discussed below.

Note also that the record format shown for the ORDERS file may not seem the most logical; however, fields most likely to contain nulls should be placed together and at the end of the record to save database space. The fields comprising periodic groups, therefore, are combined after the other fields in the record.

On the other hand, the ORDERS file record structure, while being appropriate for managing orders, may not as desirable when managing inventory. A stock control application for the items in the ORDER file can require a completely different record structure. These records are kept in a different database file called STOCK (see the figure below).

STOCK File Record
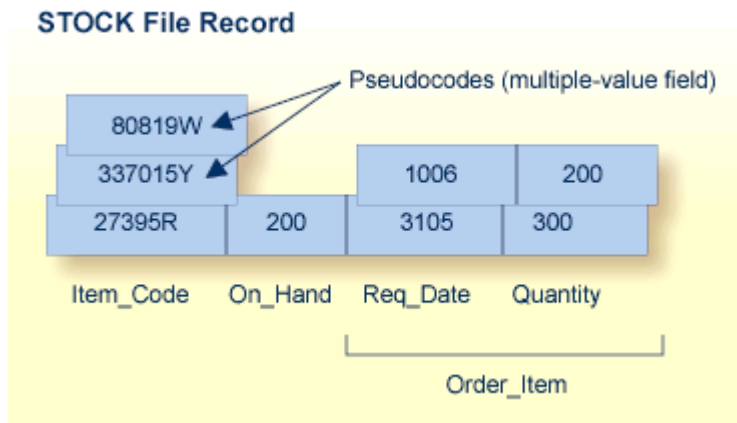
Example of a Multiple-Value Field

The record format in STOCK is more suitable to the applications required for stock management than the format in the ORDERS file. The record is designed to handle cases where an item is designated as a replacement for another that is no longer in the inventory. By allowing multiple values for the ITEM_CODE field, the current stock item can also be labelled with the numbers of discontinued items that the new item replaces, allowing references to the old items to automatically select the new replacement item. To do this, the ITEM_CODE field is defined as a multiple-value field.

For example, the items 80819W and 337015Y are no longer in stock; their item codes have become synonyms for the basic item 27395R. An application program that inquires about either discontinued item can first look through all ITEM_CODE values for the old code, and then refer to the first ITEM_CODE value in the multiple-value field to identify the replacement.

The ITEM_CODE field may contain from one to 191 values. Unlike a periodic group, however, the individual values in a multiple-value field do not keep positional integrity if one of the values is removed. For example, if the item 337015Y in the STOCK record shown above can no longer be ordered and the pseudocode is set to a null, 80819W automatically becomes the second occurrence under ITEM_CODE.

The following limits apply when using multiple-value fields or periodic groups:

- The maximum number of values of any multiple-value field is 191;
- The maximum number of occurrences of any one periodic group is 191;
- A periodic group cannot contain another periodic group;
- Depending on the compressed size of one occurrence, their usage can result in extremely large record sizes which may be larger than the maximum record size supported by Adabas.

Descriptors contained within a periodic group and subdescriptors or superdescriptors derived from fields within a periodic group cannot be used to control logical sequential reading or as a sort key in find and sort commands. In addition, specific rules apply to the ways in which search requests involving one or more descriptors derived from multiple-value fields and/or contained

within a periodic group may be used. These rules are described in the Adabas Utilities documentation, ADACMP utility.

Different Record Types in a Single Adabas File

Another method of reducing the number of files is to store data belonging to two logical record types in the same Adabas file. The following example shows how a customer file and an order file might be combined. This technique takes advantage of Adabas null-value suppression.

```
Fields in the field definition table for the combined file:

Key, Record Type, Order Data, Order Item Data

Stored records:

Key Type Order Data*

Key Type * Order Item Data

  * indicates suppressed null values.
```

The key of an order item record could be order number plus line sequence number within this order.

This technique reduces I/O operations by allowing the customer and order record types to share control blocks and higher-level (UI) index blocks. Fewer blocks have to be read before processing of the file can start, and more space is left free in the buffer pool for other types of blocks.

The customer and order records can be grouped together in Data Storage, reducing the number of blocks that have to be read to retrieve all the orders for a given customer. If all the orders are added at the same time the customer is added, the total I/O operations required will also be reduced. If the orders are added later, they might not initially be grouped in this way but they can be grouped later by using the ADAORD utility.

The key must be designed carefully to insure that both customer and order data can be accessed efficiently. To distinguish different orders belonging to the same customer, the key for a customer record will usually have the null value of the suffix appended to it, as shown below:

```
A00231          000         Order header for order A00231
A00231          001         Order item 1
A00231          002         Order item 2
A00231          003         Order item 3
A00232          000         Order header for order A00232
A00232          001         Order item 1
```

A record type field is unnecessary if the program can tell whether it is dealing with a customer or order record by the contents of the key suffix. It may be necessary for a program to reread a record to read additional fields or to return all fields that are relevant to any of the record types.

Linking Physical Files in a Single Logical File

An Adabas file with 3-byte ISNs can contain a maximum of 16,777,215 records; a file with 4-byte ISNs can contain 4,294,967,294 records. If you have a large number of records of a single type, you may need to spread the records over multiple physical files.

To reduce the number of files accessed, Adabas allows you to link multiple physical files containing records of the same format together as a single logical file. This file structure is called an "expanded file", and the physical files comprising it are the "component files". An expanded file can comprise up to 128 component files, each with a unique range of logical ISNs. An expanded file cannot exceed 4,294,967,294 records.

> **Note:** Since Adabas version 6 supports larger file sizes and a greater number of Adabas physical files and databases, the need for expanded files has, in most cases, been removed.

Although an application program addresses the logical file (the address of the file is the number of the expanded file's base component, or "anchor" file), Adabas selects the correct component file according to the data in a field defined as the "criterion" field. The data in this field has characteristics unique to records in only one component file. When an application updates the expanded file, Adabas looks at the data in the criterion field in the record to be written to determine which component file to update. When reading expanded file data, Adabas uses the logical ISN as the key to finding the correct component file.

Adabas utilities do not always recognize expanded files; that is, some utility operations automatically perform their functions on all component files, and others recognize only individual physical files. See *Expanded Files* for more information.

Data Duplication

Physical Duplication

In some cases, a few fields from a header record are required almost every time a detail record is accessed. For example, the production of an invoice may require both the order item data and the product description which is part of the product record. The simplest way to make this information quickly available to the invoicing program is to hold a copy of the product description in the order item data. This is termed physical duplication because it involves holding a duplicate copy of data which is already physically represented elsewhere-in this case, in the product record. Physical duplication can also be in effect if some fields from each detail record are stored as a periodic group in a header record.

Physical duplication seldom causes much of a problem if it is limited to fields that are updated only infrequently. In the example above, the product description data rarely changes; the rule is: the less activity on duplicated fields, the better.

Logical Duplication

Assume a credit control routine needs the sum of all invoices present for a customer. This information can be derived by reading and totalling the relevant invoices, but this might involve random access of a large number of records. It can be obtained more quickly if it is stored permanently in a customer record that has been correctly maintained. This is termed "logical duplication" because the duplicate information is not already stored elsewhere but is implied by the contents of other records.

Programs that update physically or logically duplicated information are likely to run more slowly because they must also update the duplicate copies. Logical duplication almost always requires duplicate updating because the change of any one record can affect data in other records. Logical duplication can also cause severe degradation in a TP environment if many users have to update the same record.

Adabas Record Design

Once an Adabas file structure has been determined, the next step is usually to define the fields for the file. The field definitions are entered as input statements to the ADACMP utility's COM-PRESS function, as described in the Adabas Utilities documentation This section describes the performance implications of some of the options that may be used for fields.

The fields of a file should be arranged so that those which are read or updated most often are nearest the start of the record. This will reduce the CPU time required for data transfer by reducing

the number of fields that must be scanned. Fields that are seldom read but are mainly used as search criteria should be placed last.

For example, if a descriptor field is not ordered first in the record and logically falls past the end of the physical record, the inverted list entry for that record is not generated for performance reasons. To generate the inverted list entry in this case, it is necessary to unload short, decompress, and reload the file; or use an application program to reorder the field first for each record of the file.

Combining Fields

If several fields are always read and updated together, CPU time can be saved by defining them as one Adabas field. The disadvantages of combining fields in this way are:

- More disk space may be required since combining fields may reduce the possibilities for compression;
- It may be more difficult to manipulate such fields in query language programs such as SQL.

Using Field Groups

The use of groups results in more efficient internal processing of read and update commands. This is the result of shorter format buffers in the Adabas control block. Shorter format buffers, in turn, take less time to process and require less space in the internal format buffer pool.

Numeric Fields

Numeric fields should be loaded in the format in which they will most often be used. This will minimize the amount of format conversion required.

Fixed-Storage Option

The use of the fixed storage (FI) option normally reduces the processing time of the field but may result in a larger disk storage requirement, particularly if the field is contained within a periodic group. FI fields, like NU fields, should be grouped together wherever possible.

15   Data Access Strategies

This chapter covers the following topics:

Efficient Use of Descriptors

Descriptors are used to select records from a file based on user-specified search criteria and to control a logical sequential read process. The use of descriptors is thus closely related to the access strategy used for a file. Additional disk space and processing overhead are required for each descriptor, particularly those that are updated frequently. The following guidelines may be used in determining the number and type of descriptors to be defined for a file:

- If data in certain fields needs to be resequenced before processing on the field can continue, a collation descriptor can be defined.

- The distribution of values in the descriptor field should be such that the descriptor can be used to select a small percentage of records in the file;

- Additional descriptors should not be defined to further refine search criteria if a reasonably small number of records can be selected using existing descriptors;

- If two or three descriptors are used in combination frequently (for example, area, department, branch), a superdescriptor may be used instead of defining separate descriptors;

- If the selection criterion for a descriptor always involves a range of values, a subdescriptor may be used;

- If the selection criterion for a descriptor never involves the selection of null value, and a large number of null values are possible for the descriptor, the descriptor should be defined with the null-value suppression (NU) option;

- If a field is updated very frequently, it should normally not be defined as a descriptor;

- Files that have a high degree of volatility (large number of additions and deletions) should not contain a large number of descriptors.

Collation Descriptor

A collation descriptor is used to sort (collate) descriptor field values in a special sequence based on a user-supplied algorithm. An alpha or wide field can be defined as a parent field of a "collation" descriptor.

Special collation descriptor user exits are specified using the ADARUN parameter CDXnn (CDX01 through CDX08). The user exits are used encode the collation descriptor value or decode it back to the original field value. Each collation descriptor must be assigned to a user exit, and a single user exit may handle multiple collation descriptors.

Superdescriptor

A superdescriptor is a descriptor created from a combination of up to 20 fields (or portions of fields). The fields from which a superdescriptor is derived may or may not be descriptors. Super-descriptors are more efficient than combinations of ordinary descriptors when the search criteria involve a combination of values. This is because Adabas accesses one inverted list instead of several and does not have to `AND' several ISN lists to produce the final list of qualifying records. Superdescriptors can also be used in the same manner as ordinary descriptors to control the logical sequence in which a file is read.

The values for search criteria that use superdescriptors must be provided in the format of the superdescriptor (binary for superdescriptors derived from all numeric fields, otherwise alphanumeric). If the superdescriptor format is binary, the input of the search value using an interactive query or report facility such as Natural may be difficult.

Subdescriptor

A descriptor that is derived from a portion of a field is called a subdescriptor. The field used to derive the subdescriptor may or may not be a descriptor. If a search criteria involves a range of values that is contained in the first `n' bytes of an alphanumeric field or the last `n' bytes of a numeric field, a subdescriptor may be defined from only the relevant bytes of the field. Using a subdescriptor allows the search criterion to be represented as a single value rather than a range. This results in more efficient searching, since Adabas does not need to merge intermediate ISN lists; the merged list already exists.

For example, assume an alphanumeric field AREA of 8 bytes, the first 3 of which represent the region and the last 5 the department. If only records for region `111' are desired, a search criterion of `AREA = 11100000 thru 11199999' would be required without a subdescriptor. If the first three bytes of AREA were defined as a subdescriptor, a search criterion equal to `REGION = 111' can be specified.

Phonetic Descriptor

A phonetic descriptor may be defined to perform phonetic searches. Using a phonetic descriptor in a Find command returns all the records that contain similar phonetic values. The phonetic value of a descriptor is based on the first 20 bytes of the field value with only alphabetic values being considered (numeric values, special characters and blanks are ignored).

Hyperdescriptor

The hyperdescriptor option enables descriptor values to be generated based on a user-supplied algorithm. Up to 31 different hyperdescriptors can be defined for a single physical Adabas database. Each hyperdescriptor must be named by an appropriate HEXnn ADARUN statement parameter in the job where it is used.

Hyperdescriptors can be used to implement n-component superdescriptors, derived keys, or other key constructs. For more information about hyperdescriptors, see the documentation on User and Hyperexits, as well as the ADACMP utility description in the Adabas Utilities documentation.

File Coupling

Using a single Find command, file coupling allows the selection of records from one file that are related (coupled) to records containing specified values in a second file. For example, assume two files, CUSTOMER and ORDERS, that contain customer and order information, respectively. Each file contains the descriptor CUSTOMER_NUMBER, which is used as the basis for relating (coupling) the files.

Physical Coupling

The files are physically coupled using the ADAINV utility, which creates a pair of additional indices in the inverted list indicating which records in the CUSTOMER file are related (coupled) to records in the ORDERS file (that is, have the same customer number) and vice versa. Once the files have been coupled, a single Find command containing descriptors from either file may be constructed, for example:

```
FIND CUSTOMER WITH NAME = JOHNSON
          AND COUPLED TO ORDERS
          WITH ORDER-MONTH = JANUARY
```

Physical coupling may be useful for information retrieval systems in which file volatility is very low, or the additional overhead of the coupling lists is deemed insignificant compared with the ease with which queries may be formulated. It may also be useful for small files which are primarily query-oriented.

Physical coupling may involve a considerable amount of additional overhead if the files involved are frequently updated. The coupling lists must be updated if a record in either of the files is added or deleted, or if the descriptor used as the basis for the coupling is updated in either file.

Physical coupling requires additional disk space for the storage of the coupling indices. The space required depends on the number of records that are related (coupled). The best case is where the coupling descriptor is a unique key for one of the files. This means that only a few records in one file will be coupled to a given record in the other file. The worst case is when a many-to-many relationship exists between the files. This will result in a large number of records being coupled to other records in both files.

A descriptor used as the basis for coupling should normally be defined with the null suppression option so that records containing a null value are not included in the coupling indices.

See the Adabas Utilities documentation, the ADAINV utility, for additional information on the use of coupling.

Logical Coupling

A multifile query may also be performed by specifying the field to be used for interfile linkage in the search criteria. This feature is called logical coupling and does not require the files to be physically coupled.

Although this technique requires read commands, it is normally more efficient if the coupling descriptor is volatile because it does not require any physical coupling lists. It should also be noted that the user program need only specify the search criteria and the field to be used for the soft-coupling link. *Adabas performs all necessary search, read and internal list matching operations.*

User-Assigned ISNs

The user has the option of assigning the ISN of each record in a file rather than having this done by Adabas. This technique permits later data retrieval using the ISN directly rather than using the inverted list technique. This requires that the user develop his own method for the assigning a unique ISN to each record. The resulting ISNs must be within the range of the MINISN and MAXISN parameter values specified by the ADALOD utility when the file is loaded.

Using the ISN as a Descriptor

The user may store the ISN of related records in another record in order to be able to read the related records directly without using the Inverted Lists.

For example, assume an application which needs to read an order record and then find and read all customer records for the order. If the ISN of each customer record (for more than one customer per order, a multiple-value field could be used) were stored in the order record, the customer records could be read directly since the ISN is available in the order record.

Storing the customer record ISNs avoids having to issue a FIND command to the customer file to determine the customer records for the order. This technique requires that the field containing the ISNs of the customer records be established and maintained in the order record, and assumes that the ISN assignment in the customer file will not be changed by a file unload and reload operation.

ADAM Usage

The Adabas direct access method (ADAM) facility permits the retrieval of records directly from Data Storage without access to the inverted lists. The Data Storage block number in which a record is located is calculated using a randomizing algorithm based on the ADAM key of the record. The use of ADAM is completely transparent to application programs and query and report writer facilities.

The ADAM key of each record must be a unique value. The ISN of a record may also be used as the ADAM key.

While accessing ADAM files is significantly faster, adding new records to and loading of ADAM files is slower than for standard files because successive new records will not generally be stored in the same block.

If an ADAM file is to be processed both randomly and in a given logical sequence, the logical sequential processing may be optimized by using the bit truncation feature of the ADALOD utility. This feature permits the truncation of a user-specified number of bits from the rightmost portion of each ADAM key value prior to its usage as input to the randomizing algorithm. This will cause records of keys with similar leftmost values to be stored in the same Data Storage block.

It is important not to truncate too many bits, however, as this may increase the number of overflow records and degrade random access performance. The reason is, overflow records which cannot be stored in the blocks located using the ADAM key are stored elsewhere using the standard inverted list process; overflow records must also be located using the inverted list. The only other way to minimize overflow is to specify a relatively large file and padding factor size.

ADAM will generally use an average of 1.2 to 1.5 I/O operations (including an average of overflow records stored under Associator control in other blocks of the file), rather than the three to four I/O operations required to retrieve a record using the inverted lists. Overflow records are also retrieved using normal Associator inverted list references.

The variable factors of an ADAM file that affect performance are, therefore, the amount of disk space available (the more space available, the fewer the overflow records which need to be located with an inverted list), the number of bits truncated from the ADAM key, and the amount of record adding and update activity. The ADAMER utility may be used to determine the average number of I/O operations for various combinations of disk space and bit truncation. See the Adabas Utilities documentation for additional information.

16   Disk Space Usage

The efficient use of disk space is especially important in a database environment since

- sharing data between several users, possibly concurrently and in different combinations, normally requires that a large proportion of an organization's data be stored online; and

- some applications require extremely large amounts of data.

Decisions concerning the efficient usage of disk space must be made while considering other objectives of the system (performance, flexibility, ease of use). This section discusses the techniques and considerations involved in performing trade-offs between these objectives and the efficient usage of disk space.

This chapter covers the following topics:

Data Compression

Each field may be defined to Adabas with one of three compression options:

- Fixed storage (FI), in which the field is not compressed at all. One-byte fields that are always filled (for example, "gender" in a personnel record) and alphanumeric or numeric fields with full values ("personnel number") should always be specified as fixed (FI) fields.

- Ordinary compression (the default) which causes Adabas to remove trailing blanks from alphanumeric fields and leading zeros from numeric fields;

- Null-value suppression, which includes ordinary compression and in addition suppresses the null value for a field. Adjacent null value fields are combined into a single value.

The following table illustrates how various values of a five-byte alphanumeric field are stored using each compression option.

Field Value	Fixed Storage	Ordinary Compression	Null-Value Suppression
ABCbb	ABCbb (5 bytes)	4ABC (4 bytes)	4ABC (4 bytes)
ABCDb	ABCDb (5 bytes)	5ABCD (5 bytes)	5ABCD (5 bytes)
ABCDE	ABCDE (5 bytes)	6ABCDE (6 bytes)	6ABCDE (6 bytes)
bbbbb	bbbbb (5 bytes)	2b (2 bytes)	* (1 byte)
X	X (1 byte)	2X (2 bytes)	2X (2 bytes)

The number preceding each stored value is an inclusive length byte (not used for FI fields). The asterisk shown under null-value suppression indicates a suppressed field count. This is a one-byte field which indicates the number of consecutive empty (suppressed) fields present at this point in the record. This field can represent up to 63 suppressed fields.

The compression options chosen also affect the creation of the inverted list for the field (if it is a descriptor) and the processing time needed for compression and decompression of the field.

Fixed Storage

Fixed storage indicates that no compression is to be performed on the field. The field is stored according to its standard length with no length byte. Fixed storage should be specified for small one- or two-byte fields that are rarely null, and for fields for which little or no compression is possible. Refer to the Adabas Utilities documentation the ADACMP utility, for restrictions related to the use of FI fields.

Ordinary Compression

Ordinary compression results in the removal of trailing blanks from alphanumeric fields and leading zeros from numeric fields. Ordinary compression will result in a saving in disk space if at least two bytes of trailing blanks or leading zeros are removed. For two-byte fields, however, there is no savings, and for one-byte fields, adding the length byte actually doubles the needed space. Such fields, and fields that rarely have leading or trailing zeros or blanks, should be defined with the fixed storage (FI) option to prevent compression.

Null-Value Suppression

If null-value suppression (NU) is specified for a field, and the field value is null, a one-byte empty field indicator is stored instead of the length byte and the compressed null value (see *Data Compression*). This empty field indicator specifies the number of consecutive suppressed fields that contain null values at this point in the record. It is, therefore, advantageous to physically position fields which are frequently empty next to one another in the record, and to define each with the null-value suppression option.

An NU field that is also defined as a descriptor is not included in the inverted lists if it contains a null value. This means that a find command referring to that descriptor will not recognize qualifying descriptor records that contain a null value.

This applies also to subdescriptors and superdescriptors derived from a field that is defined with null-value suppression. No entry will be made for a subdescriptor if the bytes of the field from which it is derived contain a null value and the field is defined with the null-value suppression (NU) option. No entry will be made for a superdescriptor if any of the fields from which it is derived is an NU field containing a null value.

Therefore, if there is a need to search on a descriptor for null values, or to read records containing a null value in descriptor sequence-for example, to control logical sequential reading or sorting-then the descriptor field should not be defined with the NU option.

Null-value suppression is normally recommended for multiple-value fields and fields within periodic groups in order to reduce the amount of disk space required and the internal processing requirements of these types of fields. The updating of such fields varies according to the compression option used.

If a multiple-value field value defined with the NU option is updated with a null value, all values to the right are shifted left and the value count is reduced accordingly. If all the fields of a periodic group are defined with the NU option, and the entire group is updated to a null value, the occurrence count will be reduced only if the occurrence updated is the highest (last) occurrence. For detailed information on the updating of multiple-value fields and periodic groups, see the Adabas Utilities documentation ADACMP utility, and the Adabas Command Reference documentation A1/A4 and N1/N2 commands.

Forward Index Compression

The forward (or `front' or `prefix') index compression feature saves index space by removing redundant prefix information from index values. The benefits are less disk space used, possibly fewer index levels used, fewer index I/O operations, and therefore greater overall throughput. The buffer pool becomes more effective because the same amount of index information occupies less space. Commands such as L3, L9, or S2, which sequentially traverse the index, become faster and the smaller index size reduces the elapsed time for Adabas utilities that read or modify the index.

Within one index block, the first value is stored in full length. For all subsequent values, the prefix that is common with the predecessor is compressed. An index value is represented by

```
<l,p,value>
```

where

l p is the number of bytes that are identical to the prefix of the preceding value; and

l is the exclusive length of the remaining value including the p-byte.

For example:

Decompressed	Compressed
ABCDE	6 0 ABCDE
ABCDEF	2 5 F
ABCGGG	4 3 GGG
ABCGGH	2 5 H

Index compression is not affected by the format of a descriptor. It functions as well for PE-option and multiclient descriptors.

The maximum possible length of a compressed index value occurs for an alphanumeric value in a periodic group:

```
253 bytes for the proper value if no bytes are compressed
1 byte for the PE index
1 byte for the p-byte.
```

The total exclusive length can thus be stored in a single byte.

Adabas implements forward index compression at the file level. When loading a file (ADALOD), an option is provided to compress index values for that file or not. The option can be changed by reordering the file (ADAORD).

Adabas also provides the option of compressing all index values for a database as a whole. In this case, specific files can be set differently; the file-level setting overrides the database setting.

The decision to compress index values or not is based on the similarity of index values and the size of the file:

- the more similar the index values, the better the compression results.

- small files are not good candidates because the absolute amount of space saved would be small whereas large files are good candidates for index compression.

Even in a worst case scenario where the index values for a file do not compress well, a compressed index will not require more index blocks than an uncompressed index.

Padding Factors

A large amount of record update activity may result in a considerable amount of record migration, i.e., removal of the record from its current block to another block in which sufficient space for the expanded record is available. Record migration may be considerably reduced by defining a larger padding factor for Data Storage for the file when it is loaded. The padding factor represents the percentage of each physical block which is to be reserved for record expansion.

The padding area is not used during file loading or when adding new records to a file (this is not applicable for an ADAM file, since the padding factor is used if necessary to store records into their calculated Data Storage block). A large padding factor should not be used if only a small percentage of the records is likely to expand, since the padding area of all the blocks in which nonexpanding records are located would be wasted.

If a large amount of record update/addition is to be performed in which a large number of new values must be inserted within the current value range of one or more descriptors, a considerable amount of migration may also occur within the Associator. This may be reduced by assigning a larger padding factor for the Associator.

The disadvantages of a large padding factor are a larger disk space requirement (fewer records or entries per block) and possible degradation of sequential processing since more physical blocks will have to be read.

Padding factors are specified when a file is loaded, but can be changed when executing the ADADBS MODFCB function or the ADAORD utility for the file or database.

17 **Adabas Security**

This section describes general considerations for database security and introduces the security facilities provided by Adabas and the Adabas subsystems. Detailed information about the facilities discussed in this section may be found in other parts of this documentation and in the Adabas Security documentation.

This chapter covers the following topics:

Security Planning

Effective security measures must take account of the following:

- A system is only as secure as its weakest component. This may be a non-DP area of the system: for example, failure to properly secure printed listings;

- It is rarely possible to design a "foolproof" system. A security system will probably be breached if the gain from doing so is likely to exceed the cost;

- Security can be expensive. Costs include inconvenience, machine resources, and the time required to coordinate the planning of security measures and monitor their effectiveness.

The cost of security measures is usually much easier to quantify than the risk or cost of a security violation. The calculation may, however, be complicated by the fact that some security measures offer benefits outside the area of security. The cost of a security violation depends on the nature of the violation. Possible types of cost include

- loss of time while the violation is being corrected;

- penalties under privacy legislation, breach of contracts, and so on;

- damage to relationships with customers, suppliers, employees, and so on.

Password Security

Password security allows the DBA to control a user's use of the database by

- restricting the user to certain files;

- specifying for each file whether the user can access and update, or access only;

- preventing the user from accessing or updating certain fields while allowing access or update of other fields in the same file;

- restricting the user's view of the file to records that contain specified field values (for example, department code).

The DBA can assign a security level to each file and each field within a file. In the following table, *x*/*y* indicates the access/update security level. The value 0/0 indicates no security.

File	Fields	
1 (2/3)	AA (0/0)	BB (4/5)
2 (6/7)	LL (6/7)	MM (6/9)
3 (4/5)	XX (4/5)	YY (4/5)
4 (0/0)	FF (0/0)	GG (0/15)

A user must supply an appropriate password to access/update a secured file. In the following table, *x*/*y* indicates the password access/update authorization level.

	Passwords	
	ALPHA	BETA
File 1	2/3	4/5
File 2	0/0	6/7
File 3	4/5	0/0

Assuming the files, fields, and passwords shown in the above tables, the following statements are true:

- Password ALPHA
    - can access and update field AA in file 1, but not field BB;
    - can access and update all fields in file 3;
    - cannot access or update file 2.
- Password BETA
    - can access and update all fields in file 1;
    - can access all the fields in file 2 and can update field LL, but not field MM;
    - cannot access or update file 3.
- No password is required to access any field in file 4, or to update field FF.
- Field GG in file 4 can be read only. Its update security level is 15 and the highest possible authorization level is 14.

If password BETA can access a field that password ALPHA cannot, then password BETA can also access all the fields in the same file that password ALPHA can access. There is no way in which ALPHA can be authorized to access field AA but not field BB and password BETA to access BB but not AA. The same restriction applies to update (although not necessarily to the same combinations of fields or to the advantage of the same password). ALPHA could be permitted to update all the fields which BETA can update and some others which BETA cannot update.

This restriction does not apply to file-level security. For example, ALPHA can use file 3 but not file 2, and BETA can use file 2 but not file 3. When a record is being added to a file, Adabas only checks the update security level on those fields for which the user is supplying values. For example, the password ALPHA could be used to add a record to file 1 provided that no value was specified for field BB. This could represent the situation where, for example, a customer record is only to be created with a zero balance. For record deletion, the password provided must have an authorization level equal to or greater than the highest update security level present in the file. For example, an update authorization level of 9 is required to delete a record from file 2, and, it is not possible to delete records from file 4.

Security by Value

It is also possible to limit access/update fields within a file based on the contents of the field in the file. See the Adabas Security documentation for more information.

Ciphering

Adabas is able to cipher (encrypt) records when they are initially loaded into a file or when records are being added to a file. Ciphering makes it extremely difficult to read the contents of a copy of the database obtained from a physical dump of the disk on which the database is contained. Ciphering applies to the records stored in Data Storage only. No ciphering is performed for the Associator.

Adabas SAF Security

Adabas SAF Security, a selectable unit, can be used with Software AG's Com-plete and with the following non-Software AG security environments:

- CA-ACF2 (Computer Associates);
- CA-Top Secret (Computer Associates);
- RACF (IBM Corporation)

For more information about Adabas SAF Security, contact your Software AG representative.

Natural andAdabas Online System Security

The Natural Security system may also be used to provide extensive security provisions for Adabas/Natural users. See the Natural Security documentation for additional information.

Access to the DBA facility Adabas Online System (AOS) can also be restricted. AOS Security requires Natural Security as a prerequisite.

18 Recovery/Restart Design

This chapter discusses the design aspects of database recovery/restart. Proper recovery/restart planning is an important part of the design of the system, particularly in a database environment. Although Adabas provides facilities to perform both restart and recovery, the functions must be considered separately.

This chapter covers the following topics:

Adabas Recovery

Recovery of database integrity has the highest priority; if a database transaction fails or must be cancelled, the effects of the transaction must be removed and the database must be restored to its exact condition before the transaction began.

The standard Adabas system provides transaction logic (called "ET logic"), extensive checkpoint/logging facilities, and transaction-reversing "backout" processing to ensure database integrity.

Restarting the database following a system failure means reconstructing the task sequence from a saved level before the failure, up to and including the step at which the failure occurred-including, if possible, successfully completing the interrupted operation and then continuing normal database operation. Adabas provides a recovery aid that reconstructs a recovery job stream to recover the database.

Recoverability is often an implied objective. Everyone assumes that whatever happens, the system can be systematically recovered and restarted. There are, however, specific facts to be determined about the level of recovery needed by the various users of the system. Recoverability is an area where the DBA needs to take the initiative and establish necessary facts. Initially, each potential user of the system should be questioned concerning his recovery/restart requirements. The most important considerations are

- how long the user can manage without the system;

- how long each phase can be delayed;

- what manual procedures, if any, the user has for checking input/output and how long these take;

- what special procedures, if any, need to be performed to ensure that data integrity has been maintained in a recovery/restart situation.

Planning and Incorporating Recoverability

Once the recovery/restart requirements have been established, the DBA can proceed to plan the measures necessary to meet these requirements. The methodology provided in this section may be used as a basic guideline.

1. A determination should be made as to the level and degree to which data is shared by the various users of the system.

2. The recovery parameters for the system should be established. This includes a predicted/actual breakdown rate, an average delay and items affected, and items subject to security and audit.

3. A determination should be made as to what, if any, auditing procedures are to be included in the system.

4. An outline containing recovery design points should be prepared. Information in this outline should include

   - validation planning. Validation on data should be performed as close as possible to its point of input to the system. Intermediate updates to data sharing the record with the input will make recovery more difficult and costly;

   - dumps (back-up copies) of the database or selected files;

   - user and Adabas checkpoints;

   - use of ET logic, exclusive file control, ET data;

   - audit procedures.

5. Operations personnel should be consulted to determine if all resources required for recovery/restart can be made available if and when they are needed.

6. The final recovery design should be documented and reviewed with users, operations personnel, and any others involved with the system.

Matching Requirements and Facilities

Once the general recovery requirements have been designed, the next step is to select the relevant Adabas and non-Adabas facilities to be used to implement recovery/restart. The following sections describe the Adabas facilities related to recovery/restart.

Transaction Recovery

Almost all online update systems and many batch update programs process streams of input transactions which have the following characteristics:

- The transaction requires the program to retrieve and add, update, and/or delete only a few records. For example, an order entry program may retrieve the customer and product records for each order, add the order and order item data to the database, and perhaps update the quantity-on-order field of the product record.

- The program needs exclusive control of the records it uses from the start of the transaction to the end, but can release them for other users to update or delete once the transaction is complete.

- A transaction must never be left incomplete; that is, if it requires two records to be updated, either both or neither must be changed.

End Transaction (ET) Command

The use of the Adabas ET command

- ensures that all the adds, updates, and/or deletes performed by a completed transaction are applied to the database;

- ensures that all the effects of a transaction which is interrupted by a total or partial system failure are removed from the database;

- allows the program to store up to 2000 bytes of user-defined restart data (ET data) in an Adabas system file. This data may be retrieved on restart with the Adabas OP or RE commands. The restart data can be examined by the program or TP terminal user to decide where to resume operation;

- releases all records placed in hold status while processing the transaction.

Close (CL) Command

The Adabas CL command can be used to update the user's current ET data (for example, to set a user-defined "job completed" flag). Refer to the section *User Restart Data* for more information.

Reading ET Data

After a user restart or at the start of a new user or Adabas session, ET data can be retrieved with the OP command. The OP command requires a user ID, which Adabas uses to locate the ET data, and a command option to read ET data.

The RE command can also be used to read ET data for the current or a specified user; for example, when supervising an online update operation.

System or Transaction Failure

The autobackout routine is automatically invoked at the beginning of every Adabas session. If a session terminates abnormally, the autobackout routine removes the effects of all interrupted transactions from the database up to the most recent ET. If an individual transaction is interrupted, Adabas automatically removes any changes the transaction has made to the database. Each application program can explicitly back out its current transaction by issuing the Adabas BT command.

Limitations of Adabas Transaction Recovery

The transaction recovery facility recovers only the contents of the database. It does not recover TP message sequences, reposition non-Adabas files, or save the status of the user program.

It is not possible to back out the effects of a specific user's transactions because other users may have performed subsequent transactions using the records added or updated by the first user.

Adabas Checkpoint Commands

Some programs cannot conveniently use ET commands because

- the program would have to hold large numbers of records for the duration of each transaction. This would increase the possibility of a deadlock situation (Adabas automatically resolves such situations by backing out the transaction of one of the two users after a user-defined time has elapsed, but a significant amount of transaction reprocessing could still result), and a very large Adabas hold queue would have to be established and maintained;

- the program may process long lists of records found by complex searches; restarting part of the way through such a list may be difficult.

Such programs can use the Adabas checkpoint command (C1) to establish a point to which the file or files the program is updating can be restored if necessary.

Exclusive File Control

A user can request exclusive update control of one or more Adabas files. Exclusive control is requested with the OP command and will be given only if the file is not currently being updated by another user. Once exclusive control is assigned to a user, other users may read but not update the file. Programs that read and/or update long sequences of records, either in logical sequence or as a result of searches, may use exclusive control to prevent other users from updating the records used. This avoids the need for placing each record in "hold" status.

Checkpointing Exclusively Controlled Files

Exclusive control users may or may not use ET commands. If ET commands are not used, checkpoints can be taken by issuing a C1 command.

System or Program Failure

In the event of a system or program failure, the file or files being updated under exclusive control may be restored using the BACKOUT function of the ADARES utility. This utility is not automatically invoked and requires the Adabas data protection log as input. This procedure is not necessary if the user uses ET commands (see the section *Transaction Recovery*).

Limitations of Exclusive File Control

The following limitations apply to exclusive file control:

- Recovery to the last checkpoint is not automatic, and the data protection log in use when the failure occurred is required for the recovery process. This does not apply if the user issues ET commands.

- In a restart situation following a system failure, Adabas does not check nor prevent other users from updating files which were being updated under exclusive control at the time of the system interruption.

User Restart Data

The Adabas ET and CL commands provide an option of storing up to 2000 bytes of user data in an Adabas system file. One record of user data is maintained for each user. This record is overwritten each time new user data is provided by the user. The data is maintained from session to session only if the user provides a user identification (user ID) with the OP command.

The primary purpose of user data is to enable programs to be self-restarting and to check that recovery procedures have been properly carried out. The type of information which may be useful as user data includes the following:

■ The *date and time* of the original program run and the time of last update. This will permit the program to send a suitable message to a terminal user, console operator, or printer to allow the user and/or operator to check that recovery and restart procedures have operated correctly. In particular, it will allow terminal users to see if any work has to be rerun after a serious overnight failure of which they were not aware.

■ The *date of collection* of the input data.

■ *Batch numbers*. This will enable supervisory staff to identify and allocate any work that has to be reentered from terminals.

■ *Identifying data*. This data can be a way for the program to "decide" where to restart. For example, a program driven by a logical sequential scan needs to know the key value at which to resume.

■ *Transaction number/input record position* . This may allow an interactive user or batch program to locate the starting point with the minimum of effort. Although Adabas returns a transaction sequence number for each transaction, the user also may want to maintain a sequence number because

  ■ after a restart, the Adabas sequence number is reset;

  ■ if transactions vary greatly in complexity, there may not be a simple relationship between the Adabas transaction sequence number and the position of the next input record or document;

  ■ if a transaction is backed out by the program because of an input error, Adabas does not know whether the transaction will be reentered immediately (it may have been a simple keying error) or rejected for later correction (if there was a basic error in the input document or record);

■ *Other descriptive or intermediate data*; for example, totals to be carried forward, page numbers and headings of reports, run statistics.

■ *Job/batch completed flag*. The system may fail after all processing has been completed but before the operator or user has been notified. In this case, the operator should restart the program which will be able to check this flag without having to run through to the end of the input. The same considerations apply to batches of documents entered from terminals.

■ *Last job/program name*. If several programs must update the database in a fixed sequence, they may share the same user ID and use user data to check that the sequence is maintained.

A user's own data can be read with either the OP or RE command. User data for another user can be read by using the RE command and specifying the other user's ID. User data for all users can be read in logical sequential order using the RE command with a command option; in this case, user IDs are not specified.

19 The Adabas Recovery Aid

When a system failure disrupts database operation, the Adabas Recovery Aid can create a job stream that reconstructs the database to the point of failure.

The Recovery Aid combines the protection log (PLOG) and the archived database status from previous ADASAV operations with its own recovery log (RLOG) information to reconstruct the job sequence. The result is a reconstructed job statement string (recovery job stream) that is placed in a specially named output dataset.

The two major parts of the Adabas Recovery Aid are the recovery log (RLOG) and the recovery aid utility ADARAI. The RLOG is formatted like other Adabas files, using ADAFRM, and then defined with the ADARAI utility.

The DBA must run the Recovery Aid utility, ADARAI, to

- define the RLOG and set up the Recovery Aid environment;
- display current RLOG information;
- create the recovery job stream.

This chapter covers the following topics:

The Recovery Log (RLOG)

The recovery log (RLOG) records the essential information that, when combined with the PLOG, is used by the ADARAI utility's RECOVER function to rebuild a job stream to recover and restore the database status up to the point of failure.

The RLOG information is grouped in "generations", where each generation comprises the database activity between consecutive ADASAV SAVE, RESTORE (database) or RESTORE GCB operations. The RLOG holds a minimum of four consecutive generations, up to a maximum value specified when the RLOG is activated; the maximum is 32. If RLOG space is not sufficient to hold the specified number of generations, the oldest generation is overwritten with the newest in "wraparound" fashion.

The RLOG file is formatted like other database components by running the ADAFRM utility (SIZE parameter), and then defined using the PREPARE function of the Recovery Aid ADARAI utility (with the RLOGSIZE parameter). The space required for the RLOG file is approximately 10 cylinders of 3380 or equivalent device space.

The ADARAI PREPARE function must be performed just before the ADASAV SAVE run that begins the first generation to be logged. After ADARAI PREPARE is executed, all subsequent nucleus and utility jobs that update the database must specify the RLOG file. Of course, the RLOG file can be included in any or all job streams, if desired.

The RLOG file job statement should be similar to the following:

```
//DDRLOGR1 DD DISP=SHR,DSN=... .RLOGR1
```

Starting the Recovery Aid

The activity of the Recovery Aid and RLOG logging begins when the first ADASAV SAVE/RE-STORE database or RESTORE GCB function is executed following ADARAI PREPARE.

All activity between the first and second ADASAV SAVE/RESTORE database and/or RESTORE GCB operations following the ADARAI PREPARE operation belongs to the first generation. When viewing generations with the ADARAI utility's LIST function, generations are numbered relatively in ascending order beginning with the oldest generation.

For more detailed information on setting up the Recovery Aid, see *Restart and Recovery* in the Adabas Operations documentation and the ADARAI utility description in the Adabas Utilities documentation.

20 Multiclient Support

The Adabas multiclient feature stores records for multiple users or groups of users in a single Adabas file. This feature is specified at the file level. It divides the physical file into multiple logical files by attaching an owner ID to each record. Each user can access only the subset of records that is associated with the user's owner ID. The file is still maintained as a single physical Adabas file.

The Adabas nucleus handles all database requests to multiclient files.

This chapter covers the following topics:

The Owner Concept

Each record in a multiclient file has a specific owner, which is identified by an internal owner ID attached to each record (for any installed external security package such as RACF or CA-Top Secret, a user is still identified by either Natural ETID or LOGON ID). The owner ID is assigned to a user ID. A user ID can have only one owner ID, but an owner ID can belong to more than one user.

The following table shows examples of the ETID/owner ID relationship.

ETID	Owner ID	Description
USER1	1	More than one user can use the same owner ID.
USER2	1	Here, USER1, USER2 and USER3 share the same owner ID and therefore the same records.
USER3	1	
...		
USER4	2	

The relationship between the user ID and the owner ID is stored in the profile table in the Adabas checkpoint file. The DBA maintains the profile table using Adabas Online System/Basic Services (AOS), a prerequisite for the multiclient feature.

The relation between user ID and owner ID is 1:1 or n:1; either a single user or group of users can be assigned to one owner ID. Record isolation is always performed on the owner ID level.

The owner ID has a fixed length of up to 8 bytes (alphanumeric). The length is defined by the user during file creation; it can be changed only by unloading and reloading the multiclient file. Each owner ID must be less than or equal to the length assigned for the file; otherwise, a nonzero response code occurs. To avoid wasting space, make the owner ID no larger than necessary.

The following tables show an example of owner isolation for a group of eight file records.

ISN	Owner ID	Record	Discussion
1	1	.data.	Example for a physical Adabas file with records owned by different users
2	2	.data.	
3	1	.data.	
4	3	.data.	
5	2	.data.	
6	3	.data.	
7		- no data -	
8	1	.data.	

ISN	Record	ISN	Record
1	.data.	1	- no data -
2	- no data -	2	.data.
3	.data.	3	- no data -
4	- no data -	4	- no data -
5	- no data -	5	.data.
6	- no data -	6	- no data -
7	- no data -	7	- no data -
8	.data.	8	- no data -
File as seen by a user with an owner ID=1		File as seen by a user with an owner ID=2	

Super Users

A *super-user* owner ID provides access to all records in a multiclient file. A super-user owner ID begins with an asterisk (*). Adabas allows users with such an owner ID to "match" with any other owner ID, allowing the user to read all records in a file. More than one super-user owner IDs, each beginning with "*" and allowing identical privileges, can be defined for a multiclient file.

A super-user owner ID applies only to Lx read commands and nondescriptor search (Sx) commands. Descriptor search commands by a super user return only the records having the super user's owner ID. Data records or index values stored by a super user are labeled with the super user's owner ID.

Note: If a super-user issues an L3 or L9 command, the value start option is ignored; that is, Adabas always starts at the very beginning of the specific descriptor.

Program Compatibility

No changes to an existing application program are needed to use it in a multiclient environment; however, a user ID must be supplied in the Additions 1 field of the Adabas control block of each open (OP) call made by a user who addresses a multiclient file. This allows Adabas to retrieve the owner ID from the checkpoint file. Otherwise, the application program neither "knows nor cares" whether a multiclient file or a standard Adabas file is being accessed.

Support for Soft Coupling

Multiclient support is provided for soft coupling.

Data and Index Structures

The data and index structures of a multiclient file differ from those of standard Adabas files.

Data Storage

A Data Storage (DATA) record in a standard file has the following structure:
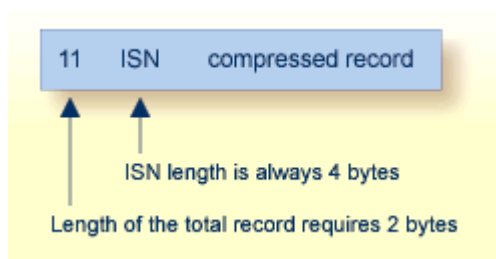
A Data Storage record in a multiclient file has the following structure:
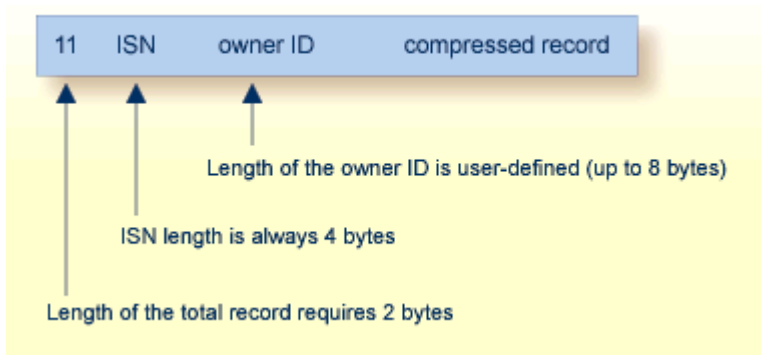
Comparison of Normal and Multiclient Record Formats

Associator

Every normal index and upper index value for a multiclient file is prefixed by the owner ID:
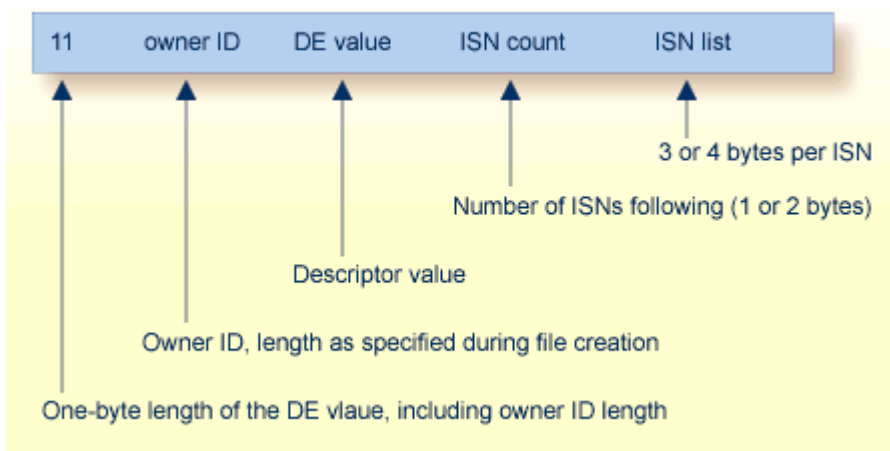
Normal Index Element Structure

The tables below illustrate a multiclient index structure. If a single descriptor value points to more than one Data Storage record, Adabas stores this extended index value only once, followed by the list of ISNs. If the same descriptor value for different owner IDs is to be stored, then multiple entries are made in the index.

ISN	Owner ID	NAME	
1	1	SMITH	The field "NAME" is a descriptor.
2	2	SMITH	
3	1	SMITH	
4	3	JONES	
5	2	JONES	
6	3	HARRIS	
7		- not stored -	
8	1	HARRIS	

Owner ID	DE value	ISN count	ISN list	
1	HARRIS	1	8	This is the index for the descriptor NAME. The sort sequence of values is: owner ID | (DE-value)
1	SMITH	2	1,3	
2	JONES	1	5	
2	SMITH	1	2	
3	HARRIS	1	6	
3	JONES	1	4	

Notes:

1. Every type of descriptor is prefixed by the owner ID: simple descriptors, sub/superdescriptors, phonetic, and hyperdescriptors. The owner ID prefix is not counted as a parent field for super- and hyperdescriptors. The maximum number of parent fields is not affected.

2. The maximum length of a descriptor value, *including* the owner ID, is 253 bytes.

3. A super-user reading index values in L3/L9 sequence gets values in sorting order by owner ID: the values for the lowest owner ID first, then the values for the next higher owner ID, and so on. Values for each owner ID are sorted in ascending order.

Performance Considerations

The multiclient feature causes no added processing overhead for find (S1,S2), read-logical (L3) and histogram (L9) commands. The index structure permits specific record selection, and there is no postselection procedure in the Data Storage.

If the selection is done on the Data Storage, Adabas must read the record and check the owner ID. If the record's owner ID does not match the current user's owner ID, the record is skipped. This might slow down a read-physical (L2) and a read-by-ISN (L1 with I option) command or a non-descriptor search command.

User Profile Table

The owner ID is part of the user's profile record, which is stored in the Adabas profile table. The profile is maintained using the Adabas Online System. See the Adabas Online System documentation for more information.

Possible Adabas Response Codes

Calls to multiclient files can result in the following non-zero Adabas response codes, which indicate that an error has occurred:

Read and Update Operation	If a user tries to read or change a multiclient file's record using an owner ID that does not apply to the record, Adabas returns either response code 3 or 113, depending on the type of read or update operation.
"Add Record" Operation	If a user has an owner ID that is either blank or too long for the owner ID length assigned to the multiclient file, Adabas returns response code 68 if this owner tries to add a new record.
Blank or Missing Owner IDs	A user with a blank or missing owner ID receives response code 3 or 113 when trying to access a multiclient file.

Utility Support for Multiclient Files

In general, multiclient files are transparent to Adabas utility processing. Special functions of the ADALOD and ADAULD utilities support the migration of an application from a standard to a multiclient environment.

The ADALOD Utility LOAD Function

Two ADALOD LOAD parameters LOWNERID and ETID support multiclient files. The parameters work together to define owner IDs and determine whether a file is a multiclient file.

LOWNERID specifies the length of the internal owner ID values assigned to each record for multiclient files.

Valid length values are 0-8. In combination with the ETID parameter, the LOWNERID parameter can be used to reload a standard file as a multiclient file, change the length of the owner ID for the file, or remove the owner ID from the records of a file.

If the LOWNERID parameter is not specified, the length of the owner ID for the input file (if any) remains the same.

ETID assigns a new owner ID to all records being loaded into a multiclient file, and must be specified if the input file contains no owner IDs; that is, the input file was not unloaded from a multiclient source file.

The following table illustrates the effects of LOWNERID and ETID settings.

LOWNERID Parameter Setting:	Owner ID Length in Input File:	
	0	2
0	Keep as nonmulticlient file	Convert into a nonmulticlient file
1	Set up multiclient file (ETID)	Decrease owner ID length
2	Set up multiclient file (ETID)	Keep owner ID length
3	Set up multiclient file (ETID)	Increase owner ID length
(LOWNERID not specified)	Keep as a nonmulticlient file	Keep as a multiclient file

Where this table indicates "(ETID)" in the "owner ID length...0" column, the ETID parameter must specify the user ID identifying the owner of the records being loaded. The owner ID assigned to the records is taken from the user profile of the specified user ID. In the "owner ID length...2" column the ETID parameter is optional; if ETID is omitted, the loaded records keep their original owner IDs.

> **Note:** If the ETID parameter is used, the ADALOD utility requires an active nucleus. The nucleus will translate the ETID value into the internal owner ID value.

The ADALOD Utility UPDATE Function

When executing the UPDATE function, ADALOD keeps the owner ID length previously defined for the file being updated. The owner IDs of the records being added are adjusted to the owner ID length defined for the file. The owner IDs of the loaded records or of any new records must fit into the existing owner ID space.

Examples:

```
ADALOD LOAD FILE=20,LOWNERID=2,NUMREC=0
```

-creates file 20 as a multiclient file. The length of the internal owner ID is two bytes, but no actual owner ID is specified. No records are actually loaded in the file (NUMREC=0).

```
ADALOD LOAD FILE=20,LOWNERID=2,ETID=USER1
```

-creates file 20 as a multiclient file and loads all supplied records for user USER1. The length of the internal owner ID is two bytes.

```
ADALOD UPDATE FILE=20,ETID=USER2
```

-performs a mass update to add records to file 20, a multiclient file. Loads all the new records for USER2.

The ADAULD Utility

The ADAULD utility unloads records from an Adabas file to a sequential output file. This output file can then be used as input to a subsequent ADALOD operation.

If a multiclient file is unloaded, the output file contains all the unloaded records with their owner IDs. This information can either be retained by the subsequent ADALOD operation, or be over-written with new information by the ADALOD ETID parameter. Any differences in LOWNERID parameter values for the unloaded and newly loaded file are automatically adjusted by ADALOD.

The ETID parameter of ADAULD can be used to restrict UNLOAD processing to only the records owned by the specified user. If the ETID parameter is omitted, all records are unloaded. If the SELCRIT/SELVAL parameters are specified for a multiclient file, the ETID parameter must also be specified.

Example:

```
ADAULD UNLOAD FILE=20,ETID=USER1
```

-unloads all records owned by USER1 in physical sequence.

The ADACMP Utility

The ADACMP utility either compresses user data from a sequential input file into the Adabas internal structure, or decompresses Adabas data to a sequential user file. The COMPRESS function makes no distinction between standard and multiclient files, processing both in exactly same way. The DECOMPRESS function can decompress records selectively if the INFILE parameter specifies a multiclient file and a valid ETID value is specified.

The DECOMPRESS function skips the owner ID, if present. The output of a DECOMPRESS operation on a multiclient file contains neither owner ID nor any ETID information.

If the INFILE parameter specifies a multiclient file for the DECOMPRESS function, decompression can be limited to records for a specific user using the ETID parameter. ADACMP then reads and decompresses records for the specified user. If the ETID parameter is not specified when decompressing a multiclient file, all records in the file are decompressed.

Example:

```
ADACMP  DECOMPRESS  INFILE=20,ETID=USER1
```

-decompresses records which are owned by USER1 from file 20 to a sequential output file.

21 Expanded Files

This chapter covers the following topics:

Overview

An expanded file is a logical file comprising one or more physical component files. Each component file contains records numbered by logical instead of physical ISN numbers. These physical component files must have

- *identical* field definition tables (FDTs); and
- *different* logical ISN ranges defined by the file's MINISN and MAXISN parameters. The ISN ranges cannot overlap.

The component files are chained together in sequence according to their ascending ISN ranges. The file with the lowest ISN range is called the "anchor" file; its file number is the number of the whole expanded file.

An expanded file can comprise up to 128 component files; it cannot exceed 4,294,967,294 records. An Adabas component file with 3-byte ISNs can contain a maximum of 16,777,215 records; a component file with 4-byte ISNs can contain 4,294,967,294 records.

> **Note:** Now that Adabas supports larger file sizes and a greater number of Adabas physical files and databases, the need for expanded files has, in most cases, been removed.

Expanded files are supported by the Adabas

- command that processes ISN lists, S8.
- sort commands S2 and S9. Before using this feature, investigate how it will affect database performance and impact users.
- prefetch/multifetch functions, which are enhanced for ET/BT support during expanded file processing.

Defining Expanded Files

Using ADALOD

Expanded files are defined at load time. Each physical component file is loaded separately using the ADALOD LOAD function. For all but the first component file, the ANCHOR parameter must be specified to refer to the anchor file. ADALOD LOAD then performs the following tasks:

- Compares the FDT of the new file to the FDT of the anchor file to ensure that they are the same.

- Checks the new ISN range (MINISN to MAXISN for the new component file) against the ISN ranges of the anchor and all other component files to ensure that there is no duplication;

- Checks for specification of the NOACEXTENSION parameter (Address Converter extensions are not permitted in component files);

- Checks that the MAXRECL parameter of the new component file is equal to that for the existing anchor file. All component files must have the same MAXRECL value.

- Loads the new component file, and;

- Links the new component file into the expanded file chain.

Example:

ADALOD LOAD statements define an expanded file (only the relevant parameters are shown):

```
ADALOD  LOAD FILE=11,NOACEXTENSION
ADALOD       MINISN=1,MAXISN=16000000
ADALOD       ...

ADALOD  LOAD FILE=23,NOACEXTENSION
ADALOD       ANCHOR=11
ADALOD       MINISN=36000001,MAXISN=50000000
ADALOD       ...

ADALOD  LOAD FILE=17,NOACEXTENSION
ADALOD       ANCHOR=11
ADALOD       MINISN=20000001,MAXISN=36000000
```

This example loads file 11 as an expanded file comprising:

 File 11, ISN range: 1-16,000,000
 File 17, ISN range: 20,000,001-36,000,000
 File 23, ISN range: 36,000,001-50,000,000

Using the Online System

An expanded file can also be defined using the Adabas Online System *Define File* function. This function creates a new, empty file that can be specified as an anchor or component file for an expanded file. Existing files can be chained together using the *Expanded File Maintenance* function.

Rules for Defining Expanded Files

1. The NOACEXTENSION parameter must be set to prevent any extension of the Address Converter (i.e., increase to MAXISN) for the specified file.

2. The MINISN parameter must be specified when loading a component file for an expanded file.

3. The file number for the component file can be freely chosen.

4. A single file is loaded as an expanded file when the ANCHOR and FILE parameters specify the same file number.

5. An existing single file which is to be expanded may be referenced as the anchor file when the second component file is loaded. ADALOD then sets NOACEXTENSION for the first file, and makes it the anchor file.

   > **Note:** An anchor file created in this way loses its anchor status when all component files are removed. If necessary, you can insert the file into itself to reestablish its anchor status.

6. The ISN ranges for the component files cannot overlap, but there may be gaps of unused ISNs between file ranges.

7. The component files can be loaded in any sequence.

8. If a new component file is loaded that has an ISN range lower than the range of the current anchor file, the newly loaded file becomes the new anchor file. The ANCHOR parameter of any component file loaded thereafter must refer to the new anchor file.

Inserting a Component File

Component files can be inserted into an expanded file using the ADALOD LOAD function as described in *Defining Expanded Files* or using Adabas Online System.

Using the online system, a new file can be created and inserted into an expanded file using the *Define File* function. A file that already exists can be inserted into an expanded file using the *Insert Component File* function.

Refer to the section *Rules for Defining Expanded Files* on page 57 for possible effects of adding a component file.

Removing a Component File

A component file may be both removed from the expanded file and deleted using the Adabas Online System *Delete File* function. To remove a component file from the expanded file chain without deleting the file, the Adabas Online System *Remove Component File* function can be used.

Refer to the section *Rules for Defining Expanded Files* on page 57 for possible effects of removing a component file.

Deleting Expanded Files

The Adabas Online System *Delete File* function also allows you to delete the complete expanded file; that is, to delete the anchor and all component files. The ADADBS utility's *DELETE* function can also be used to delete the complete expanded file.

Inspecting an Expanded File

In addition to the normal information about individual files, the report produced by the ADAREP utility shows the component file list for each expanded file in the database. The expanded file information itself is also available using the Adabas Online System *Display File* function.

Expanded Files and the Adabas Nucleus

A user call that refers to an expanded file is automatically directed to the appropriate physical component file by the Adabas nucleus. The user or application receives no indication that the selected file is an expanded file.

If the file number in the Adabas control block specifies the component file of an expanded file, the call is interpreted as being for the complete expanded file. Thus, user applications that accessed an existing component file in the past need not be changed if that file is integrated into an expanded file: the calls automatically apply to the complete expanded file. However, for convenience Software AG recommends that calls refer to the anchor file.

If a function performed on an expanded file produces results from more than one component file, those results are combined to produce a single result. For example, an L2 command (read physical sequential) for an expanded file is performed on each component file in sequence, beginning with the anchor file. Upon reaching the end-of-file for a component file, the L2 automatically continues with the next component file. The results are accumulated sequentially from all files that were read.

On the other hand, an L3 command (read logical sequential by descriptor) is performed as separate parallel calls to each component file, and the results are merged into a single sequence before they are returned to the caller.

Recommended Nucleus Changes for Expanded Files

To better accommodate parallel processing of component files for a single command, an increase in the following ADARUN parameter values is recommended for the nucleus session:

Parameter	Description
LI	Length of the table of ISNs (TBI)
LQ	Length of the table of sequential commands
LWP	Length of the Adabas work pool area
LS	Length of the search/sort area
NQCID	Maximum number of active command IDs (CIDs) allowed per user

Restrictions When Using Expanded Files

The following limitations apply to programs running on an expanded file:

■ Physical and/or soft coupling is not currently supported for expanded files.

■ Multiclient support is not provided for expanded files.

■ Once established, component files of an expanded file cannot be renumbered.

■ When Adabas Security is used for an expanded file, the following should be the same for all component files:

   ■ protection profile

   ■ password

   ■ security-by-value profile

   ■ cipher code

Expanded Files and Adabas Utilities

Although the expanded file is transparent to the user making Adabas calls, the DBA running the Adabas utilities must be aware of the existence of an expanded file. Adabas utility functions process expanded files in one of two ways:

■ they process the complete expanded file; or

■ they process component files.

Functions That Process Complete Expanded Files

Utility functions that process the entire expanded file include the following ADADBS, ADARES, and ADASAV functions:

ADADBS DELETE Function

Deletes a complete expanded file only.

ADARES REGENERATE and BACKOUT FILE Functions

Process the expanded file as a whole whenever one of the component files is specified in the file list. All other component files must then also be specified.

ADASAV

■ **RESTORE(file)Function**
Processes the expanded file as a whole whenever one of the component files is specified in the file list. All other component files must then also be specified.

■ **SAVE (file)Function**
Processes the expanded file as a whole whenever one of the component files is specified in the file list. When running the SAVE (file) function while the Adabas nucleus is active, all other component files must then also be specified.

Functions That Process Component Files

Utility functions that process component files include the following ADADBS, ADAINV, ADALOD, ADAORD, ADAACK, ADADCK, ADAICK, ADAVAL, ADAULD, and ADASCR functions.

ADAACK, ADADCK, ADAICK, ADAVAL, ADAULD

All functions of these utilities check single component files only.

ADADBS

■ *CHANGE / NEWFIELD Functions.* These functions modify the field definition table (FDT) of a single component file only. The DBA must perform the CHANGE or NEWFIELD function for all component files in the expanded file. ADADBS prints a message indicating that the specified file is part of an expanded file, and then completes with condition code 4.

■ *RELEASE Function.* Releases the index for a descriptor of a single component file. The DBA must perform the RELEASE function for all component files in the expanded file. ADADBS prints a message indicating that the specified file is part of an expanded file, and then completes with condition code 4.

ADAINV

■ *INVERT Function.* Creates the index for a new descriptor of a single component file. The DBA must perform the INVERT function for all component files in the expanded file. ADAINV prints a message indicating that the specified file is part of an expanded file, and then completes with condition code 4.

■ *COUPLE Function.* The ADAINV COUPLE function is not available for expanded files.

ADALOD UPDATE Function

Adds records to/deletes records from a single component file. When performing a mass update on some or all component files, the complete list of ISNs to be deleted from all component files can be supplied. ADALOD automatically selects only the ISN values from the specified range that are appropriate for the component file currently being processed. The same is true when adding new records with USERISN=YES.

When new records are being added with USERISN=NO but no free ISN is found, the loader cannot allocate a new Address Converter extent since the ISN range cannot be increased (NOACEXTENSION is active for all component files). Instead, ADALOD creates the index as though end-of-file had been reached. The remaining records not loaded may be added later to another component file using the SKIPREC parameter.

ADALOD does not check for unique descriptor values across component file boundaries.

ADAORD REORFILE / REORFASSO / REORDATA Functions

Each reorder the respective areas of a single component file. Since the file is not logically changed, the functions need not be performed on all component files of an expanded file.

ADASCR (Adabas Security) Functions

Defines security profiles for individual component files only. The protection, password, security-by-value and cipher code for each component file should be defined the same for all component files in an expanded file.

22 Database Maintenance Tasks

This part of the DBA documentation describes the tasks involved in defining, maintaining and running an Adabas database.

This information is organized under the following headings:

- **Defining an Adabas Database**
- **Database Space Management**
- **Database Monitoring and Tuning**
- **Error Handling and Message Buffering**
- **Universal Encoding Support (UES)**
- **Multiple Platform Support**

23 Defining an Adabas Database

This chapter describes the procedure for defining an Adabas database. It is important for the DBA to understand the information provided for each step before attempting to define a new database.

Defining a new database involves the following steps:

After you have completed these steps successfully, you can use the ADACMP and ADALOD utilities to load user files into the database.

Step 1 : Estimate the Size of the Database

Components Required by the Nucleus

The Adabas nucleus requires three database components: Data Storage, an Associator, and a Work area.

Data Storage

The Data Storage component contains the compressed data records of each file in the database.

Associator

The Associator contains elements for each file in the database and for the database as a whole.

For each file in the database, the Associator includes an inverted list, an address converter, and a field definition table (FDT):

- The *inverted list*, which resolves Adabas search commands and reads records in logical sequence, comprises the normal index (NI) and as many as 14 upper indexes (UI). All of the values for each descriptor in the file are contained in the NI along with a count of the records that contain each value and a list of the ISNs of those records. To increase search efficiency, UI levels are automatically created by Adabas as required, each level to manage the next lower level index. The first level UI, like the NI it manages, contains entries for only one descriptor in each index block. All other UI levels contain entries for all descriptors in each index block. Upper Indexes require a minimal amount of space (two blocks is the minimum).

- The *address converter* maps the logical identifier of a record (ISN) to the relative Adabas block number (RABN) of the Data Storage block where the record is stored. It comprises a list of RABNs in ISN order; for example, the fifteenth entry in the address converter contains the RABN for ISN 15.

- The *field definition table* (FDT) defines the logical contents of an Adabas file. It contains the name, level, length, format, and specified options for each field in the file.

For the database as a whole, the Associator includes storage management tables and coupling lists:

- *Storage management tables* list the Associator and Data Storage blocks that are available for allocation, along with the amount of unused space in each Data Storage block.

- *Coupling lists* exist for physically coupled files only are used to resolve search commands in which descriptors from more than one file are used.

Work

The Work area stores information in four parts:

- Part 1. Stores data protection information required by the routines for autorestart and autobackout.

- Part 2. Stores intermediate results (ISN lists) of search commands.

- Part 3. Stores final results (ISN lists) of search commands.

- Part 4. Stores data related to distributed transaction processing.

  Note: If you have Adabas Transaction Manager Version 7.5 or later installed, Work part 4 of DDWORKR1 is no longer supported. Instead, a second Work dataset, DDWORKR4 is required. DDWORKR4 is a container dataset used for the same purpose as Work part 4 of DDWORKR1, but it can be used in parallel by all members in a cluster. The DDWORKR4 dataset should be allocated and formatted in the normal way, using a block size greater than or equal to DDWORKR1. It should be at least as large as the cluster's LP parameter of the database or cluster.

Other Components

Sort and Temp Areas

The Adabas utilities ADAINV, ADALOD, and ADAVAL require two additional datasets, SORT and TEMP, for sorting and intermediate storage of data.

The sizes of TEMP and SORT vary according to the utility function to be executed. These datasets can be allocated during the job and then released, or permanent datasets can be allocated and reused.

Logs

Adabas has the following optional logs:

- The *command log* (CLOG) records information from the control block of each Adabas command that is issued. The CLOG provides an audit trail and can be used for debugging and for monitoring usage of resources. Single, dual, or multiple (2-8) datasets can be used (multiple datasets are recommended).

- The *protection log* (PLOG) records before-images and after-images of records and other elements when changes are made to the database. It is used to recover the database (up to the last ET) after restart. Single, dual, or multiple (2-8) datasets can be used (multiple datasets are recommended).

- The *recovery log* (RLOG) records additional information that the Adabas Recovery Aid uses to construct a recovery job stream.

> **Note:** Each CLOG, PLOG, and RLOG dataset is limited to 16,777,215 (x'FFFFFF') blocks/RABNs.

General Space Requirements

The space requirements for the Associator (NI, UI, and AC) and Data Storage are calculated automatically for each file by the ADALOD utility and the ADACMP utility, respectively. If you want to allocate a specific amount of space to a file or estimate the space needed for a file without actually executing these utilities, you can use the formulas provided in this chapter.

If the number and size of the files that will eventually be loaded into the database are not known at the time that the database is established, it is not necessary to allocate a large amount of extra space to the Associator and/or Data Storage, since the space may be increased subsequently by using the ADD or INCREASE function of the ADADBS utility.

The initial allocation for Associator and Data Storage should, however, allow for the loading of all currently planned files in addition to a reasonable amount of database expansion (adding new files or updating existing files).

When estimating the Associator space, the following requirements for the database as a whole must be added to the estimates calculated for each file within the database (normal index, upper indexes, and address converter):

- The first 30 Associator blocks are used by Adabas for storing internal control information. Note that the physical block sizes for Associator, Data Storage, and Work vary from one Adabas component to another and according to the device type on which each component is located.

- Associator blocks equalling five times the value specified by the MAXFILES parameter are reserved by Adabas for file control information. The MAXFILES parameter is set when running the ADADEF utility.

General Procedure for Estimating Space

1. Estimate the following requirements for *each file*; then add the estimates together for an estimate for the whole database:

   ■ Associator (normal index, upper indexes, address converter)

   ■ Data Storage

2. Estimate the following requirements for the *database as a whole*:

   ■ Associator (space reserved by Adabas)

      ■ First 30 blocks for internal control information;

      ■ (`MAXFILES * 5`) blocks for file control information (the ADADEF parameter MAXFILES specifies the maximum number of files that can be loaded into the database);

   ■ Work area; sort area; temp area; logs

Estimation Formulas

The following sections provide formulas for estimating the space that should be allocated to each component.

■ Associator, in terms of

   ■ normal index (NI)

   ■ upper index (UI)

   ■ address converter (AC)

■ Data Storage

■ Work, in terms of

   ■ part 1 (data protection information)

   ■ part 2 (intermediate results of search commands)

   ■ part 3 (ISN lists from search commands)

   ■ part 4 (data related to two-phase commit processing)

■ sort space

Rules of Precedence in the Formulas

The formulas follow the normal rules of precedence; that is, expressions are evaluated in the following order:

1. Elements in parentheses;

2. Multiplication and division operations;

3. Addition and subtraction operations;

4. Left to right (when elements have the same precedence level, the one on the left is evaluated first).

Normal Index (NI)

Use the following formula to estimate the normal index space required for each descriptor in the file:

```
NIRBYTES = ISNSIZE * AVUQVAL * RECORDS + DESCVALS * (AVLENG + 2)
```

where

NIRBYTES	is the space requirement for normal index, in bytes.
ISNSIZE	is the length of ISNs in the file (3 or 4 bytes). The ISN length is specified by the ADALOD parameter ISNSIZE.
AVUQVAL	is the average number of unique values for the descriptor in each record.
RECORDS	is the number of records to be contained in the file, which is specified by the ADALOD parameter MAXISN.
DESCVALS	is the number of unique values for the descriptor in the file.
AVLENG	is the average length of the values for the descriptor.

AVUQVAL

AVUQVAL is less than or equal to 1 unless the descriptor is a multiple-value field (MU) or part of a periodic group (PE).

If the descriptor is defined with the NU (null suppression) option, AVUQVAL equals the average number of values per record *minus* the percentage of records that contain a null value (the field is empty). For example, if each record has one value for the descriptor and 20 per cent of the values are null

```
AVUQVAL = 1 - 0.2 = 0.8
```

Similarly, if an MU field has an average of 10 values per record and 20% of the values are null

```
AVUQVAL = 10 - 2 = 8
```

AVLENG

If the descriptor field is not defined with the FI (fixed length) option, AVLENG equals the average compressed length of the field, including the length byte. If the descriptor is defined with the FI option, AVLENG equals the standard length of the field.

ISNSIZE * AVUQVAL * RECORDS

ISNSIZE * AVUQVAL * RECORDS represents the space required to store the ISNs. It is the important factor for descriptors that have many duplicate values.

DESCVALS * (AVLENG + 2)

DESCVALS * (AVLENG + 2) represents the space required to store the descriptor values. It is the important factor for descriptors that have a large proportion of unique values.

Convert Bytes to Blocks

Use the following formula to convert bytes to blocks. Round the result up to the next block.

```
NIRBLOCKS = NIRBYTES / (ASSOBLKSIZE * (1 - PADFACTOR / 100))
```

where

NIRBLOCKS	is the NI space requirement, in blocks.
NIRBYTES	is the NI space requirement, in bytes (from the NIRBYTES formula).
ASSOBLKSIZE	is the ASSOR1 block length (see Adabas Device types and Block Sizes).
PADFACTOR	is the ASSOR1 block padding factor, which is a percentage of the block length expressed as a value between 1-90.

Examples

The following examples assume that ASSOR1 is stored on a 3380 device.

Example 1:

Descriptor AA has one value per record and no null values. There are 50 different values for AA in the file. The average compressed length for the values is 3 bytes.

ISNSIZE=3
MAXISN=20000
PADFACTOR=10 (%)

```
NIRBYTES   = 3 * 1 * 20,000 + 50 * (3 + 2)
                    = 60,000 + 250
                    = 60,250 bytes
```

```
NIRBLOCKS  = 60,250 / (2004 * (1 - 0.1))
                    = 33.41
                    = 34 blocks
```

Example 2:

Descriptor BB has one value per record and no null values. There are 20,000 different values for BB in the file. The average compressed length for the values is 10 bytes.

ISNSIZE=4
MAXISN=20000
PADFACTOR=10 (%)

```
NIRBYTES   = 4 * 1 * 20,000 + 20,000 * (10 + 2)
                    = 80,000 + 240,000
                    = 320,000 bytes
```

```
NIRBLOCKS  = 320,000 / (2004 * (1 - 0.1))
                    = 177.42
                    = 178 blocks
```

Example 3:

Descriptor CC is a null-suppressed multiple-value (MU) field with an average of 10 occurrences and 3 null values per record. There are approximately 300 different values for CC in the file. The average compressed length for the values is 4 bytes.

ISNSIZE=3
MAXISN=10000
PADFACTOR=5 (%)

```
NIRBYTES   = 3 * 7 * 10,000 + 300 * (4 + 2)
                    = 210,000 + 1,800
                    = 211,800 bytes
```

```
NIRBLOCKS  = 211,800 / (2004 * (1 - 0.05))
                    = 111.25
                    = 112
```

Example 4:

Descriptor DD is a null-suppressed field contained within a periodic group. DD contains an average of 5 values per record; there is an average of 3 null values per record. There are 10 different values for DD in the file. The average compressed length for the values is 5 bytes.

ISNSIZE=4
MAXISN=10000
PADFACTOR=5 (%)

```
NIRBYTES  = 4 * 2 * 10,000 + 10 * (5 + 2)
                = 80,000 + 70
                = 80,070 bytes
```

```
NIRBLOCKS  = 80,070 / (2004 * (1 - 0.05))
                  = 42.06
                  = 43
```

Upper Index (UI)

Use the following formula to estimate the UI space required for each descriptor in the file:

```
UIRBYTES = NIRBLOCKS * (AVDESCLEN + ISNSIZE + RABNSIZE + 1)
```

where

UIRBYTES	is the UI space requirement, in bytes.
NIRBLOCKS	is the NI space requirement, in blocks (from the NIRBLOCKS formula).
AVDESCLEN	is the average compressed length of the values for the descriptor.
ISNSIZE	is the length of ISNs in the file (3 or 4 bytes). The ISN length is specified by the ADALOD parameter ISNSIZE.
RABNSIZE	is the length of RABNs in the database (3 or 4 bytes). The RABNSIZE is specified for all files in a database when the database is defined.

> **Note:** RABNSIZE refers only to the length of the relative Adabas block number. It does not refer to the block size.

Convert Bytes to Blocks

Use the following formula to convert bytes to blocks. Round the result up to the next block.

```
UIRBLOCKS = UIRBYTES / (ASSOBLKSIZE * (1 - PADFACTOR / 100))
```

where

UIRBLOCKS	is the UI space requirement, in blocks.
UIRBYTES	is the UI space requirement, in bytes (from the UIRBYTES formula).
ASSOBLKSIZE	is the ASSOR1 block length (see Adabas Device Types and Block Sizes).
PADFACTOR	is the ASSOR1 block padding factor, which is a percentage of the block length expressed as a value between 1-90.

Example

This example assumes that the Associator is stored on a 3380 DASD; therefore, ASSOR1 has 2004 bytes per block.

The NI block requirement for this file is estimated to be 45 blocks. The average compressed length of the values for the descriptor is 3 bytes. The database has 3-byte (24-bit) RABNs; the file has 3-byte ISNs. The ASSOR1 block padding factor is 5 (%).

```
UIRBYTES   = 45 * (3 + 3 + 3 + 1)
                   = 450
```

```
UIRBLOCKS  = 450 / (2004 * (1 - 0.05))
                       = 450 / 1903.8
                       = 0.24
                       = 1 block    ( a minimum of 2 blocks can be allocated for
the UI)
```

Address Converter (AC)

Use the following formula to estimate the address converter space required for the file. Round the result up to the next whole block.

```
ACBLOCKS = (MAXISN +!) * RABNSIZE / ASSOBLKSIZE
```

where

ACBLOCKS	is the space requirement for the address converter, in blocks.
MAXISN	is the MAXISN setting for the file.
RABNSIZE	is the length of RABNs in the database (3 or 4 bytes). The RABNSIZE is specified for all files in a database when the database is defined.
	Note: RABNSIZE refers only to the length of the relative Adabas block number. It does not refer to the block size.
ASSOBLKSIZE	is the Associator block size (see Adabas Device types and Block Sizes).

Examples

The following examples assume that the Associator is stored on a 3380 DASD; ASSOR1 has 2004 bytes per block.

Example 1:

MAXISN=2000000
RABNSIZE=3

```
ACBLOCKS  = (2,000,000 * 3) / 2004
                = 6,000,000 / 2004
                = 2994.01
                = 2995 blocks
```

Example 2:

MAXISN=2000000
RABNSIZE=4

```
ACBLOCKS  = (2,000,000 * 4) / 2004
                = 8,000,000 / 2004
                = 3992.02
                = 3993 blocks
```

Data Storage

Use the following formula to estimate the space required for Data Storage. Round the result up to the next whole block.

```
DATASTORAGE = MAXISN / ((DSBLKSIZE * (1 - (PADFACTOR / 100))) / AVRECLEN)
```

where

DATASTORAGE	is the space requirement for Data Storage, in blocks.
MAXISN	is the MAXISN setting for the file.
DSBLKSIZE	is the Data Storage block size, rounded down to the next integer (see Adabas Device types and Block Sizes).
PADFACTOR	is the Data Storage block padding factor, which is a percentage of the block length expressed as a value between 1-90.
AVRECLEN	is the average compressed record length.

Example

MAXISN = 1000000
Average compressed record length = 50
Model 3380 blocksize for DATA = 4820
Data Storage block padding factor = 5 (%)

```
DATASTORAGE  = 1,000,000 / ((4820 * (1 - 0.05)) / 50)
                        = 1,000,000 / (4579 / 50)
                        = 1,000,000 / 91
                        = 10,989.01
                        =  10,990 blocks
```

How Adabas Allocates Work Space

When you allocate the Work datasets, allocate enough space for all parts. The minimum allowable Work space is 300 blocks. Three ADARUN parameters can be used to break up the space into parts 1-4 as follows:

> **Note:** Work part 4 or DDWORKR1 is no longer supported if you have Adabas Transaction Manager Version 7.5 or later installed. Instead, you should use the DDWORKR4 dataset. DDWORKR4 is a container dataset used for the same purpose as Work part 4, but it can be used in parallel by all members in a cluster. The DDWORKR4 dataset should be allocated and formatted in the normal way, using a block size greater than or equal to DDWORKR1. It should be at least as large as the cluster's LP parameter of the database or cluster.

- The ADARUN LP parameter specifies the size of Work part 1. The default setting is 1000 blocks; the minimum is 200. A database with little or no updating needs 500-1000 blocks. Work part 1 begins with RABN 1; the last RABN is the value of LP.

- The ADARUN LWKP2 parameter specifies the size of Work part 2. If LWKP2=0 (the default), Adabas calculates the size automatically, using the formula described in *Work Part 2: Intermediate Search Results* .

Work part 2 begins in the block following the last block of Work part 1; thus, the first RABN of part 2 is given by

```
1 + LP
```

- The ADARUN LDTP parameter specifies the size of Work part 4 when ADARUN DTP=RM. If LDTP=0 (the default), the length of Work part 4 is equivalent to the length of Work part 1 (ADARUN LP). If a non-zero value is specified, it must be greater than the value specified for LP. If a smaller value is specified, Adabas changes it to equal the LP value.

> **Note:** Work part 4 of DDWORKR1 is no longer supported if you have Adabas Transaction Manager Version 7.5 or later installed. Instead, you should use the DDWORKR4 container dataset.

Work part 4 begins in the block following the last block of Work part 2; thus, the first RABN of part 4 is given by

```
1 + LP + LWKP2
```

- After allocating parts 1, 2, and possibly 4, Adabas allocates the remaining blocks to Work part 3. It is important that you allocate enough space to the DDWORKR1 dataset to leave at least 50 blocks for part 3.

Work Part 1: Data Protection Information

The data protection area for all transactions running in parallel must fit into 1/4 of the Work part 1 (that is, LP) area. Following are general guidelines for determining the proper size for Work part 1:

1. The total Work part 1 size should be four times the estimated size required for a single average transaction in bytes times the maximum number of transactions that run in parallel. This value is then divided by the Work block size (in bytes) minus 200 to convert bytes to blocks.

2. If some transactions are very long, then those transactions alone plus all short transactions executed in parallel should be used to determine the size of a single average transaction.

3. The size of a single average transaction is determined by estimating the average number of updates (modifications, additions, and deletions) per transaction and multiplying that number by the estimated bytes required per update. To this is added space for ET data and for the ET record in bytes.

4. The size required per update is determined by the average compressed record length in bytes times 4 (before image, after image, and DVT space for each) plus 100 bytes for each protection record header (that is, 100 times 4).

A formula that expresses these guidelines is

```
WK1= ( 4 * TASIZE * TAP ) / ( BLKSIZE - 200 )
```

where

WK1   is size of Work part 1 in blocks

TASIZE   is the size of a single average transaction in bytes

TAP   is the maximum number of transactions actually executed in parallel

BLKSIZE is the Work block size in bytes

```
TASIZE= ( ( ( 4 * AVCRL ) + 400 ) * UPDTA ) + ETDATA + 100
```

where

AVCRL	is the average compessed record length in bytes
UPDTA	is the average number of updates per transaction
ETDATA	is the average length of ET data in bytes

Example

If AVCRL = 300 bytes, UPDTA = 4, and ETDATA = 200 bytes, then

```
TASIZE= ( ( ( 4 * 300 ) + 400 ) * 4 )+ 200 + 100 = 6700 bytes
```

If TAP = 100 and BLKSIZE = 5492, then

```
WK1 = ( 4 * 6700 * 100 ) / ( 5492 - 200 ) = 506.46 blocks
```

Work Part 2: Intermediate Search Results

Use the following formula to estimate the space required for the Work part 2 area. Round the result up to the next whole block.

```
WORK2 = 22 + 2 * ((4 * RECORDS) / (BLKSIZE - 16))
```

where

WORK2	is the Work part 2 space requirement, in blocks.
RECORDS	is the number of records in the file with the most records. This number equals

```
TOPISN - MINISN + 1
```

where:

TOPISN	is the highest ISN currently used in the file.
MINISN	is the lowest ISN used in the file.

The MINISN value is specified with the ADACMP/ADALOD parameter MINISN; 1 is the default. You can use the ADAREP utility to display the TOPISN and MINISN values for the files in a database.

BLKSIZE	is the block size of the device where the Work dataset is stored, (see Adabas Device types and Block Sizes).

Note: An Adabas internal table requires one byte of storage for each Work part 2 block.

Example

The number of records in the largest file in the database is 500,000. The Work dataset is stored on a 3380 device.

```
WORK2 = 22 + 2 * ((4 * 500,000) / (5492 - 16))
              = 752.46
              = 753 blocks
```

Work Part 3: ISN Lists from Search Commands

Adabas allocates to Work part 3 (resultant ISN lists) the Work space remaining after the allocation of the part 1 (data protection information) and part 2 (intermediate results) areas.

The *minimum* requirement for this area is 50 blocks.

If insufficient space is provided for this area, Adabas may be unable to execute additional search commands until the space currently occupied by ISN lists has been released. Consider the following factors when estimating the space needed for the Work part 3 area:

- The number of concurrent search commands to be processed (each ISN list is stored in a separate block), and the expected size of the resulting ISN lists (each ISN is stored as 4 bytes, regardless of the ISNSIZE specified for the file);

- The number of saved ISN lists resulting from previous search commands with the SAVE ISN LIST option which will be held concurrently;

- The amount of memory which will be required by Adabas as a result (each block allocated to this area requires 4 bytes of memory).

Example

A maximum of 100 search commands with an average of 25 resulting ISNs per command are to be processed concurrently during the session.

Adabas will need 100 blocks in the Work part 3 area.

Work Part 4: Data Related to Distributed Transaction Processing

> **Note:** Work part 4 of DDWORKR1 is no longer supported if you have Adabas Transaction Manager Version 7.5 or later installed. Instead, you should use the DDWORKR4 dataset. DDWORKR4 is a container dataset used for the same purpose as Work part 4, but it can be used in parallel by all members in a cluster. The DDWORKR4 dataset should be allocated and formatted in the normal way, using a block size greater than or equal to DDWORKR1. It should be at least as large as the cluster's LP parameter of the database or cluster.

Work part 4 maintains information about some of the global transactions involved in distributed processing. For example, during phase one of the commit process, a global transaction's protection data may be copied from Work part 1 to Work part 4 if it is no longer possible to store the information in Work part 1.

If an overflow of Work part 4 is pending, the nucleus can force a transaction termination. This clears Work part 4 except for transaction IDs (XIDs) and local transaction status information.

The space allocated for Work part 4 must be at least that allocated for Work part 1:

```
LDTP>=LP
```

Whether a larger size is needed for Work part 4 depends on the applications running against the database and on the system load.

Because the information maintained in Work part 4 cannot currently be moved to a different area, you can alter the size of Work part 4 between sessions only as follows:

- you can decrease the size of Work part 4 if it was not used at all in the previous session.
- you can increase the size of Work part 4 if it was used in the previous session.

Sort

The following formulas *estimate* the sort dataset space used for sorting all values of a single descriptor. Multiple descriptors are sorted successively: all values are sorted for the first descriptor, then all values for the second descriptor, and so on. Therefore, estimate the space for the largest possible descriptor sort; that will be enough for all descriptors.

Use the following formula to estimate the space required for the sort area:

```
DESCSPACE = (AVDESCLEN + (1 + ISNSIZE)) * NUMRECS * AVPEOCCUR * AVMVOCCUR
```

where

DESCSPACE	is the total descriptor space required, in bytes.
AVDESCLEN	is the average compressed descriptor length, in bytes.
ISNSIZE	is the size of the ISN being used (either 3 or 4).
NUMRECS	is the number of records.
AVPEOCCUR	is the average number of periodic group occurrences, if the descriptor is in a periodic group. Otherwise, set this value to 1.
AVMVOCCUR	is the average number of multiple-value field occurrences, if the descriptor is a multiple-value field. Otherwise, set this value to 1.

Work Pool Size

Use the following formula to estimate the space required for the work pool:

```
LWPAVAIL = LWPSIZE - 1216 - (32 * SORTDEVTRKS) - SORTDEVBSIZ
```

where

LWPAVAIL   is the available part of the work pool space, in bytes.

LWPSIZE   is the total work pool size, in bytes (the utility's LWP parameter value).

SORTDEVTRKS is the number of sort device tracks per cylinder (see Adabas Device types and Block Sizes).

SORTDEVBSIZ   is the sort device block size, in bytes.

Sorted Partial Sequences

To determine the space required for sorted partial sequences, use one of the following calculations. The one to use depends on the AVDESCLEN value (average descriptor length) used to calculate the DESCSPACE value (total descriptor space required).

■ If AVDESCLEN is less than 12

```
LENGSEQ = (LWPAVAIL * (AVDESCLEN + (1 + ISNSIZE))) / 2
```

where

LENGSEQ   is the length of sorted partial sequences.

LWPAVAIL   is the available Work pool space.

AVDESCLEN is the average compressed descriptor length, in bytes.

ISNSIZE   is the size of the ISN being used (either 3 or 4).

■ If AVDESCLEN is equal to or greater than 12

```
LENGSEQ = (LWPAVAIL * 2) / 3
```

where

LENGSEQ   is the length of sorted partial sequences.

LWPAVAIL   is the available Work pool space.

Device Surfaces

Use the following formula to calculate the number of device surfaces rounded up to the next integer:

```
SURFACES = (DESCSPACE / LENGSEQ) / SORTDEVTRK
```

where

SURFACES	is the number of surfaces required for sort space, rounded up to the next integer.
DESCSPACE	is the total descriptor space required, in bytes.
LENGSEQ	is the length of sorted partial sequences.
SORTDEVTRKS	is the number of sort device tracks per cylinder (see Adabas Device types and Block Sizes).

Estimated Sort Size

Using the intermediate values calculated for LENGSEQ and SURFACES, compute the estimated sort size as follows:

```
SORTSIZE = (SURFACES * SORTDEVTRKS * LENGSEQ * 2) / (SORTDEVBSIZ - 4)
```

where

SORTSIZE	is the estimated sort area size, in blocks. This value should be rounded up to the next full cylinder.
SURFACES	is the number of surfaces required for sort space, calculated earlier and rounded up.
SORTDEVTRKS	is the number of sort device tracks per cylinder (see Adabas Device types and Block Sizes).
LENGSEQ	is the length of sorted partial sequences.
SORTDEVBSIZ	is the sort device block size, in bytes.

Number of Descriptors Sorted

Use the following formula to estimate the number of descriptors that can be sorted in the SORTSIZE space calculated in the previous formula (assuming the same descriptor definition that was used when calculating DESCSPACE):

```
DESCOUNT = SURFACES * SORTDEVTRKS * LENGSEQ / (AVDESCLEN + (1 + ISNSIZE))
```

where

DESCOUNT   is the number of descriptors defined in the earlier DESCSPACE calculation that can be
                held in the SORTSIZE space calculated above.

SURFACES   is the number of surfaces required for sort space, calculated earlier and rounded up.

SORTDEVTRKS is the number of sort device tracks per cylinder (see Adabas Device types and Block Sizes).

LENGSEQ   is the length of sorted partial sequences.

AVDESCLEN   is the average compressed descriptor length, in bytes.

ISNSIZE   is the size of the ISN being used (either 3 or 4).

Step 2 : Allocate Space

1. Use standard operating-system procedures to allocate datasets for the following Adabas components:

   - Required by the Adabas nucleus:

     - Associator (ASSO)

     - Data Storage (DATA)

     - Work area (WORK1)

     - Work area (WORK4), if Adabas Transaction Manager 7.5 or later is installed.

   - Required by some Adabas utilities:

     - sort area (SORT)

     - temp area (TEMP)

   - Optional (but recommended) logs:

     - dual or multiple command log (CLOG)

     - dual or multiple protection log (PLOG)

     - recovery log (RLOG)

   Normally, ASSO, DATA, and WORK are each allocated as a single operating system dataset.
   However, you can allocate the Associator and Data Storage on up to five separate datasets
   each; the datasets can be allocated on the same or different device types.

2. To minimize contention and distribute I/O activity more evenly across hardware channels,
   place the ASSO, DATA, WORK, PLOG, and RLOG datasets on different physical volumes. If
   only two volumes are available, place ASSO on one volume and DATA and WORK datasets
   on the second.

   The WORK and PLOG datasets should be on different volumes, since a PLOG I/O operation is
   always followed by a WORK I/O operation.

The RLOG dataset should always be placed on a separate device of the same type.

Disk access time may be considerably reduced by separating TEMP from DATA, and SORT from ASSO. When loading files containing 100,000 records or more, splitting SORT across two volumes reduces disk arm movement.

3. Specify the disk space allocation in the job control (JCL/JCS or VM CONTROL minidisk) of the format utility (ADAFRM). See the Adabas Utilities documentation for specific information and job examples.

Examples

Example 1 : Database Allocation Using Two Volumes

Volume 1	Volume 2			
ASSO	DATA			
TEMP	WORK			
PLOG1	SORT			
PLOG2				

Example 2 : Database Allocation Using Three Volumes

Volume 1	Volume 2	Volume 3		
ASSO	DATA	WORK		
PLOG1	SORT	TEMP		
	PLOG2			

Example 3 : Database Allocation When Loading a Large File

Volume 1	Volume 2	Volume 3	Volume 4	Volume 5
ASSO	DATA	DATA	DATA	SORT (2nd half)
	PLOG1	PLOG2	SORT (1st Half)	WORK
		TEMP		

Performance Note

Software AG does not recommend using hardware compression (IDRC) for protection log files. The ADARES utility BACKOUT function will run at least twice as long under OS/390 or z/OS when processing compressed data. Also, the BACKOUT function is not supported for compressed data on VSE/ESA, VM/ESA, or z/VM systems.

Step 3 : Format the Space

Before loading the first file into the database, use the ADAFRM utility to format the ASSO, DATA, and WORK datasets. Refer to the Adabas Utilities documentation for information about the ADAFRM utility.

Format TEMP and SORT before using any Adabas utility that requires them. You can allocate and format temporary datasets and delete them after executing the utility, or allocate and format permanent datasets for repeated use.

> **Note:** When using the Recovery Aid (including the RLOG), you must catalog all temporary datasets. When running with the Recovery Aid, the general rule is to catalog temporary datasets in jobs that require the Associator datasets.

Format the CLOG, PLOG, and RLOG datasets before starting the first session in which the logging is to be performed.

Step 4 : Define Database Parameters

Once all database components have been physically allocated and formatted, use the ADADEF utility to define database parameters such as database identification, maximum number of files, system file assignment, and so on.

The sizes of the ASSO, DATA, and WORK datasets must be defined with ADADEF DEFINE parameters. Note that defining the sizes to Adabas is different from allocating the space; the datasets must be allocated and formatted before you can define them to Adabas. The sizes of the other datasets are defined to Adabas as follows:

- TEMP and SORT: when you execute the utility that uses them;
- CLOG and PLOG: at the start of a nucleus session, with ADARUN parameters;
- RLOG: when logging begins, using the PREPARE function of the ADARAI utility.

> **Note:** Each log dataset (CLOG, PLOG, or RLOG) is limited to 16,777,215 (x'FFFFFF') blocks/RABNs.

24 Database Space Management

This chapter provides the DBA with all pertinent information related to database space management. Information is provided about

- Adabas physical and logical extents;

- Adabas relative block number (RABN);

- the role of the Adabas nucleus and utilities in allocating/deallocating space;

- using the database status report to monitor database space usage;

- potential space utilization problems and recommended action.

This information is organized under the following headings:

Adabas Physical Extents

An Adabas physical extent (database container) is a collection of physical blocks assigned to a given database component (Associator, Data Storage, Work) during the definition of the database (see the ADADEF utility, ASSOSIZE, DATASIZE and WORKSIZE parameters).

The space for a physical extent is allocated using the standard allocation procedures of the operating system in use.

An Adabas physical extent may be allocated within a single operating system extent which consists of a primary extent only, or may be allocated as a primary extent together with one or more secondary extents. The secondary extents need not be contiguous to the primary extent or to each other.

An Adabas physical extent may be contained on a single physical volume or may extend across multiple volumes. The Associator and Data Storage components may each contain up to five Adabas physical extents.

Relative Adabas Block Number (RABN)

Adabas information is stored in space allocated in "blocks". A block's size depends on

- the physical device on which the block is located; and

- the Adabas component to which the block is assigned.

For example, the default device type used by Adabas is the IBM 3380 disk. This device is assumed in many utility and operating parameters as the device type unless another is specified.

When 3380 space is allocated for Adabas, it must be designated as Associator (ASSO), Data Storage (DATA), the Work area (WORK), logging area (PLOG, CLOG, RLOG), sort area, or temp area. A

3380 block allocated to ASSO contains 2004 bytes, but a 3380 block allocated to DATA contains 4820 bytes. Block sizes are predefined for each device type and Adabas component; Adabas Device types and Block Sizes lists block sizes by device type and component.

Adabas block sizes are not fixed by hardware; however, they are referred to as "physical" blocks to coincide with the level of description used for physical block (FBA) devices. Software AG tries to maintain consistent block definitions, by device type, from release to release. However, in some cases the block size for a component type may change to accommodate expanded Adabas facilities. Thus, a specific Adabas component (PLOG, ASSO, etc.) may need to be reformatted before you can run a new Adabas release.

Adabas identifies and addresses each physical block within a database component by its "relative Adabas block number" (RABN) , which indicates the block's position relative to the beginning of the component. RABNs are assigned in ascending sequence in Data Storage, Associator, or Work, starting with 1 for the first physical block in the first Adabas physical extent (the first track of the first extent is not used). If multiple physical extents are used, the RABN assignment continues across the physical extents.

The number of RABNs that can be assigned to ASSO and DATA depends on the RABNSIZE parameter, which is specified when the database is defined. RABNSIZE specifies the length of relative Adabas block numbers in the database (not the length of the block itself).

- If RABNSIZE=3 (block number is 24 bits or three bytes), the maximum number of RABNs is 16,777,215.

- If RABNSIZE=4 (block number is 31 bits or four bytes), the maximum number of RABNs is 2,147,483,646.

See *Relative Adabas Block Number (RABN) Calculation* in the Adabas Operations documentation for more information about RABNs.

Adabas Logical Extents

An Adabas logical file extent is a group of consecutive RABNs allocated by the Adabas nucleus or an Adabas utility.

For each file loaded into the database, a minimum of one of each of the following types of Adabas logical extents is allocated to the file:

Logical extent	Allocated from the physical extent . . .
Data Storage	Data Storage
address converter	Associator
normal index	Associator
upper index	Associator

Additional logical extents are allocated by the Adabas nucleus or an Adabas utility when additional space is needed as a result of file maintenance.

A maximum of one address converter may be allocated for any file. The maximum number of logical file extents that can be defined for any other type of logical extent is five.

When the Adabas nucleus starts up, the FCBs are checked. If there are alreay five extents of any extent type, the Adabas nucleus prints messags suggesting that the files should be reordered. In addition, if the last file extent of each type has only five or less free ISNs or blocks left, the nucleus locks the file for utility use only and nucleus response code 75 is issued. Normal users trying to access the file will then get response code 17 or 48.

Adabas Space Allocation and Deallocation

This section provides an overview of Adabas space allocation and deallocation procedures. A full understanding of these procedures will help ensure correct and efficient database space management.

Free Space Table

All space available for allocation is stored in the free space table (FST). This table contains all RABN extents that are currently available for an allocation to any file.

Space Allocation by the Adabas Nucleus

When processing an add or update record command, the Adabas nucleus may need to allocate an additional extent to any of the following file components:

- address converter
- normal index
- upper index
- Data Storage

Address Converter (AC)

The size of the address converter is initially defined by the MAXISN parameter in the ADALOD utility. The actual "highest expected" ISN is slightly higher because the address converter is stored in entire blocks.

Example:

If RABNSIZE=3, MAXISN=5000 on a model 3380 with 668 entries per block (2004/3) results in 8 blocks. The highest ISN expected (before further expansion) is therefore 5343 (668 ◎ 8 - 1).

If RABNSIZE=4, MAXISN=5000 on a model 3380 with 501 entries per block (2004/4) results in 10 blocks. The highest ISN expected is therefore 5009 (501 ◎ 10 - 1).

If the Adabas nucleus requires an additional extent for a file when executing N1 commands, the allocation routine attempts to locate a new extent of 25% of the current size:

- If an unused extent between 25% and 28% can be found using the free space table (FST), that space is taken immediately;

- If only longer extents are available in the FST, a new extent of exactly 25% is taken;

- If only smaller extents are available in the FST, the longest available extent is taken;

- If an additional AC extent is required, and the maximum has already been assigned, Adabas will return an appropriate response code to the calling program;

- If a file has the attribute "one AC extent only" (e.g., if the file is an expanded file), an attempt to allocate a second AC extent will cause a response code.

Normal Index (NI), Upper Index (UI), Data Storage (DS)

For the purpose of allocating a new extent, the following formulas are used:

$$Z1 = MIN\left(2 * B, (E-U) * B/U\right)$$

$$Z = MIN\left(MAX(Z1, B/8 + 10), 1000000\right)$$

where

 B number of blocks currently allocated.

 E highest ISN expected.

 U highest ISN currently allocated.

If an extent found in the FST is contiguous with the end of a previous extent, it is allocated for a maximum of "Z" blocks.

If *no* such extent can be found in the FST

■ but an extent between Z and 9 * Z/8 is found, it is allocated.

■ but an extent with more than 9 * Z/8 blocks is found, then a new extent is allocated with exactly "Z" blocks.

■ the longest extent in the FST is allocated as the new extent.

Additionally, if the MAXNI, MAXUI, or MAXDS parameter is specified for the current file, the nucleus allocates no more than the specified maximum number of blocks for the NI, UI, or DS, respectively.

Space Allocation with the ADADBS Utility

ADD/INCREASE Associator, Data Storage

If the physical extent for the Associator or Data Storage has been exhausted, the ADD or IN-CREASE function (using Adabas Online System or the ADADBS utility) may be used to provide additional physical space.

The ADD function requires the allocation of an *additional* dataset to the Associator or Data Storage. The new dataset may be located on the same or a different device type than those currently in use. Both the Associator and Data Storage may consist of no more than five datasets each.

The INCREASE function results in the physical extension of an *existing* dataset. The new space must, however, first be formatted using the ADAFRM utility. There is no restriction on the number of times the INCREASE function may be used.

Following an ADD function, the new dataset must be formatted using the ADAFRM utility before it can be used, and the appropriate changes must be made to all Adabas job control as described in the Adabas Operations documentation.

After increasing Data Storage four times, it may be necessary to run the REORASSO function of the ADAORD utility to reorder the Data Storage space table (DSST) to a single extent and allow four more increases in Data Storage.

To permit formatting or reordering, the nucleus session terminates automatically following an ADADBS ADD or INCREASE operation.

ALLOCATE Function

The ALLOCATE function (Adabas Online System or ADADBS utility) may be used to allocate an extent of a specific size for any of the following file components:

- Data Storage
- address converter
- normal index
- upper index

It is also possible to specify where the extent is to be allocated by specifying a starting RABN.

Using this function, the DBA may, based on knowledge of the projected size of a file, allocate extents of a specific size, rather than having Adabas perform the assignment. This may avoid having Adabas allocate an extent which is too small or too large (see ADALOD utility). MAXNI/MAXUI and MAXDS values in effect for the accessed file are not checked.

DEALLOCATE Function

The DEALLOCATE function (Adabas Online System or ADADBS utility) may be used to deallocate an extent allocated for any of the following file components:

- Data Storage
- address converter
- normal index
- upper index

It is also possible to specify where deallocation is to begin by specifying a starting RABN. The deallocated space is returned to the free space table (FST).

DELETE Function

The DELETE function (Adabas Online System or ADADBS utility) causes an existing file to be deleted from the database. All space which was assigned to the file is returned to the free space table and is available for reuse. DELETE can delete complete expanded files only.

RECOVER Function

The RECOVER function (Adabas Online System or ADADBS utility) may be used to recover space which was allocated during an execution of the ADAINV or ADALOD utility which terminated abnormally. The recovered space is returned to the free space table and is available for reuse.

REFRESH Function

The REFRESH function (Adabas Online System or ADADBS utility) results in the setting of a file status to 0 records loaded and 1 extent allocated to each file component. Any additional extents other than the first extent are returned to the free space table.

RELEASE and UNCOUPLE Functions

The RELEASE and UNCOUPLE functions (Adabas Online System or ADADBS utility) results in the deletion of an inverted list or physical coupling lists. The space used for the list can be recovered only by using ADAORD. When releasing a descriptor for an expanded file, each component file must be released individually. ADADBS displays a message whenever a descriptor of an expanded file is being released.

Space Allocation with the ADAINV Utility

COUPLE/INVERT Functions

The COUPLE and INVERT functions (Adabas Online System or ADAINV utility) may result in the assignment of additional blocks for the NISIZE file component (but not DSSIZE or MAXISN). This occurs if the available space becomes full during processing of the input data.

In such a case, if there are any index blocks freed during deletion by the nucleus, these blocks are reused. Then, if available, a range of blocks in the free space table whose size is within the range M1 through M2 will be taken.

M1 and M2 are computed as follows:

```
M2 = M1 + M1/8
M1 = MAX (A2, NIB/4 + KZ)
```

where

 KZ   zappable value (default = 10)
 NIB number of NI blocks in use

and

```
A2 = MIN (A1, NIB * 2)
A1 = IUN * NIB/IUS
```

where

 IUN number of unused ISNs
 IUS   number of used ISNs

When inverting a descriptor for an expanded file, each component file must be individually inverted. ADAINV displays a message whenever a descriptor of an expanded file is being inverted.

Space Allocation with the ADALOD Utility

LOAD Function

The LOAD function of the ADALOD utility is used to load a file into a database.

DSSIZE Parameter

The number of blocks or cylinders specified with the DSSIZE parameter is allocated and assigned to the first DS extent at the beginning of ADALOD execution.

The DSRABN and/or DSDEV parameters may be used to force the allocation to a specific RABN and/or device.

If during processing of the input data, this first allocated extent becomes full, a search is made for a range of free blocks in the free space table whose size is within the range M1 through M2.

M1 and M2 are computed as follows:

```
M2 = M1 + M1/8
M1 = MAX (A2, DSB/4 + KZ)
```

where

 KZ   zappable value (default = 10)
 DSB number of DS blocks in use

and

```
A2 = MIN (A1, DSB * 2)
A1 = IUN * DSB/IUS
```

where

 IUN number of unused ISNs
 IUS   number of used ISNs

If enough space is found in the free space table and that space follows immediately an already allocated extent, this space is added to the end of the extent. In this case no new extent is allocated.

If a new extent is needed, the free space table is scanned and the number of blocks needed to satisfy the size of M1 through M2 is taken for the new extent. Up to five extents are possible. If space is not available, ADALOD ends with an error message.

MAXISN Parameter

The MAXISN value is converted into a number of blocks and rounded up to a full block boundary. This range of blocks is allocated at the beginning of ADALOD execution and is assigned to the first address converter extent for the file.

The ACRABN parameter may be used to force the allocation to begin at a specific location.

If during processing of the input data, this first allocated extent becomes full, ADALOD tries to allocate another AC extent whose size is 25% of the sum of all currently existing AC extent sizes.

- If an unused range of blocks is available in the free space table in the range of 25% through 28% of the size currently in use, this range is immediately allocated as a new AC extent for the file;

- If only longer free ranges are available, a new AC extent of 25% is taken from the smallest free range of blocks;

- If only smaller free ranges are available, the largest available is taken.

NISIZE/UISIZE Parameters

At the beginning of its execution, ADALOD allocates and assigns the blocks or cylinders specified by the NISIZE and UISIZE parameters to the first NI and UI extents, respectively.

The NIRABN and UIRABN parameters can be used to force extent allocation to begin at a specific RABN.

If you omit the NISIZE or UISIZE parameters, ADALOD does not initially allocate NI and UI space. Instead, ADALOD waits until all incoming descriptor values have been written to the Temp dataset, and then estimates NISIZE and UISIZE values as follows:

- If no input records were processed:

```
NISIZE = Number of descriptors +1
UISIZE = 2 blocks
```

- If input records were processed:

  For each descriptor in the file, up to 16 temp dataset blocks are selected and read. The contents of these blocks are sorted and estimated to the total amount of temp blocks used for this descriptor.

  The chosen algorithm returns the NISIZE and UISIZE values for each descriptor, which ADALOD adds together and then multiplies by the factor "K", which is

```
K = (MAXISN - MINISN + 1) / number of records loaded
```

  If, during operation, ADALOD determines that the resulting value is not enough, ADALOD allocates subsequent extents during its run. The sizes of these extents are computed in the same way as for additional DSSIZE extents, as described above.

UPDATE Function

The UPDATE function of the ADALOD utility performs a mass add/delete of records to/from an existing file, reorganizes and (if necessary) expands the Associator and/or Data Storage space.

The ADALOD UPDATE functions allocates additional AC space if the MAXISN parameter specifies a new, higher maximum ISN value-even if the restructuring of the AC, NI and UI performed by UPDATE results in more unused current space. ADALOD UPDATE adds Data Storage space if the current Data Storage space cannot hold the new records.

MAXISN Parameter

If a MAXISN value is specified for the UPDATE operation that is greater than the current value for the file, the difference between the old and the new MAXISN setting is computed. The number of AC blocks to satisfy this amount is then allocated from the free space table as an additional extent. The ACRABN parameter may be used to force the allocation to begin at a specific location.

If during processing of the input data the current AC and/or Data Storage extent becomes full, ADALOD tries to allocate another AC and/or Data Storage extent whose size is 25% of the sum of all currently existing AC and/or Data Storage extent sizes.

- If an unused range of blocks is available in the free space table in the range of 25% through 28% of the size currently in use, this range is immediately allocated as a new AC extent for the file;

- If only longer free ranges are available a new AC extent of 25% is taken from the smallest free range of blocks;

- If only smaller free ranges are available, the largest available is taken.

Space Allocation by the ADAORD Utility

ADAORD reorders the respective Adabas Associator component (AC, NI/UI, DSST) and Data Storage to reclaim unusable space for reuse. Although ADAORD functions affect only the selected component files of an Adabas expanded file, there is no change to the logical consistency of an expanded file; therefore, ADAORD does not have to be performed on each component file of an expanded file, unless desired.

Function	Accessed Table Types
REORFASSO	AC, NI, UI
REORASSO	AC, NI, UI, DSST
REORFDATA, REORDATA	DS
REORFILE, REORDB	AC, NI, UI, DS

For each accessed file and for each accessed table type (depending on the function), the following action is taken:

- All used space is returned to the free space table.

- All tables with a specific location (ACRABN, DSRABN, NIRABN, UIRABN) are allocated and assigned as a first extent. The sizes used are either supplied (MAXISN, DSSIZE, NISIZE, UISIZE) or taken from the original file.

- All tables without a specific location are allocated and assigned as a first extent.

If one of the extents become full, the same action is taken as described for the ADALOD (UPDATE) utility (see the previous section).

Space Allocation by ADASAV (RESTORE FILES Function)

If a file to be restored is already present in the database (OVERWRITE parameter must be specified) the space used by all these files is returned to the free space table. If a component file of an expanded file is specified, then all related component files must also be specified.

- RESTORE FILE=...

  For each file to be restored, the original RABNs must be available. ADASAV tries to allocate the required extents at their original position with their original size. If one of these allocations fails, ADASAV will terminate with ERROR-060.

- RESTORE FMOVE=...

  For each file to be restored, at least the amount of original space used will be allocated. The allocation of the first extent for each file table can be forced to a specific location by using one of the optional parameters ACRABN, DSRABN, NIRABN, UIRABN. The sizes of these tables may be increased using MAXISN, DSSIZE, NISIZE, UISIZE.

If space is available, multiple input extents may be compressed in a new single extent. If there is not enough contiguous free space available, ADASAV will split the tables over several new extents (up to five for each table). If such space is not available, ADASAV will terminate with ERROR-060.

Using the Database Status Report to Control Space Use

Database space utilization information can be obtained directly from the database status report produced by executing the ADAREP utility or using Adabas Online System.

In addition to a file allocation map and a block allocation map, this report lists the number of blocks

- used (and unused) for the Associator physical extent (or extents);

- used (and unused) for the Data Storage physical extent (or extents);

- allocated for the Work physical extent;

- used (and unused) for each file for the address converter, normal index, upper index, and Data Storage logical extent (or extents).

See the ADAREP chapter in the Adabas Utilities documentation for a detailed explanation of the information provided on this report.

The DBA should frequently review this report to identify potential space utilization problems.

The next section contains guidelines on problems which may be detected using the status report, and recommendations as to what action should be taken to prevent and/or resolve each problem.

Potential Space Use Problems and Recommended Actions

This section provides a summary of the problems most often encountered concerning database space utilization, and the recommended corrective action to be taken to prevent and/or correct problems.

Full Physical Extents

1. The Associator physical extent is nearly or completely full.

    - The physical extent may be increased (see ADADBS utility, INCREASE function);

    - A new physical extent may be added (see ADADBS utility, ADD function);

    - The Associator may be reordered using the ADAORD utility. This will be of benefit only if a large amount of Associator space fragmentation exists;

    - Unused file extents can be release using the ADADBS DEALLOCATE function;

    - Any Adabas files no longer required may be deleted (see the ADADBS utility; DELETE function);

    - Any file coupling lists no longer needed may be deleted (see the ADADBS utility, UNCOUPLE function);

2. The Data Storage physical extent is nearly or completely full.

    - The physical extent may be extended (see the ADADBS utility, INCREASE function);

    - A new physical extent may be added (see the ADADBS utility, ADD function); this is recommended only when the new extent is on a new device type.

    - Data Storage may be reordered (see the ADAORD utility, REORDATA function). This will be of benefit only if a large amount of Data Storage space fragmentation exists, or the Data Storage padding factor is decreased;

    - A given file may be reordered (see the ADAORD utility, REORFILE function);

    - Any Adabas files no longer required may be deleted (see the ADADBS utility, DELETE function).

Maximum Physical Extents Reached

1. The maximum of five Associator physical extents has been reached.

   ■ The last extent can be increased using the ADADBS INCREASE function;

   ■ The Associator can be reordered by executing the ADAORD REORASSO function;

   ■ All files can be unloaded using the ADAORD RESTRUCTURE function and then reloaded into a larger database using ADAORD STORE.

2. The maximum of five Data Storage physical extents has been reached.

   ■ The last extent can be increased using the ADADBS INCREASE function;

   ■ Data Storage can be reordered (see the ADAORD utility, REORDATA function). This will result in the elimination of Data Storage space fragmentation;

   ■ All files can be unloaded using the ADAULD utility and then reloaded into a larger database.

Maximum Logical Extents Reached

1. The maximum logical extents for the address converter, normal index, or upper index for a file has been reached.

   ■ The REORFILE or REORFASSO function of the ADAORD utility can be executed to reorder all Associator entries for the file.

   ■ ISN reusage can be invoked using the ADADBS utility.

   ■ The file can be unloaded, deleted, and reloaded.

2. The maximum logical extents limit for either Data Storage or the Data Storage space table for a file has been reached.

   ■ The file (and all other files) can be reordered using the REORFDATA or REORFILE function of the ADAORD utility. This condenses multiple Data Storage extents into fewer extents.

   ■ The file can be unloaded, deleted, and reloaded.

25 Database Monitoring and Tuning

The DBA's tasks in the area of monitoring and tuning are

Monitoring Resource Use

The DBA is responsible for monitoring the database environment on a continuing basis to ensure that an efficient level of service is provided while maintaining database integrity.

The DBA should implement a set of procedures designed to foresee degradation before the event and to adjust the operation or design of the database in an orderly and controlled way. This set of procedures includes

■ identifying potential sources of degradation;

■ establishing tools for monitoring database performance; and

■ controlling the implementation of adjustments.

Reporting on Resource Use

The DBA should report regularly on database use and performance to both data processing and user management. The reports should be factual, but should also include recommendations for tuning the database environment. It should be remembered that tuning, while benefiting the organization as a whole, may adversely affect the service received by one or more users. Any decision on tuning should, therefore, be made by all affected users.

Monitoring Database Controls

The DBA should establish appropriate controls and monitor them to ensure the integrity of the database.

Computer-generated control totals can be checked and cross-footed between computer processing runs or generated reports. Batch responses (or inquiries) may include such information as the exact run time, search parameters, time of last update of data, and the primary parameter controls. This increases the confidence level and helps to ensure the integrity of the database.

The problem of control totals takes different forms at different installations. Although hard and fast rules are not possible in this area, some general guidelines can be given.

The DBA needs to ensure that proper consideration is given to the following areas in the design of each application system that will use the database:

- What controls can be checked on every batch update run? For example, record counts, additions, deletions, updates.

- What controls require a full file pass to check them? For example, value field hash totals.

- What input transactions, Adabas logs, etc., should be retained in order to be able to recover when control totals are found to be wrong at the end of a given period?

- Are "localized" control totals (that is, by branch, product group) of any use in identifying the areas affected by a file control total error?

Performance Management, Statistics, and Tuning

The following table illustrates some of the monitoring statistics that may be used and what adjustments to (or tuning of) the database environment may result.

Changes in....	May require tuning of				
	database structure	access method used	hardware or software configuration	processing priority	disk storage allocation
terminal and line traffic		Y	Y	Y	Y
response times (application performance)	Y	Y	Y	Y	Y
access totals by user and descriptor	Y	Y			Y
database size	Y	Y	Y		Y
database growth rate	Y	Y	Y		Y

When any alteration is made to a production database, care must be taken to ensure a continued high level of reliability and integrity. Whatever the change, the DBA must make sure that the decision is the right one and that it is properly and accurately implemented. He should retain absolute control over the tuning process and ensure that it follows the formal acceptance procedures.

The DBA must be careful not to overreact to changes in the items listed in the table. A sudden change in line traffic, response times, etc., may only be temporary. It is important to determine whether the change represents a permanent trend or a temporary disturbance to the normal way of operating.

The table can be used to determine what tuning may be necessary when a new project will cause a significant change in terminal and line traffic, response times, etc. The DBA can then act in advance to minimize these effects before the new application system is implemented.

Adabas Session Statistics

The statistics printed at the end of each Adabas session may be used to monitor Adabas performance. Specifically, the session statistics comprise

- input/output (I/O) statistics;

- command statistics; and

- pool/queue usage statistics.

Input/Output Statistics

The following I/O statistics are provided:

I/O Counts (Including Initialization)

	Reads	Writes
ASSO	50	21
DATA	2388	2184
WORK	9	1385
PLOG	9	1603
CLOG	0	0
TOTAL:	2456	5193
LOG. READS	33899	
BUFFER EFF.	13.9	

The input/output (I/O) counts represent the number of physical I/Os executed during the session to the Associator (ASSO), Data Storage (DATA), Work (WORK), the data protection log (PLOG), and the command log (CLOG).

Also provided are the number of logical reads issued for the buffer pool (LOG. READS) and the buffer efficiency (BUFFER EFF.) which is the number of logical reads divided by the number of Associator and Data Storage reads. The higher the value for buffer efficiency, the more efficient is buffer pool usage. If the value is less than 10, the DBA may wish to increase the size of the Adabas buffer pool (see the Adabas Operations documentation, the ADARUN LBP parameter description).

Distribution of ASSO/DATA I/Os by VOL-SER Number (Excluding Initialization)

VOL-SER	HIGH RABN	COUNT
ADA003	(ASSO: 894)	38
ADA003	(ASSO: 2544)	6
ADA003	(DATA: 894)	0
ADA003	(DATA: 1344)	4572
TOTAL:		4616

The distribution of I/Os for the Associator and Data Storage per physical volume is also provided. The data provided are the highest RABN accessed/updated (HIGH RABN) and the number of I/Os (COUNT). The DBA can use this data to determine if any adjustments are necessary to the buffer pool parameters and/or to the physical allocation of the database.

Command Statistics

In the following example, command statistics are provided for a session in which Adabas executed 12,687 calls in five threads.

Distribution of Commands by Source

The following table shows the source of commands for the session: either from the same environment (local) or from a remote environment across a network:

Source	Number
REMOTE LOGICAL	0
REMOTE PHYSICAL	0
LOCAL LOGICAL	0
LOCAL PHYSICAL	12,686

Distribution of Commands by Thread

The following table shows the thread activity for the session:

Thread	Number
1	7,328
2	2,728
3	1,240
4	814
5	541

Thread	Number
TOTAL:	12,651

If the thread with the highest number has an activity count greater than zero it can be assumed that the Adabas nucleus would be able to process a larger number of commands if the number of threads were increased. Increasing the number of threads would prevent commands from waiting in the command queue for selection.

Distribution of Commands by File

The following table shows the distribution of commands by file:

File	Number
0	4,247
1	8,404
TOTAL:	12,651

Commands that are not file-related (e.g. BT, ET) are counted against file 0.

Distribution of Commands by Type

The following table shows the distribution of commands by command type:

Command Type	Number
A1/4	4,198
ET	4,191
L1/4	4,242
OP	56
TOTAL:	12,687

The command type UC indicates "privileged call" issued by Adabas utilities.

> **Note:** The command type REST indicates commands such as C1, C5, RI and HI.

Additional Session Statistics

```
THERE WERE      56 USERS PARTICIPATING
MOST CALLS (    57) INITIATED BY USER user ID
MOST I/O-S   (    14) INITIATED BY USER user ID
MOST THR.-TIME (04:16:32) WAS USED BY USER user ID
```

28   Formats had to be translated

0   Formats had to be overwritten

0   Autorestarts were done

20   Throw-backs due to ISN problem

16   Throw-backs due to space problem

186 Buffer-flushes were done

Formats Translated/Overwritten

Adabas read and update commands require a Format Buffer that specifies the fields to be read or updated. This Format Buffer is interpreted and converted into an internal Format Buffer by Adabas, which enters each resulting internal Format Buffer into the internal Format Buffer pool. Each internal Format Buffer is identified by a combination of user and command IDs.

For each new read/update command, Adabas looks to see if a user ID/command ID entry is already present in the format buffer pool. If not, Adabas translates the command's new format buffer and enters it into the pool. Once the format buffer pool becomes full, an existing entry must be overwritten to accommodate a new entry.

The format translation process is CPU intensive. Therefore, the DBA should ensure that an excessive number of format overwrites are not occurring by doing the following:

1. Ensure that user programs are making correct use of command IDs; that is, using non-blank command IDs when appropriate and releasing command IDs when no longer needed. For further information on command ID use, refer to the Adabas Command Reference documentation.

2. Consider increasing the size of the internal format buffer pool (with the ADARUN LFP parameter, described in the Adabas Operations documentation).

The Adabas nucleus produces statistics on format translations and format overwrites at the conclusion of each session. The Adabas operator command DSTAT may also be used to obtain this information.

Autorestarts

The number of Autorestarts performed during the session.

Command Throwbacks

The number of times a command could not be executed because the Adabas nucleus was waiting for

- an available ISN; or
- Adabas work pool space.

In such an event, the command is "thrown back" into the command queue for processing at a later point in time.

If either of these numbers is greater than zero:

1. adjust the ratio between the ADARUN LWP (work pool size) and LS (sort work area) parameters;

2. increase the size of the Adabas work pool (ADARUN LWP parameter);

3. evaluate ADARUN TT (transaction time limit) parameter;

4. check application program hold logic;

5. increase the Adabas hold queue size (ADARUN NH parameter); and

6. use superdescriptors to reduce complexity of search commands.

The ADARUN parameters are described in the Adabas Operations documentation.

Buffer Flushes

The number of buffer flushes performed during the session.

The Adabas buffer pool represents a virtual database that is shared by all active users. It contains the most frequently used Associator and Data Storage blocks, and its purpose is to minimize physical I/O activity.

The size of the buffer pool is determined by the ADARUN LBP parameter. LBP should be set as large as possible with the restriction that setting too large a value may cause excessive paging by the operating system.

Buffer and Queue Statistics

Session statistics include the maximum buffer and queue use during the session. These statistics are presented for all buffers and queues (except the buffer pool) for which "high-water marks" can be computed. The following table shows high-water marks for a sample session:

Pool Area	ADARUN Parameter	High-Water Mark	%
AB	NAB = 10	12032	29
CQ	NC = 20	3648	95
DUQ	LDEUQP = 5000	500	10
FI	LFP= 12000	1760	14
HQ	NH = 100	552	23
SC	LCP= 10000	0	0
TBI	LI = 10000	0	0
TBS	LQ = 10000	0	0
UQ	NU = 20	4880	86
UQF	NU = 20		
WORK	LWP = 14000	70464	50
XID	XID = 0	0	0

> **Note:** The UQF is the user queue extension that holds the file list. The size of its pool is computed using the UQ pool size.

The high-water marks are provided together with the applicable ADARUN parameter setting that was in effect for the session.

The DBA should monitor each high-water mark and, if necessary, make adjustments to the appropriate ADARUN parameters.

Command Logging

Adabas command logging may be used to generate information on all the commands issued by users to Adabas. Some of the information provided is

- user identification;
- time of day;
- the command used;
- the file accessed;
- the record accessed;

■ the Adabas response code received;

■ the time required for the command to perform.

Command logging is controlled by the ADARUN parameter LOGGING.

26 Error Handling and Message Buffering

This chapter covers the following topics:

Overview

The error handling and message buffering facility helps implement 24X7 operations by analyzing and recovering from certain types of errors automatically with little or no DBA intervention. It also generates additional information so that the error can be diagnosed by the user and by Software AG.

The ADARUN SMGT parameter is set to activate the facility; if message buffering is to be used, the ADARUN MSGBUF parameter is used to size the buffer. The wrap-around message buffer collects Adabas messages for later review by Adabas Online System in case online access to the console or to DDPRINT messages becomes unavailable. The buffer aids problem analysis and performance tuning.

The error handling functions of the facility are implemented as operands of the operator command SMGT and can be invoked from the operator console or from Adabas Online System. See the Adabas Operations documentation for detailed information.

The current implementation of the facility makes it possible for the nucleus to protect itself or provide additional error recovery information for

- a parameter error 31 : autorestart error;

- a parameter error 73 : checkpoint file is full during initialization;

- non-response codes by capturing pertinent areas of storage to aid in the diagnosis; and

- program interruptions by providing additional dump areas.

Range of Operations

User Exit Failures

User exits and hyperexits that are essential to the operation of the Adabas nucleus can be marked as critical (the default) or not using one of the operands of the SMGT operator command:

- If a user exit is defined as *critical*, is is not affected by the error handling and message buffering facility: an abnormal termination in it causes the Adabas nucleus to terminate abnormally as well.

- If the user exit is defined as *notcritical* and an abnormal termination occurs in it, the facility maintains an active Adabas nucleus, optionally refrains from invoking that exit, takes a dump of the nucleus at the point when the exit failed, and issues messages to the system log to inform

the DBA of the problem. The DBA can then examine the diagnostic information, use it to fix the problem, then load and reactivate the corrected exit using operands of the SMGT operator command.

> **Note:** If an Adabas exit attaches a subtask, the subtask is not protected by the error handling and message buffering facility.

Recovery or Plug-In (PIN) Routines

The extensions (plug-in routines or "PINs") are designed to analyze and, in some cases, determine the cause of an abend while allowing the nucleus to continue processing. The PIN determines whether it is safe to allow the nucleus to continue processing and prints appropriate messages to notify the DBA when this is the case.

The PIN routine user exit ADASMXIT can be used to obtain additional information about response codes and abends. The user exit allows you to specify particular response codes or response code/subcode combinations to be monitored. Once you have modified the user exit, you can reload it and make your changes effective without bringing the database down.

Each plug-in (PIN) service routine handles a predefined condition when encountered, allowing the Adabas nucleus to

- remain active when it otherwise would terminate abnormally; or
- print extended error diagnostics as an aid to error recovery.

Based on its execution, a PIN module can either transfer control to the Adabas nucleus so that it can resume normal processing-usually with a response code-or it can return control to the error handling and message buffering facility, allowing the Adabas nucleus to terminate abnormally.

While the PIN is executing, most Adabas functionality is available to the PIN as the registers at the time of the abnormal event are available. The PIN decides whether the nucleus should remain active.

A PIN can also be used to format an intelligent dump in a number of circumstances to help debug a particular response or abend code.

If the PIN determines that the nucleus is to remain active, the PIN sets a response code.

PIN Processing

In the event of an abnormal termination or nonzero response code, the error handling and message buffering facility looks for a PIN routine first for the specific condition and location detected, then for the specific condition, and finally for the location (any condition). If an appropriate PIN is found, it is invoked; otherwise, the nucleus is terminated.

If a PIN routine is invoked and it

- handles the condition, processing then continues with no further intervention from the error handling facility.

- cannot handle the condition, the PIN returns control to the error handling facility and the nucleus is terminated.

For a response code error, the error handling facility first determines whether the response code is one that it monitors.

- If the answer is no, the PIN returns control to the nucleus and the response code is returned to the user normally.

- Otherwise, the appropriate PIN is invoked to print additional information about the response code that to help resolve the problem. The PIN then returns control to the nucleus and the response code is returned to the user normally.

Once the PIN has processed the response code, it returns control to the error handling facility so that normal response code processing can continue.

Default PIN Module ADAMXY

The ADAMXY module is the default PIN module comprising the PIN routines that are distributed with Adabas and automatically installed during initialization.

> **Note:** It is possible to disable the default PIN ADAMXY using the SMGT,DELPIN or SMGT,DEACTPIN operator commands.

The following table describes the interrupts that are handled by the PIN routines in ADAMXY. For each interrupt, extended dump formating is provided to aid in error analysis:

Code	Exception Type	The processor . . .
01	Operation	is about to execute an instruction that has an invalid operation code.
02	Privileged Operation	attempts to execute a supervisory instruction while in problem state.
03	Execute	interrupts a program deliberately to aid problem diagnosis.
04	Protection (also Segment and Page)	attempts to alter system or hardware storage; access fetch protected system or hardware storage; or access or modify storage that is not allocated. Requires record/file-level locking with user notification in job

Code	Exception Type	The processor . . .
		log. Note that code 16 (segment exception) and code 17 (page exception) are also presented to the error handling facility as code 04.
05	Addressing	encounters a reference to an invalid read address.
06	Specification	attempts to either set or branch to an old address or an instruction that required a field to be aligned but did not have an aligned argument.
07	Data	encounters a corrupted data record, probably a field that should be packed decimal is not.
08	Fixed Point	a high-order carry occurs; or high-order significant bits are lost in a fixed-point add, subtract, shift, or sign-control operation.
09, 11, 15	Divide	encounters a zero divisor in a division instruction; probably a corrupted record. Code 9 is for binary; code 11 is for packed decimal; and code 15 is for floating point arithmetic.

The message

```
*****DEFAULT PIN OUTPUT*************
```

is generated whenever the default PIN ADAMXY is invoked. This is followed by all output concerning the program interrupt processing of ADAMXY. The message

```
*****END OF DEFAULT PIN OUTPUT*****
```

-is generated whenever ADAMXY is completed.

Additional PIN Modules Provided

Some of the PIN modules discussed in this section are delivered with selectable units of Adabas. They are established automatically when the relevant server component initializes at nucleus startup:

PINAFP
PINATM
PINAVI
PINCOR
PINSAF

The remaining PIN modules discussed in this section are included with Adabas but are not part of ADAMXY and are not automatically installed at Adabas initialization:

PINAUTOR
PINOPRSP
PINRSP
PINUES

PINRSP and PINUES are installed using the SMGT,ADDPIN=module-name command when the nucleus is active. Because PINAUTOR and PINOPRSP are invoked during system initialization when operator commands are not available, they are activated by renaming a particular module in the Adabas load library:

- renaming NOOPRSP to PINOPRSP activates that PIN;

- renaming NOAUTOR to PINAUTOR activates that PIN.

See the Adabas Installation documentation for more information about installing PIN modules.

PINAFP

PINAFP is delivered with the selectable unit Adabas Fastpath. It is established automatically when the Adabas Fastpath server component initializes at nucleus startup (ADARUN FASTPATH=YES).

In the event of a program interrupt (see the table in the section *Default PIN Module ADAMXY*) in the Adabas Fastpath server component, control is passed to PINAFP, which formats and prints the main memory areas used by the component. These diagnostics are written to the DDPRINT dataset with the title

```
ADABAS FASTPATH - memory-area-name : SNAP BY PINAFP
```

PINAFP then returns control to the error handling and message buffering facility so that Adabas can terminate abnormally.

If necessary, PINAFP can be activated and deactivated. However, after PINAFP is reactivated, it will not be reestablished until the next nucleus session.

PINATM

PINATM is delivered with the selectable unit Adabas Transaction Manager (ATM). It is established automatically when the ATM job initializes (ADARUN DTP=TM).

In the event of a program interrupt (see the table in the section *Default PIN Module ADAMXY*) in the ATM logic, control may be passed to PINATM, which formats and prints the main memory areas used by ATM. These diagnostics are written to the DDPRINT dataset with the title

```
ADABAS TRANSACTION MANAGER - memory-area-name : SNAP BY PINATM
```

PINATM then returns control to the error handling and message buffering facility so that Adabas can terminate abnormally.

If necessary, PINATM can be activated and deactivated. However, after PINATM is reactivated, it will not be reestablished until the next ATM session.

PINAUTOR

If NOAUTOR has been renamed to PINAUTOR in the Adabas load library and a parameter error 31 occurs during autorestart, PINAUTOR acquires control. PINAUTOR attempts to identify the file that is in error and exclude it from autorestart if possible.

Before excluding a file from autorestart processing, PINAUTOR checks that

- the file is not the security or checkpoint file; and

- the response code is not 9, 65, 72, 88, 97, 99, 148, or 151 as these are not valid for the exclusion process.

Additionally, PINAUTOR checks whether certain files or particular response codes for a particular database are designated as ineligible for exclusion. For example, it may be senseless to start a database without the particular file on which it depends. You can customize ADASMXIT to include the files and the response codes that cannot be excluded. You can also specify the maximum number of autorestarts with exclusions that can be attempted.

The AREXCLUDE procedure is automatically invoked to exclude a file. See the Adabas Operations documentation, ADARUN parameter AREXCLUDE for more information. Note that excluded files may become inconsistent and need to be restored from backup using ADASAV RESTORE.

If a file is excluded from autorestart, an SM-PINAUTOR2 message is generated followed by an ADAN50 message, both indicating the file number of the excluded file.

Whenever PINAUTOR is invoked, the message PINAUTOR OUTPUT is generated followed by messages pertaining to the specific PINAUTOR situation as described in the Adabas Messages and Codes documentation. To indicate that PINAUTOR processing is completed, the message END PINAUTOR OUTPUT is generated.

PINAVI

PINAVI is delivered with the selectable unit Adabas Vista. It is established automatically when the Adabas Vista server component initializes at nucleus startup (ADARUN VISTA=YES).

In the event of a program interrupt (see the table in the section *Default PIN Module ADAMXY*) in the Adabas Vista server component, control is passed to PINAVI, which formats and prints the main memory areas used by the component. These diagnostics are written to the DDPRINT dataset with the title

```
ADABAS VISTA - memory-area-name : SNAP BY PINAVI
```

PINAVI then disables the program in which the interrupt occurred and returns control to Adabas so that Adabas can continue. Disabling the program does not disrupt the Adabas service; however, access to Adabas Vista files may be restricted. In this case, a nonzero response code returned to the user identifies the restrictions.

If necessary, PINAVI can be activated and deactivated. However, after PINAVI is reactivated, it will not be reestablished until the next nucleus session.

PINCOR

PINCOR is delivered with System Coordinator for Adabas Options. It is established automatically when the System Coordinator server component (ADAPOP) initializes at nucleus startup.

If a program interrupt occurs in the System Coordinator server component, control is passed to PINCOR, which formats and prints the main memory areas used by the component.

These diagnostics are written to the DDPRINT dataset with the title

```
COMMON RUNTIME - memory-area-name : SNAP BY SMGT
```

PINCOR then returns control to the error handling and message buffering facility so that Adabas can terminate abnormally.

PINOPRSP

⛔ **Caution:** PINOPRSP makes it possible to initialize a database and operate it without writing checkpoints. If database recovery procedures become necessary, the missing checkpoint information can result in critical errors. To prevent this, you must reorder the checkpoint file immediately after PINOPRSP is invoked and be able to account for all changes in the status of the database between initialization and the reordering of the checkpoint file.

If NOOPRSP has been renamed to PINOPRSP in the Adabas load library and a parameter error 73 occurs during system initialization indicating a checkpoint overflow condition, PINOPRSP is invoked.

The message "PINOPRSP OUTPUT" is generated indicating that PINOPRSP has been invoked. PINOPRSP then generates a message warning the DBA that Adabas will activate even though the checkpoint file is full:

```
response code  INTERCEPTED BY PINOPRSP BECAUSE THE CHECKPOINT FILE IS
FULL. THE ADABAS NUCLEUS WILL ACTIVATE BUT THE CHECKPOINT FILE NEEDS
TO BE REORDERED AS SOON AS POSSIBLE.
```

The "response code" is either 75 or 77 in this case. No checkpoint is written but the nucleus activates. Corrective action needs to be taken as soon as possible. The message "END PINOPRSP OUTPUT" is then generated to indicate that PINOPRSP processing is completed.

PINRSP

The SMGT,ADDPIN=PINRSP operator command activates PINRSP, which provides extended information to aid in diagnosing a response code.

> **Note:** Only response codes set by Adabas can be logged. A response code such as 22 (invalid command code), which is set by the Adabas SVC before it reaches Adabas, is not logged.

If PINRSP is installed without modifying the Adabas PIN routine user exit ADASMXIT, all response codes are logged. You can customize ADASMXIT to

- include specific response codes or response code/subcode combinations;
- indicate the number of times a particular response code can be monitored.

When a nonzero response code is encountered, PINRSP acquires control. The message "PINRSP OUTPUT" is generated to indicate that PINRSP has control. Depending on the response code encountered, different areas are logged.

With respect to areas logged, five categories of response codes are described:

1. Basic response code logging includes
    - the active thread
    - the FCB if possible
    - the areas that have been GETMAINed and are currently in use
2. Index-related response codes such as 177
    - basic response code logging
    - the index structure from the thread
    - active CQEs
    - buffer pool headers
3. Response codes such as 40 where additional IUB areas may be pertinent
    - basic repsonse code logging
    - IUBs
    - active CQEs

4. Response codes such as 255 where additional attached buffer information may be necessary

   ■ basic reponse code logging

   ■ active CQEs

   ■ attached buffer information

5. Response codes such as 72 where the user queue may be helpful

   ■ basic response code logging

   ■ active CQEs

   ■ user queue

Example:

Rather than obtain a CLOG when you need additional information to diagnose a particular response code, you can modify ADASMXIT to capture the response code, reassemble it, and load it while the nucleus is up. The information is then logged the next time the response code is encountered.

Once you have the information, you can modify ADASMXIT to remove the response code and reload it so that information is no longer captured. Alternatively, you can set ADASMXIT initially to log the information only `n' number of times.

You can also use PINRSP in conjunction with ADASMXIT to suppress the ADAN77 message that is generated for response codes 201, 202, or 203. This may be useful in situations where a new application receives enough security errors to fill the SYSLOG. Although Software AG does not recommend this action, you may temporarily modify ADASMXIT to suppress N77 messages and activate PINRSP with response codes 201, 202 and 203 indicated in ADASMXIT.

If message suppression is activated, the ADAN77 message "Message suppression in effect" is generated and the PINRSP output providing format information related to the response code is suppressed.

Once PINRSP has completed processing, the message END PINRSP OUTPUT is generated.

PINSAF

PINSAF is delivered with the selectable unit Adabas SAF Security (ADASAF). It is invoked automatically when the ADASAF initializes at nucleus startup.

In the event of a program interrupt (see the table in the section *Default PIN Module ADAMXY*) in ADASAF, control is passed to PINSAF, which formats and prints the main memory areas used by ADASAF. These diagnostics are written to a dataset with the title

```
ADABAS SAF INTERFACE - control-block-name : SNAP BY SMGT
```

PINSAF then returns to the error handling facility so that Adabas can terminate abnormally.

> **Note:** For security reasons, PINSAF does not allow Adabas to continue after an abend in ADASAF.

Like other PIN routines, PINSAF can be activated and deactivated. However, after PINSAF is re-activated, ADASAF itself must be restarted before PINSAF will function correctly. Refer to the Adabas SAF Security documentation for more information.

PINUES

PINUES handles Adabas response codes in the context of the universal encoding support (UES) system. PINUES captures input/output errors when trying to

- load an encoding object that does not exist; or
- convert invalid data.

> **Note:**

When used with other PIN routines that handle the same error conditions, the PIN loaded last is called to handle the error. For example:

```
F NUC227,SMGT,ADDPIN=PINRSP
F NUC227,SMGT,ADDPIN=PINUES
```

In this example, PINUES is loaded after PINRSP and therefore handles the error conditions it can handle (response codes 17, 48, and 55). All other response codes are processed by PINRSP.

The message PINUES OUTPUT is generated to show that PINUES has acquired control. The message END PINUES OUTPUT is generated when PINUES processing is completed.

Error Conditions Handled

- Response codes 17 and 48 may occur on an OP command if the ECS objects are not available that are needed to determine the data conversion between user and Adabas file. In this case, PINUES calls ADAMXF with the options IUB and UQE to obtain diagnostic output.
- Response code 55 may occur if ECS returns a response indicating that the conversion or moving of text failed. In this case, the conversion parameters and buffers are snapped to obtain diagnostic output.

Output Produced

Whenever PINUES writes diagnostic information, the following lines are printed on the console:

```
******** P I N U E S  OUTPUT ********'

ADANX1 dbid COMMAND cmd COMMAND ID hex-cid FNR file-number
       RESPONSE adabas-response-code SUBCODE adabas-subcode FLD
field-name'
       TID hex-internal-user-id UID open-userid JOB job-name'
****** END P I N U E S  OUTPUT ******'
```

Session Snapshots

```
18:17:37 ADAN19 00227 BUFFERFLUSH IS  A S Y N C H R O N O U S
18:17:37 ADAN01 00227 A D A B A S  V7.1.0  IS ACTIVE
18:17:37 ADAN01 00227 MODE = MULTI
18:17:37 ADAN01 00227 RUNNING WITHOUT RECOVERY-LOG
18:18:04 ADAI29 OPER CMD: SMGT,ADDPIN=PINUES
18:18:04 ADANTG 00227 PIN MODULE PINUES   LOADED
18:18:04 ADANO2 00227 SMGT COMMAND PROCESSED
18:18:04 ADAN41 00227 1999-01-00 18:18:03 FUNCTION COMPLETED


18:36:33 ADAN7A 00227 ECS ERROR    -2 IN FUNCTION GETHANDL
18:37:21
18:37:21 ******** P I N U E S  OUTPUT ********
18:37:21 ADANX1 00227 COMMAND OP COMMAND ID 00000000 FNR 00014
18:37:21           RESPONSE 017 SUBCODE 023
18:37:21           TID 00000013 UID BLAUTOPF JOB TXG.....
18:37:21 ADAH51 00227 DUMP FORMAT CALLED
```

The following output is produced by the ADAMXF module:

```
18:37:22 ADAH52 00227 DUMP FORMAT COMPLETED
18:37:22 ****** END P I N U E S  OUTPUT ******
18:37:22
18:49:45 ADAN7A 00227 ECS ERROR     54 IN FUNCTION CVFTXTX
18:49:45
18:49:45 ******** P I N U E S  OUTPUT ********
18:49:45 ADANX1 00227 COMMAND A1 COMMAND ID 00000000 FNR 00014
18:49:45           RESPONSE 055 SUBCODE 004
18:49:45           TID 00000017 UID ANDECHS. JOB TXG.....

  ECS CONVERSION PARAMETERS

  0C190BE0+0000  00106570 00000080 00000200 00000000 *................*
  0C190BF0+0010  00000004 0C10ACB4 00000004 0010A6E0 *..............w.*
  0C190C00+0020  00004000 0C190BEC 0000020C 00000000 *................*
  0C190C10+0030  001065AD 0C190BE4 00000000 00000000 *.......U........*

  ECSE FROM ENCODING 3026 TO ENCODING 3035
```

```
   00106570+0000   02000002 00000BD2 00000BDB 00000004 *.......K........*
   00106580+0010   00000000 001062C8 00106350 001064E8 *.......H.......Y*
   00106590+0020   0000BD20 0000BDB0 0000000C 00000000 *................*
   001065A0+0030   001065AD 0004362C 40000000 00FEFE00 *................*
   001065B0+0040   00000340 40000000 04404000 00000000 *................*
   001065C0+0050   00000000 00000000 00000000 00000000 *................*
   001065D0+0060   00000000 00000000 00000000 00000000 *................*
   001065E0+0070   00000000 00000000 00000000 00000000 *................*
   001065F0+0080   00000000 00000000 02000001 00000BDB *................*


   ECONV INPUT AREA


   0C10ACB4+0000   4141F1F2 40404040 40404040 40404040 *..12............*


   ECONV OUTPUT AREA


   0010A6E0+0000   414150C2 50C350C4 50C550C6 50C750C8 *...B.C.D.E.F.G.H*
   0010A6F0+0010   50C150C2 50C350C4 50C550C6 50C750C8 *.A.B.C.D.E.F.G.H*
       LINES 0010A700 TO 0010A7C0 SAME AS ABOVE
   0010A7D0+00F0   50C150C2 50C350C4 50C550C6 00000000 *.A.B.C.D.E.F....*
   0010A7E0+0100   00000000 00000000 00000000 00000000 *................*
       LINES 0010A7F0 TO 0010E6D0 SAME AS ABOVE

18:49:45 ****** END P I N U E S  OUTPUT ******
```

PIN Routine User Exit

The PIN routine user exit (entry name ADASMXIT) can be used to

- supply parameters to the various PINs. If the exit is not installed, the parameters are set to the default values.

- examine a condition when it is encountered before the PIN routine is invoked so that recovery actions other than those provided by Adabas can be implemented.

The ADASMXIT load module must be located so that it can be loaded by the nucleus, either in the load library concatenation or in a system call library such as the MVS system link list. If you are running either ADASMP or Adaplex+ on OS/390 MVS, the ADASMXIT module must be placed in an authorized load library.

The PIN routine user exit is written in Assembler.

User Exit Inputs

The exit is entered with the following registers set:

R13 Adabas PIN routine save area

R14 Return address / AMODE

R15 Entry point address

R0 Function code:

1 -	nucleus initialization
2 -	abend
4 -	response code
5 -	nucleus termination

R1 Parameter list

+0	address of two user words
+4	address of condition description block (CDB) for functions 2, 4

User Exit Outputs

There are no outputs. Return codes are ignored and all registers other than 15 must be returned unchanged.

Condition Description Block

For each program check, abnormal termination, or response code error, a control block called the condition description block (CDB) is generated that describes the event that occurred, where it occurred, and what the registers and machine state were when it occurred. The CDB is passed to the error handling and message buffering facility for use in determining whether a PIN routine is to be called or whether an Adabas user exit is to be terminated. A PIN routine uses the CDB to obtain information about the occurrence of the condition.

Modifying and Reloading the Exit

The PIN routine user exit may be modified, reassembled, and reloaded with the nucleus active. To load a newly reassembled exit, issue the console operator command

`SMGT,XD=SXnn`   to deactivate the PIN routine user exit

`SMGT,XLOAD=SXnn`   to load the modified version of the exit

`SMGT,XA=SXnn`   to activate the exit

See the Adabas Online System documentation for another way to accomplish this task.

Using the Exit with PINAUTOR

If PINAUTOR is enabled

- without the PIN routine user exit, the maximum number of files that can be excluded from autorestart is 10 (the default) and all files except the checkpoint and security (system) files are eligible for exclusion. All response codes are eligible for exclusion except those that Adabas disallows as a general rule.
- with the PIN routine user exit, you can modify its AUTOPARM to change the maximum number of files that can be excluded from autorestart and prevent specific files and/or response codes from ever being excluded.

AUTOPARM Example

```
AUTOPARM DS  0D
MAXARPIN DC  F'6'       Maximum of 6 files can be excluded from autorestart
BADRSPS  DC  XL1'48'    Response code 72 cannot be excluded from autorestart
         DC  XL29'00'   28 more entries are possible
NOTFILE  DC  XL2'0041'  File number 65 cannot be excluded from autorestart
         DC  XL48'00'   23 more entries are possible
```

Using the Exit with PINRSP

If PINRSP is enabled

■ without the PIN routine user exit, all response codes are monitored with no specific subcode checking. Each response code is monitored a maximum of ten times. The ADAN77 message is not suppressed.

■ with the PIN routine user exit, you can modify the response code to indicate the specific response codes and subcodes that are to be monitored and the maximum number of times each response code is to be monitored. You can set `N77MSG' to YES in conjunction with response code 201/202/203 monitoring to suppress message ADAN77.

Response Table Entry Example

> **Note:** There is one entry per response code up to 255.

```
XL5'000A000000'
.
.... subcodes (up to 3; specified in hexadecimal)
.
.... maximum number of times to invoke PINRSP for response code (default=10)
.
.... 00 don't log, 01 log
```

Example:

The ninth entry in the table corresponds to response code 9.

The entry X'0105020311' indicates that response code 9, subcodes 2, 3, and 16 are logged. Response code 9 is logged a maximum of 5 times.

27 Universal Encoding Support (UES)

> **Note:** UES support requires that you use a version 7 or above Adabas SVC or router.

This chapter covers the following topics:

Overview

The Universal Encoding Support (UES) is a database option that enables Adabas to

- perform data conversions;
- handle wide character encodings;
- set the basis for internationalization tasks such as collation sequences.

Data conversion needs arise when communicating with different systems, i.e., conversion between different code pages for alphanumeric data or conversion of numerical data due to different machine architectures (see also section **Multiple Platform Support**).

Wide character encodings are used in Asian language environments. Due to the need for a large number of different characters, non-single-byte character sets have been defined. In addition, Unicode, a "Universal Character Set", is more frequently used (see also section **Wide Character Encodings**).

A frequently listed internationalization task is searching and sorting data in a language specific order rather than binary order as defined by the encoding (see also section Collation Descriptor Exits in *User Exits*.

Wide-Character Encodings

In most cases, an Asian text character cannot be encoded using a single byte. For example, Japanese with more than 10,000 characters in its set is encoded using two or more bytes per character. Because of the encoding required, these are called double-byte character sets (DBCS) or multiple-byte character sets (MBCS) as opposed to the single-byte character sets (SBCS) characteristic of most Western languages.

Previous versions of Adabas have stored DBCS-encoded data in alphanumeric fields. Problems with this solution include the following:

- the default 'blank' of alphanumeric fields may be different from the blank required for double- or multiple-byte character fields;
- field truncations caused by length overwrites can result in changed or invalid characters because the string is cut off at a byte boundary rather than at a character boundary.

■ client/server applications are difficult to implement when client and server use different encodings for their double- or multiple-byte character sets.

Although version 7 of Adabas continues to support the storage of DBCS-encoded data in alphanumeric fields, it introduces a wide-character (W) field format to store data with a well defined encoding and character set.

The default encoding for Wide format is Unicode for both storage and user. This default can be changed on user and storage level to the encoding appropriate for the intended usage.

In the figure below, the Japanese kana (first two) and kanji (second two) characters are encoded in mainframe modal (mixed) and non-modal (pure)

■ DBCS for use on EBCDIC-based machines

■ JIS for use on ASCII-based machines

and in Unicode, a fixed 2-byte encoding that is more universal than the other encodings and is used as the default encoding in Adabas.

```
          か  な  漢 字                    "kana [and] kanji"

   <SO> , f , o  | X   2   <SI>

    0E  4486 4496 4F58 48F2  0F         IBM-DBCS mixed
        4486 4496 4F58 48F2             IBM-DBCS only

        82A9 82C8 8ABF 8E9A             Shift JIS (MS CP932)
<ESC> $  B  $ +  $ J  4 A   ; z    <ESC>  (  J
 1B   24 42 242B 244A 3441 3B7A     1B    28 4A    JIS

        304B 306A 6F22 5B57             Unicode
```

Wide-Character Encoding Example

Modal encodings shift back and forth between single- and double-byte character encodings. Mixed DBCS strings always start and end in single-byte mode.

Double-byte character only field lengths must be an even number of bytes.

For EBCDIC encodings, the padding or blank character is X'40' or X'4040'. On Hitachi machines, the wide space is X'A1A1' and the single byte space is X'40'. Adabas allows a single byte space to appear in double-byte mode without a mode switch.

Wide-Character Data Support

Adabas supports wide-character data with

■ extended alphanumeric format fields; and

■ wide-character format fields.

For an existing database or file, the encoding is assigned to alpha or wide fields using the ADADBS utility without an unload/reload. The field-level option NV (pass a field unconverted to/from a caller) is available.

Extended Alphanumeric Fields

Adabas extends alphanumeric fields to support wide-character data by defining encoding keys on both the database and file levels: the file level encoding takes precedence over the database encoding. The encoding specifies the format in which the data is to be stored. It is also used as the default format in which data is exchanged with a local user.

The encoding must be compatible with EBCDIC; that is, the space character must be X'40'. For internal processing reasons, only one of the following encoding "families" is supported for a given file:

■ EBCDIC (single-byte character set)

■ mixed host-DBCS

■ host-DBCS with DBCS-only option

Advantages and Disadvantages

The advantages of using extended alphanumeric fields include

■ immediate support of existing databases that contain DBCS data;

■ applications such as Natural continue running without changes; and

■ no logic changes in the Adabas nucleus for calls from the same encoding/architecture since alphanumeric fields do not define an internal coding.

The disadvantage is that DBCS is not a "universal" encoding and unlike Unicode, it does not support all characters used in the world's languages.

Limitations

For an application, all alphanumeric fields have the same encoding. It is not possible to use different encodings for different fields in the same session.

Conversion Considerations

When converting from pure single-byte character encodings, the field length of variable fields may change requiring a shift of the converted record.

Wide-Character Fields

Adabas defines a wide-character (W) format for fields. W format fields are similar to alphanumeric (A) format fields in that encoding keys are defined on both the database and file levels: the file encoding takes precedence over the database encoding. It differs from A field encoding in that

- if no encoding is specified, the default Unicode encoding is used.
- the "internal" encoding specifies the format in which the data is stored.
- the "user" encoding specifies the default format for data presented to the user.

A descriptor is stored (and sorted) with internal encoding.

Advantages and Disadvantages

The advantages of using wide-character (W) fields include the following:

- round-trip problems are avoided because the character set of the local encoding can be a superset of all character sets of user and special encodings;
- space is saved because internal encodings allow the use of UTF-8 when supported by ECS; and
- native Unicode (the user encoding), the standard Java text encoding, can be directly stored and retrieved.

The disadvantages are that

- Natural and other products do not immediately support the new format; and
- support for W format fields currently has the limitations listed in the next section, some of which may be resolved in future releases of Adabas.

Limitations

- For an application, all wide-character (W) fields have the same encoding. It is not possible to use different encodings for different fields in the same session.

- A W field cannot be the source for a phonetic descriptor or hyperdescriptor.

- Format conversions are not possible from numbers (U, P, B, F, G) to W format.

- A W field cannot be part of a coupled field, physical or soft.

- A W field cannot be part of a format selection criterion (conditional format). This limitation is due primarily to the single-byte character encoding of the criteria input (format buffer, search buffer, and utility).

- A W field cannot be part of a security-by-value criterion.

- A W field cannot be used with an edit mask.

- Format buffer literals are handled as unconvertible single-byte character strings.

Special DBCS Format Conversion Rules

To ensure a smooth transition from existing applications that use mixed-DBCS and DBCS-only data, special format conversion rules have been defined:

1. A modal DBCS encoding comprising the superset of single-byte and double-byte characters is treated as "mixed-DBCS" encoding for alphanumeric fields and as "DBCS-only" encoding for wide-character fields.

2. When converting from wide-character "DBCS-only" to the user's alphanumeric "mixed-DBCS" encoding, the encoding difference is ignored.

For example, if the user encoding for both alpha and wide formats is defined as "DBCS" and in the FDT, field AA is defined as alpha and field WW is defined as wide:

Format Buffer	Value in User Buffer
AA[,A]	mixed-DBCS
AA,W	DBCS-only
WW,A	DBCS-only
WW[,W]	DBCS-only

28 Multiple Platform Support

This chapter covers the following topics:

Overview

Prior to Adabas version 7, converting data for Adabas buffers between different machine architectures (ASCII, EBCDIC) was handled by Entire Net-Work. With the increasing use of applications where clients and servers (that is, the databases) have different encodings, it has become necessary to expand the data transfer and conversion capabilities of Adabas itself. To this end, Entire Net-Work determines whether the target database has translation capabilities, and if so, passes the unconverted data on to the database for conversion there.

An additional advantage of translating data within Adabas is that other transport mechanisms can now be supported. For UES-enabled databases, Adabas version 7 supports Entire Net-Work access to the OS/390 or z/OS mainframe database through the TCP/IP protocol from web-based applications or from PC-based applications such as Software AG's Jadabas. See the ADARUN parameters TCP and TCPURL in the Adabas Operations documentation for more information.

Adabas data translation occurs as follows:

- The client application can specify a special encoding and communicate it to the Adabas nucleus at session open (OP command).
- The LNKUES/ADALNK converts Adabas buffer data depending on the architecture of the caller.
- A number of utilities provide for special encoding and architecture settings.

EBCDIC to ASCII and ASCII to EBCDIC translation tables are located in the Adabas Installation documentation. A table listing the encoding keys provided with Adabas version 7 is located in the Adabas Command Reference documentation and the Adabas Utilities documentation.

Encodings

Adabas recognizes four types of encodings that can be specified in parallel:

Encoding	Character string encoding ...
File	is stored and processed internally
Default User	is used as the default for Adabas local call interface requests and for ADACMP DDEBAND.
User	overrides the "default user" encoding for a user session or an ADACMP execution. This is used to adapt to the special needs of a client program.
Collation	is in acceptable sort order. The collation can be defined by language and country standards commonly identified with a "locale" definition.

Since user data does not require conversion, Adabas equates the local "default user" and "file" encoding to increase processing speed. Remote requests with ASCII architecture are converted using the database "default ASCII user" encoding.

Double-byte character sets are converted using the native mainframe EBCDIC architecture encoding: host DBCS from IBM, Fujitsu Ltd., or Hitachi.

Special applications or remote clients select a specific "user" encoding that fits their processing environment at session open.

To ensure round-trip compatibility between architectures and encodings, Adabas uses a file encoding that holds the superset of all characters defined in the "default user" and any specific "user" encodings. For wide-character fields, such a file encoding defaults to the universal character set encoding Unicode.

Collation encoding is defined for a descriptor field. Values for this encoding are obtained algorithmically by calling a collation exit programmed to produce culturally correct sorted keys; that is, a character dictionary. Collation encoding may be defined for both alphanumeric and wide-character fields; the collation encoding/exit is defined on the file level for alpha and/or wide descriptor fields.

ADACOX Conversion Exit

With ADABAS 7.4.2 the new conversion exit ADACOX is available. ADACOX supports *context-sensitive* conversion between Windows-1256 and EBCIDIC Arabic or EBCDIC Farsi code page with UES enabled databases.

While Arabic characters are *unshaped* in Windows-1256, the supported EBCDIC encodings use *shaped* forms depending on previous or following characters. In addition, for certain consecutive characters the combined form is used e.g. LAM-ALEF ligature.

Currently, no support is included for conversion from logical to visual order and vice versa, or symmetric character conversion.

The conversion exit will always be loaded for UES enabled databases.

When a new conversion between two encodings is first used, the exit is queried whether it supports the conversion. If it does, the exit will be called for any such conversion; if it does not, Adabas and/or Entire Conversion Services will do the conversion.

However, the conversions defined in ADACOX need to be backed by corresponding ECS objects; for example, for the conversion 420 to 1256 the character set properties are determined by the ECS objects.

Conversion of High Value in Value Buffer

When performing searches using the S operator, the high value is usually a sequence of X'FF' bytes.

With UES=YES, the source and target code pages control the conversion of data. The conversion of the character X'FF' depends on its mapping to the target code page. It is therefore possible that the X'FF' will not remain the X'FF' in the converted value.

For example, when converting from 819 (ISO 8859-1 Latin1) to 37 (EBCDIC), the Latin small letter 'y' with diaeresis is mapped from X'FF' to X'DF'. As a result, searches find fewer or even no records.

Adabas version 7.4 solves this problem as follows: with UES=YES and Alpha (or Wide) conversion, all FROM-TO Search/Logical Read Criteria are handled in such a way that in the TO criterion the high value characters at the value end:

- are preserved when converted into the internal search value, and

- are excluded from value conversion.

> **Note:** This solution is not implemented for the value operators (EQ, GT, GE, LE, and LT). It is limited to the TO value of FROM-TO search criteria (S operator). This applies to alpha and wide-format fields and to the Alpha/Wide format parts of Super and Sub Descriptors.

Data Translation Restrictions

The following restrictions of Entire Net-Work are continued with Adabas translation:

- compressed records (FB=C) are not converted.

- text literals are not converted and are passed as is. When reading records, a literal is returned unchanged (for example, FB AA, `-do not convert-',BB).

- prefetch option P is not supported in conversion.

- ET data is not converted. When reading, ET data is padded with EBCDIC blanks.

Additional restrictions imposed by Adabas include the following:

- for all C`Xn' command codes used by CSCI, only the control block is converted; not the buffers. This applies only for Adabas version 7 servers.

- Entire System Server (NPR) / XCOM applications are not included in the scope of Adabas translation. Those applications need to do their own translation.

- OS/2 unpacked numbers sign X`Dn' is not used and therefore, is not supported.

■ Adabas does not provide user translation exits for field-level translation. Such exits are provided by Entire Net-Work.

Platform Considerations

Although differences between Adabas versions running on various platforms are gradually being reduced, the following considerations apply when porting applications:

	Mainframe	Open Systems	OpenVMS
Fixpoint field length	2 and 4 only	1,2,4,8	1,2,4,8
Binary superdescriptor format default is U (unpacked)	No	Yes	Yes
Signed binary superdescriptor	Yes	No	No
Binary superdescriptor format conversion	Yes	No	No
Superdescriptor with MU and PE fields	Yes	No	No
Superfields and subfields	Yes	No	No
Superdescriptor with floating-point format parents	No	Yes	Yes
Maximum length of unpacked fields	29	29	27
Maximum length of packed fields	15	15	14
Prefetch option for read and ET/BT commands	Yes	No	No
Long alpha (LA) field option	Yes	No	No
Field arithmetic update option in format buffer	No	Yes	Yes
MC command	No	Yes	Yes
Hyperfield value generation from value buffer	No	Yes	Yes

Additionally, user data provided in mainframe ADALNK user exits is not sent to ASCII machines.

29   Adabas Online System Demo Version

A demo version of Adabas Online System (AOS) and access to the online services for selected other Adabas products and facilities is included with Adabas as shown in the following diagram:
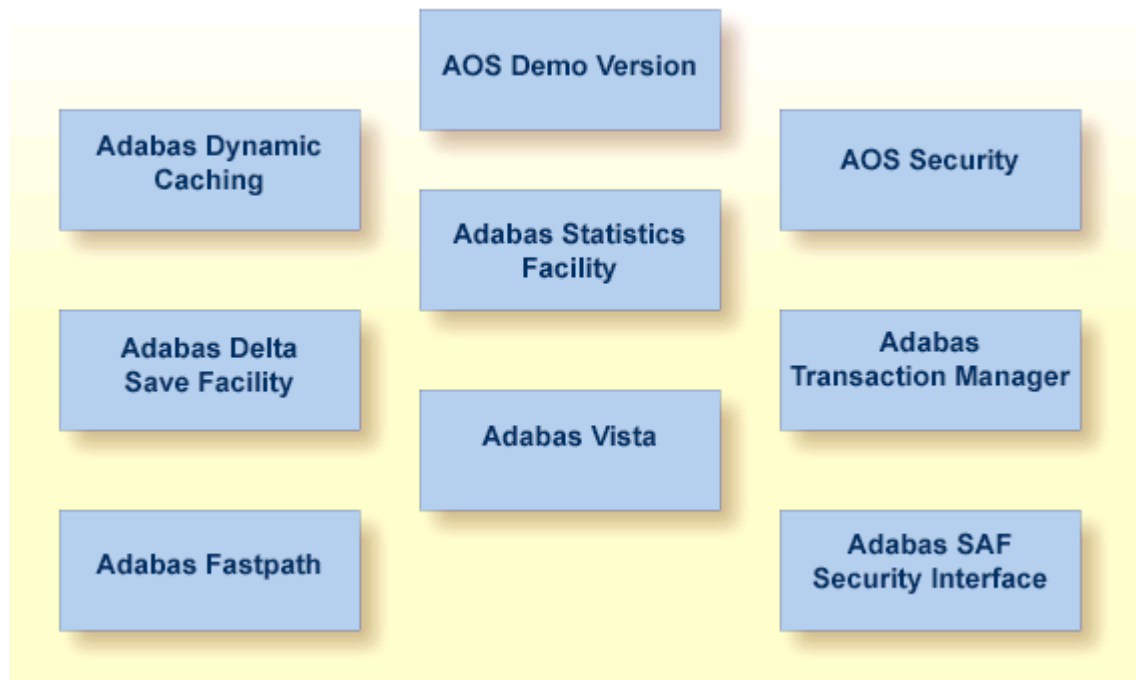
AOS Demo Version

This part of the DBA tasks documentation describes the operation and use of the AOS demo version. The use of the online services for the other Adabas products and facilities is described in the manuals for those products and facilities. The demo services available for Adabas Fastpath (AFPLOOK) and Adabas Vista (AVILOOK) are described in *AFPLOOK* and *AVILOOK*, elsewhere in this manual. AOS Security is described in the Adabas Security documentation.

This information is organized under the following headings:

- **Overview**
- **Main Menu Functions**
- **Session Monitoring**
- **List Checkpoints**
- **File Maintenance**
- **Database Maintenance**
- **System Operator Command Functions**
- **Database Report**

30 Overview

The AOS demo version includes the following functions that are comparable to Adabas operator commands and utilities and are used for Adabas database analysis and control:
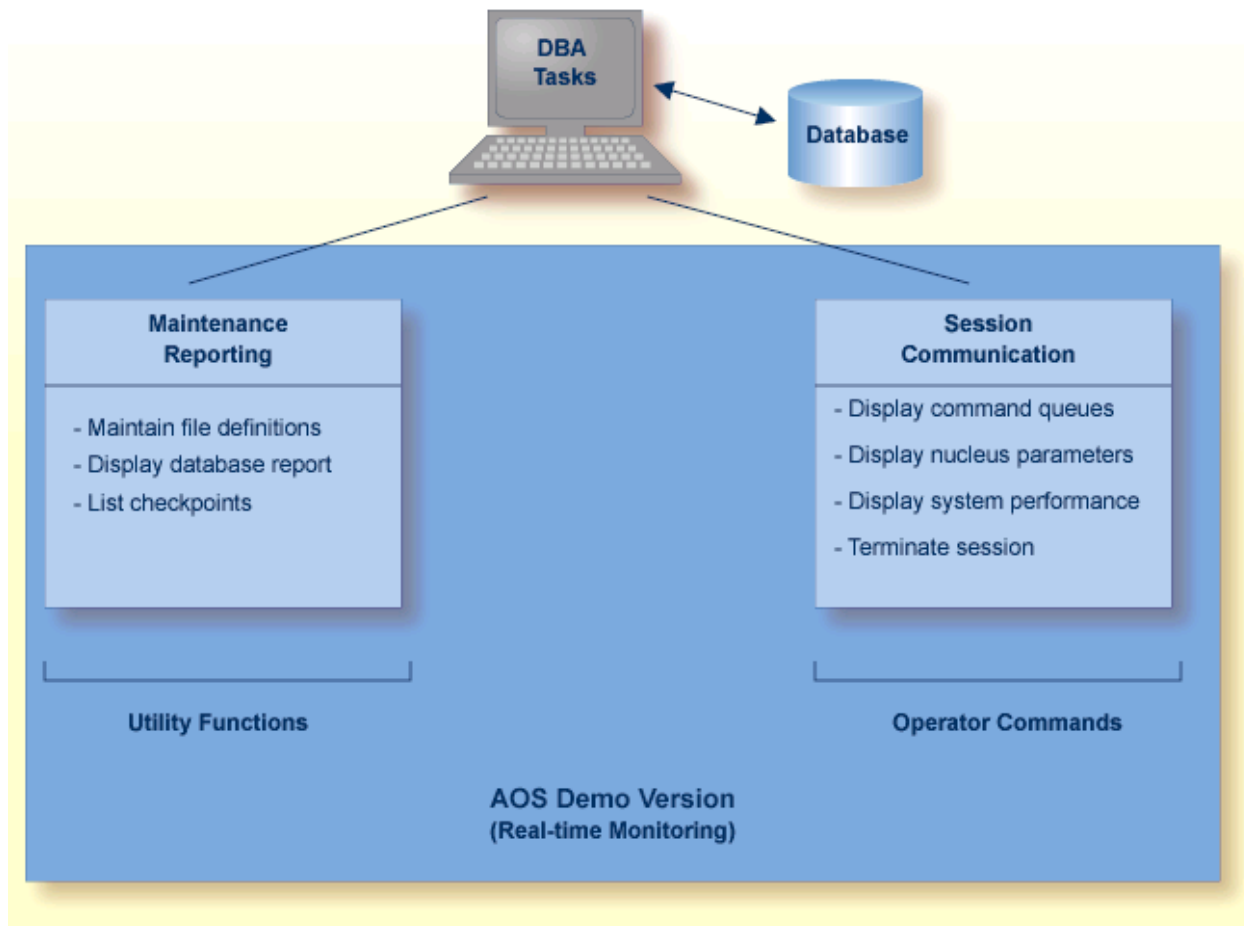
Overview of the AOS Demo Version

See the Adabas Installation documentation for information about installing the demo version of AOS.

The selectable unit AOS includes services that correspond to additional utility functions and operator commands. See the Adabas Online System documentation. for information.

This chapter covers the following topics:

What You Can Do with the AOS Demo Version

The DBA can use the AOS demo version to monitor aspects of an Adabas database while an Adabas session is active. Using menu options, the DBA can view resource and hold queue status; display space allocation; display file and database parameters; create new FDTs; and stop a current Adabas session.

For analyzing performance and monitoring database operation, the AOS demo version displays the system from the viewpoint of either a user or a particular system resource. For example, you can

- check hold queue status;
- view nucleus parameters;
- monitor command and file usage and system performance information;
- list file layout and extent status; and
- list file distribution of the database by VOLSER.

For controlling the overall Adabas session, the AOS demo version can be used to

- create new FDTs; and
- terminate an Adabas nucleus session (ADAEND).

31 Main Menu Functions

To enter the AOS demo version, log on to the Natural application SYSAOS and enter MENU at the NEXT prompt, if one appears.

> **Note:** If the full version AOS is installed on your system, enter MENU instead. See the Adabas Online System documentation for more information.

```
14:40:37            ***** A D A B A S  BASIC  SERVICES *****          2002-05-29
                          -  Main Menu  -                             PMAIN02


              Code  Basic Services              Code  Other Services
              ----  ----------------------      ----  ---------------------
               A    Session monitoring           1    Adabas Cache Facility
               C    Checkpoint maintenance       2    Delta Save Facility
               F    File maintenance             *     Trigger Maintenance
               M    Database maintenance         4    AOS Security
               O    Session opercoms             5    Transaction Manager
               R    Database report              6    Adabas Statistics
               *    Space calculation            7    Vista
               ?    Help                         8    Fastpath
               .    Exit                         9    SAF Security
              ----  ----------------------      ----  ---------------------

  Code ....... _
  Database ... 1955    (WIS1955)

Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help                Exit
```

The Main Menu displays the functions available with Adabas Online System. AOS functions that are not *not* available with the demo version are marked with an asterisk (*) on the screen.

From the Main Menu, you can access available "basic services" functions or one of the "other services" installed on your system. Other services installed at your site are highlighted.

The Main Menu indicates the main DBA tasks in the demo version:

Code	Function
A	*Session monitoring* allows you to display nucleus parameters, session and thread status information, ISNs in the hold queue, and maintenance levels for nucleus modules
C	*Checkpoint maintenance* allows you to list checkpoint information
F	*File maintenance* allows you to define a new FDT for a new file
M	*Database maintenance* is not active for the demo version
O	*Session opercoms* allows you to add or delete PIN modules used by the extended error recovery facilities; display currently loaded PIN routines and activate or deactivate them, display locked files, and terminate a session normally (ADAEND)

Code	Function
R	*Database report* allows you to view general database layout information, tables of database files, and detailed information for any file

Subsequent sections in this chapter describe the major functions of the AOS demo version and menu/screen structures in the order that they appear on the Main Menu.

This chapter covers the following topics:

Specifying the AOS Demo Version Database

The database on which the AOS demo version is installed becomes the default database for demo version functions. However, you can specify the database of any active Adabas nucleus session. Subsequent AOS demo version functions refer to that database until you specify another database or exit the AOS demo version.

Using Program Function (PF) Keys

Available PF keys and their functions are listed at the bottom of each screen. The following program function (PF) keys appear on all screens within the AOS demo version; other navigation keys appear on some screens:

PF1	Help
PF3	Exit to previous screen
PF12	Return to the Main Menu

Selecting a Menu Option

To select a function or option, enter the option code in the Code field.

Selecting a Main Menu function displays a menu of choices for that function.

Getting Help

From any AOS demo version menu, you can use PF1 to display a brief comment about the current menu. See *Using Program Function (PF) Keys* .

AOS Demo Version Messages

The AOS demo version issues a message confirming each completed function. If an error occurs, a message appears containing a reference number and describing the error.

Before analyzing an error, try reviewing the help information (PF1) for the last step you performed to see if any requirements were overlooked; then retry the operation.

Response code 22 is returned if the Adabas session is terminated and restarted while the AOS demo version is active. In this case, the application should be stopped and restarted.

32 Session Monitoring

Adabas session monitoring functions display major Adabas resources. These functions are most useful when analyzing system performance or seeking the cause of performance problems.

```
14:38:19              ***** A D A B A S  BASIC  SERVICES *****          2002-05-29
                         -  Session Monitoring  -                       PAC0002


 Code    Service                          Code    Service
 ----    -------------------------        ----    ----------------------------
  *      Display cluster members           *      Refresh nucleus statistics
  *      Maintain user profiles            *      Current session statistics
  D      Display parameters                *      Maintain TCP/IP URL
  *      Modify parameters                 U      Display session utilization
  Q      Display queues                    Z      Display maintenance levels
  ?      Help
  .      Exit
 ----    -------------------------        ----    ----------------------------



         Code ......... _
         Database ID .. 1955    (WIS1955)               NucID .. 1022



Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                                          Menu
```

You can use the Session Monitoring environment to monitor the Adabas nuclei in a multiprocessing (Parallel Services or Cluster Services) environment. When the DBID of a Parallel or Cluster Services database is entered on the Session Monitoring menu, subsequent screens include a field to specify the ID of the nucleus in the cluster that you want to monitor.

Each of the functions on the Session Monitoring menu is discussed in the following sections:

Code	Function
D	Display Adabas nucleus (ADARUN) parameters
Q	Display the contents of the hold queue
U	Display session status and thread usage
Z	Display Adabas nucleus modules: maintenance levels and ZAPs applied

This chapter covers the following topics:

Display ADARUN Parameters

You can view Adabas nucleus (ADARUN) parameters.

To view the parameters, select option D from the Session Monitoring menu.

Three screens are used for displaying parameters:

```
16:33:03          *****  A D A B A S   BASIC  SERVICES  *****        1999-01-28
  DBID 105                     -  Display Parameters  -               PACP002

  Modify parameters below, as required:
  -------------- Pools ---------------     ----------- Queues -----------------
  Sort Area         (LS) .. 19968      Command Queue        (NC) .. 100
  Int. User Buffer  (LU) .. 65535      Hold Queue           (NH) .. 9000
  Buffer Pool      (LBP) .. 127936     User Queue           (NU) .. 700
  Format Pool      (LFP) .. 12000      ---------- Time Windows ------------
  ISN List Table    (LI) .. 10000      Transaction Time     (TT) .. 300
  Seq. Cmd. Table   (LQ) .. 2500       Max Transaction Time (MXTT) .. 3600
  Work Pool        (LWP) .. 500000     Nonactivity ACC-User (TNAA) .. 300
  Attached Buffer  (NAB) .. 35         Nonactivity ET-User  (TNAE) .. 300
  Security Pool    (LCP) .. 10000      Nonactivity EXU-User (TNAX) .. 300
  UQ-DE Pool     (LDEUQP) .. 5000      Max Nonactivity Time(MXTNA) .. 3600
  Flush I/O Pool  (LFIOP) .. 125000    Time Limit Sx-Cmds (TLSCMD) .. 286
  Err. Recovery(SMGTBUFF) .. 0         Max Time for Sx-Cmds(MXTSX) .. 3600
                                       Command Time         (CT) .. 9000
                                       SYNS60 Interval    (INTNAS) .. 3600


                                                            Page 1 of 3
  PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
  Help               Exit                                 +        Menu
```

```
16:33:03          *****  A D A B A S   BASIC  SERVICES  *****        1999-01-28
  DBID 105                     -  Display Parameters  -               PACP002

  Modify parameters below, as required:
  --------- Miscellaneous -----------     -------- User Specific Limits ---------
  ReadOnly session (READONLY) .. NO    Hold Queue Limit  (NISNHQ) .. 2200
  UTI only session  (UTIONLY) .. NO    CIDs per User       (NQCID) .. 75
  OPEN required      (OPENRQ) .. NO    ISNs / TBI Element (NSISN) .. 51
  Ignore DIB Entry   (IGNDIB) .. NO    ------------ Buffer Pool -------------
  Local nucleus       (LOCAL) .. NO    Bufferflush Dur.  (TFLUSH) .. 1
  Number of Threads     (NT) .. 4      Parallel LFIOP I/O (FMXIO) .. 60
  Non DE Search      (NONDES) .. YES   Async. by Vol-Ser (ASYTVS) .. YES
  Log AOS/DBS Update (AOSLOG) .. NO
  Batch Support       (BATCH) .. NO
  Data Protection Area  (LP) .. 500
  Ignore Work Part 4 (IGNTPC) .. NO
```

```
   WORK-Part-4 Area     (LTPC) .. 0
   WORK-Part-2 Area     (LWKP2) .. 0
                                                        Page 2 of 3
```

```
Modify parameters below, as required:
   ---- Command Logging ----        --------- Protection Logging ----------
   Command Logging .. NO            PLOG required         (PLOGRQ) .. NO
   LOGCB ............ NO            DUAL PLOG Size      (DUALPLS) .. 0
   LOGFB ............ NO            DUAL PLOG Device    (DUALPLD) .. 0
   LOGRB ............ NO            ----------- Other Services ------------
   LOGSB ............ NO            Triggers and Procedures (SPT) .. NO
   LOGVB ............ NO            Delta Save Facility     (DSF) .. NO
   LOGIB ............ NO            Cache Facility        (CACHE) .. NO
   LOGIO ............ NO            Transaction Manager     (ATM) .. NO
   LOGUX ............ NO            TCP/IP Support        (TCPIP) .. NO
   LOGSIZE .......... 8904          Ext. Error Recovery   (SMGT) .. NO
   DUAL CLOG Size ... 0            2 Phase Commit Support  (TPC) .. NO
   DUAL CLOG Dev. ... 0            Review Support       (REVIEW) .. NO


                                                        Page 3 of 3
   PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
   Help               Exit
```

Display Hold Queue

Selecting *Queue displays* (option Q) from the Session Monitoring menu displays the following menu:

```
09:00:20         *****  A D A B A S   BASIC  SERVICES  *****       1997-01-29
                          -  Queue Displays   -                    PACQ002

                        Code  Service
                        ----  ---------------------------
                         *    Display User Queue Elements
                         *    Display Command Queue
                         H    Display Hold Queue
                         ?    Help
                         .    Exit
                        ----  ---------------------------
     Code .............. _
     Max No. Elements ... 100
     Last Activity ...... 0        (elapsed time in seconds)
     Selection Criteria
       ET-ID (User-ID) .. _____    User Type ... ___
       Job Name ......... _____
       Terminal ID ...... _____
     Database ID ........ 105        (RD-105)
   Command ==>
```

```
 PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
 Help                Exit      Clear UID                            Menu
```

Option H displays a list of the ISNs currently in hold status.

If the queue is currently empty, an appropriate message appears.

Display System Status and Thread Usage

Selecting *Resource utilization* (option U) from the Session Monitoring menu invokes the Resource Utilization menu:

```
11:44:10          *****  A D A B A S   BASIC  SERVICES  *****      1997-01-30
                        -  Resource Utilization   -                 PACU002

                     Code    Service
                     ----    -------------------------------
                      *      Command usage
                      *      File usage
                      *      High water marks (pools/queues)
                      *      Workpool (LWP) usage
                      *      PLOG status
                      S      System status
                      T      Thread usage
                      *      WORK status
                      ?      Help
                      .      Exit
                     --------  -------------------------------
           Code ......... _
           File Number .. 0
           Database ID .. 105    (RD-MPM105)


 Command ==>
 PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
 Help                Exit                                          Menu
```

Each option allows you to refresh (PF4) the displayed values, a convenience for long-term monitoring of Adabas system functions.

System Status

System status (option S) displays I/O counts for the ASSO, DATA, WORK, and PLOG datasets; remote and local call distribution; and other current session status information.

```
16:44:13         *****  A D A B A S   BASIC  SERVICES  *****        1999-01-28
  DBID 105                      -  System Status  -                   PACUS02

                Physical
             Reads          Writes              Call Distribution
          -------------------------    -------------------------------------
   ASSO              132          29   Remote Logical .............          0
   DATA                3           9   Remote Physical ............          0
   WORK                4          29   Local  Logical .............        145
   PLOG                            0   Local  Physical ............          0
   Logical Reads ........        194   Logical Reads (binary) .....   000000C2
   Buffer Efficiency ....        1.4   No. of HQEs active .........          0
                                       No. of UQEs in User Queue ..          2
   Format Translations ..          0   No. of CQEs waiting in CQ ..          0
   Format Overwrites ....          0
                                       Total intern. Autorestarts .          0
   Throw Backs for ISN ..          0   No. of PLOG switches .......          0
   Throw Backs for Space.          0   No. of Bufferflushes .......          8


  PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
  Help                Exit     Refresh                              Menu
```

Thread Usage

Thread usage (option T) displays a table of all defined Adabas threads, the status of each, the command type currently in process in each active thread, and the number of commands processed by each thread in the current session.

```
11:47:18         *****  A D A B A S   BASIC  SERVICES  *****        1997-01-30
  DBID 105                      -  Thread Status  -                   PACUT02

       Nr. I Thread Status          I Command Type     I Nr. CMDs  I
       ----------------------------------------------------------------
         1  I Active                I Simple Cmd.      I 18992      I
         2  I Not active            I                  I 109        I
         3  I Not active            I                  I 0          I
         4  I Not active            I                  I 0          I
         5  I Not active            I                  I 0          I
            I                       I                  I            I
```

Display Maintenance Levels

Selecting *Display maintenance levels* (option Z) from the Session Monitoring menu displays information about the Adabas nucleus modules:

```
13:43:09          *****  A D A B A S   BASIC  SERVICES  *****        1999-06-09
  DBID 105                 -  Display Maintenance Levels  -           DPACZ02

  Select Module Name:  _____
  ----------------------------------------------------------------------------
    ADARUN    RUNMVS   Date 1998-10-27, Version 7.1. 0, Zap Base A010000
              RUNIND   Date 1998-10-27, Version 7.1. 0, Zap Base AI10000
    ADATSP             Date 1998-10-30, SM Level 00, Zap Level 0000
                       Zaps 0034 0040 0043 0083 0084 0099
    ADATCP             Date 1998-10-30, SM Level 00, Zap Level 0000
                       Zaps 0136
    ADAMSG             Date 1998-10-30, SM Level 00, Zap Level 0000
    ADAIOR             Date 1998-10-29, SM Level 00, Zap Level 0000
    ADAIOS             Date 1998-10-29, SM Level 00, Zap Level 0000
                       Zaps 0001 0003 0004 0005 0007
    ADANCO             Date 1998-11-01, SM Level 00, Zap Level 0000
                       Zaps 0036


 Command ===>
 PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
 Help               Exit                 --        -        +       Menu
```

Maintenance levels for each module are displayed. Any ZAPs that are applied to the module are also listed.

The list of modules can be limited by entering a specific module name in the Select Module Name field at the top of the screen. A starting value may also be used. For example, specifying ADANC3 displays information for the ADANC3 module only. Specifying ADANC* lists all modules with names that begin with ADANC.

33 List Checkpoints

Selecting *Checkpoint maintenance* (option C) from the Main Menu invokes the Checkpoint Mainten-
ance menu:

```
14:41:59             ***** A D A B A S  BASIC  SERVICES *****            2002-05-29
                        -  Checkpoint Maintenance  -                     PCP0002


                     Code    Service
                     ----    ----------------------
                      C      List checkpoints
                      *      Delete checkpoints
                      ?      Help
                      .      Exit
                     ----    ----------------------

       Code ............. _
       Date(YYYY-MM-DD) . 0000-00-00
       Ext. CP-list ..... N
       Checkpoint Name .. ALL
       Database ID ...... 1955   (WIS1955)



Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                                          Menu
```

Option C lists checkpoints currently in the checkpoint file.

The result can be either a basic or an extended list, depending on the setting of the "External CP-
list" field, which can be used to override the CPEXLIST operating control parameter.

You can start the list of checkpoints on a particular day by entering the date in the Date field in
exactly the format shown.

You can specify the database for which the checkpoint list is to be written.

You can restrict the list to a particular type of checkpoint by changing the ALL designation in the Checkpoint Type field to one of the following:

SYNC nucleus initialization

SYNF user open EXF

SYNP utility without NUC

SYNS ADARES

SYNV volume ID change

SYNX utility

SYN1 ADASAV DB begin

SYN2 ADASAV DB begin

SYN4 ADASAV file begin

SYN5 ADASAV file begin

For more information about checkpoint types, see ADAREP in the Adabas Utilities documentation .

The following screen displays a normal checkpoint list:

```
15:12:22         *****  A D A B A S  Basic   Services  *****       1997-01-30
  DBID 105                       - List Checkpoints -                 PCPC002

   CP   CP    Date      Time       PLOG     Block      Vol/Ser   User Job Name
  Name Type                        Number   Number      Number   Type
  ---- ---- ---------- --------    ------   ----------  -------- ---- --------
  SYNC  01  1996-04-01 00:29:41                                       MPM105
  SYNS  60  1996-04-01 02:00:37                                       ADABAS
  SYNP  06  1996-04-01 02:43:55                                       PMS105SS
  SYNV  0A  1996-04-01 02:59:51                                       PMS105SS
  SYNV  0A  1996-04-01 02:59:51                                       PMS105SS
  SYNV  0A  1996-04-01 02:59:51                                       PMS105SS
  SYNV  0A  1996-04-01 02:59:51                                       PMS105SS
  SYNC  01  1996-04-01 03:12:30                                       MPM105
  SYNS  60  1996-04-01 04:34:43                                       ADABAS
```

This screen illustrates an extended checkpoint list providing additional information about each checkpoint:

```
12:58:49          *****  A D A B A S  Basic   Services  *****        1997-01-31
  DBID 105                       - List Checkpoints -                 PCPC002

  CP   CP    Date      Time      PLOG      Block       Vol/Ser   User Job Name
  Name Type                     Number    Number       Number    Type
  ---- ---- ---------- --------  ------  ----------   --------   ---- --------
  SYNC  01  1996-04-01 00:29:41                                        MPM235
       SESSION OPEN       IGNDIB = N , FORCE = N
  SYNS  60  1996-04-01 02:00:37                                        ADABAS
       STATISTIC RECORD
  SYNP  06  1996-04-01 02:43:55                                        PMS235SS
       SAVE DB
  SYNV  0A  1996-04-01 02:59:51                                        PMS235SS
       SAVE DB             VOL-SER = 502461    SESSION = 933
  SYNV  0A  1996-04-01 02:59:51                                        PMS235SS
       SAVE DB             VOL-SER =           SESSION = 933
  SYNV  0A  1996-04-01 02:59:51                                        PMS235SS
       SAVE DB             VOL-SER = 502215    SESSION = 933
```

34 File Maintenance

Selecting option F from the Main Menu invokes the File Maintenance menu:

```
14:42:25          ***** A D A B A S  BASIC  SERVICES *****        2002-05-29
                        - File Maintenance  -                     PFL0002
                  Code    Service
                  ----    -------------------------------
                  C       Define/modify FDT
                  *       Release descriptor
                  *       Delete existing file
                  *       Define new file
                  *       Modify file parameters
                  *       Reorder file online
                  *       Refresh file to empty status
                  *       Allocate/deallocate file space
                  *       Maintain expanded files
                  ?       Help
                  .       Exit
                  ----    -----------------------------
        Code ......... _
        File No ...... 0     Descriptor Name .. __
        Database ID .. 1955  (WIS1955)

Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                                           Menu
```

From the File Maintenance menu, option C displays the FDT/SDT Definition / Modification menu:

```
15:34:30          *****  A D A B A S   BASIC   SERVICES  *****        1998-07-30
                    -  FDT/SDT Definition / Modification  -           PFLC002

                    Code     Service
                    ----     ------------------
                     *       Add new field(s)
                     *       Change field length
                     D       Define new FDT
                     *       Online invert
                     *       Define/add SDT
                     ?       Help
                     .       Exit
                    ----     ------------------


           Code ......... _
           File No. ..... 50
           Field Name ... __
           Database ID .. 105    (RD-MPM105)

  Command ==>
  PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
  Help     Def. File Exit                                           Menu
```

From the FDT/SDT Definition / Modification menu, option D displays the Define FDT screen, which can be used to define a new FDT for a new file:

```
5:13:34           *****  A D A B A S   BASIC   SERVICES  *****        1997-02-12
  DBID 105                         -  Define FDT  -                   PFLCD02

  File Number ... 200                              New FDT ... Y
  Enter Field Description(s) :


        I Level I Name I Length I Format I Options           I
        I-----------------------------------------------------
        I   __  I  __  I   ___  I   _    I  __ __ __ __ __    I
        I   __  I  __  I   ___  I   _    I  __ __ __ __ __    I
        I   __  I  __  I   ___  I   _    I  __ __ __ __ __    I
        I   __  I  __  I   ___  I   _    I  __ __ __ __ __    I
        I   __  I  __  I   ___  I   _    I  __ __ __ __ __    I
        I   __  I  __  I   ___  I   _    I  __ __ __ __ __    I
        I   __  I  __  I   ___  I   _    I  __ __ __ __ __    I
        I   __  I  __  I   ___  I   _    I  __ __ __ __ __    I
        I   __  I  __  I   ___  I   _    I  __ __ __ __ __    I
        I   __  I  __  I   ___  I   _    I  __ __ __ __ __    I


  PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
  Help               Exit                                           Menu
```

FDTs for existing files cannot be redefined with this option.

This function corresponds to the Adabas utility function ADACMP COMPRESS.

35 Database Maintenance

Selecting option M from the Main Menu invokes the Database Maintenance menu:

```
14:42:42            ***** A D A B A S  BASIC  SERVICES *****        2002-05-29
                        -  Database Maintenance  -                  PDM0002

                   Code    Service
                   ----    ----------------------------
                    *      Add new dataset to ASSO/DATA
                    *      Increase/decrease ASSO/DATA
                    *      List/reset DIB block entries
                    *      Recover unused space
                    *      Uncouple two ADABAS files
                    ?      Help
                    .      Exit
                   ----    ----------------------------


       Code .......... _
       File No. ...... 0
       Coupled File .. 0
       Database ID ... 1955   (WIS1955)



Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                                           Menu
```

None of the Database Maintenance functions are active for the demo version.

36 System Operator Command Functions

Selecting *Session opercoms* (option O) from the Main Menu displays the following menu:

```
14:43:03              ***** A D A B A S  BASIC  SERVICES *****           2002-05-29
                        -  Session Opercoms  -                           PACIO02

 Code  Service                          Code  Service
 ----  -----------------------------    ----  -----------------------------
  *    Allocate/Deallocate CLOG/PLOG     S    Stop user(s)
  E    Extended Error Recovery           T    Termination Commands
  *    Force Dual CLOG or PLOG switch    *    Manage Online Utilities
  L    Lock or unlock files             *    User Table Maintenance
  *    Reset ONLINE-DUMP-Status
  ?    Help
  .    Exit
 ----  -----------------------------    ----  -----------------------------
       Code .......... _
       Userid(ETID) ... _____
       CLOG/PLOG Ind .. _       Global.. _
       Database ID .... 1955   (WIS1955)             NucID .. 1022



Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                                           Menu
```

The following functions are available to the AOS demo version:

Code	Function
E	Add or delete PIN modules used by the extended error recovery facilities; display, activate or deactivate current PIN routines
L	Display locked files
S	Stop a specific user, all users of a specific file or job, or all inactive users
T	Terminate a session normally (ADAEND)

This chapter covers the following topics:

Extended Error Recovery

Selecting option E (extended error recovery) from the Session Opercoms menu displays the Extended Error Recovery menu:

```
14:44:35          *****  A D A B A S   BASIC  SERVICES  *****        1999-05-12
                         -  Extended Error Recovery  -               DPACIE2

                     Code    Service
                     ----    ----------------------------------
                      *      Display message buffer
                      *      Display/modify environment
                      *      Display/modify Exit routines
                      M      Add/Delete PIN modules
                      P      Display/modify PIN routines
                      *      Refresh threshold and alert exits
                      *      SNAP a nucleus dump
                      ?      Help
                      .      Exit
                     ------  ----------------------------------


      Code ..........  _
      Start Address ..            End Address ...
      Database ID .... 823    (RD-CK-823)


   Command ==>
  PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
  Help               Exit                                            Menu
```

From this menu you can

■ add or delete PIN modules;

■ display, activate, or deactivate specific PIN routines.

Add / Delete PIN Modules

Selecting option M (add/delete PIN modules) from the Extended Error Recovery menu displays a list of currently available PIN modules:

```
13:42:45        *****  A D A B A S   BASIC  SERVICES  *****      1999-06-18
 DBID 823                 - Add/Delete PIN Modules -             PACIEM2

Mark entries with 'A' to Add or 'D' to Delete:


        M    Module    Description                      Message
        -    --------  --------------------------       -------
        _    ADAMXY    Standard Nucleus PIN Routines
             PINAAF    SAF Security
             PINAFP    Adabas Fastpath
             PINATM    Adabas Transaction Manager
             PINAVI    Adabas Vista
        _    PINRSP    Adabas Response Code Handler
        _    PINUES    Universal Encoding Support
```

To load a PIN module into memory, enter `A' in the M column next to the module name.

This command is successful only if the exit module exists in a library accessible to the Adabas nucleus.

To remove a PIN module from memory, enter a `D' in the M column next to the module name.

When deleting a PIN module from memory, all related PIN routines are also removed.

These functions are the same as the extended error recovery operator commands

```
SMGT,{ADDPIN | DELPIN}=module-name
```

Display/Modify PIN Routines

Selecting option P (display/modify PIN routines) from the Extended Error Recovery menu displays a list of PINs currently loaded in memory:

```
13:08:49        *****  A D A B A S   BASIC  SERVICES  *****      1999-06-16
 DBID 105                - List/Modify PIN Routines   -          PACIEP2

Mark entries with 'A' Activate, or 'D' Deactivate:        Total Pins: 012

 M  Condition        Error Location           Status  Uses  Module     Message
 -  --------  ----------------------------    -------  ----  --------   -----------
 _  000C1000  All Locations                   Active    0    ADAMXY
 _  000C2000  All Locations                   Active    0    ADAMXY
 _  000C3000  All Locations                   Not Act   0    ADAMXY
 _  000C4000  All Locations                   Active    0    ADAMXY
```

```
  _   000C5000  All Locations                    Active    0   ADAMXY
  _   000C6000  All Locations                    Active    0   ADAMXY
  _   000C7000  All Locations                    Not Act   0   ADAMXY
  _   000C8000  All Locations                    Active    0   ADAMXY
  _   000C9000  All Locations                    Active    0   ADAMXY
  _   000CB000  All Locations                    Active    0   ADAMXY
  _   000CF000  All Locations                    Active    0   ADAMXY
  _   00047000  All Locations                    Active    0   ADAMXY


PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help                Exit      Refr       --        -        +       Menu
```

For all PIN routines on the list, the screen indicatess the conditions that cause them to be executed, the current status, the number of times they have been used, and the module in which they are located.

To change the status of the PINs from this screen, enter in the M column next to the PIN number

A to activate a PIN

D to deactivate a PIN

After changes have been made, use PF4 to refresh the screen.

These functions are the same as the extended error recovery operator commands

```
SMGT,DISPLAY=PINS
  SMGT,{ACTPIN | DEACTPIN}=pin-number
```

Display Locked Files

Selecting option L from the Session Opercoms menu displays the following:

```
16:02:10        *****  A D A B A S   BASIC  SERVICES  *****     1999-01-28
                   -  Lock / Unlock Files   -                    PACIL02


              Code    Service
              ----    --------------------------------
              D       Display locked files
              *       Lock file for all users
              *       Advance lock file
              *       Lock file except for UTI/EXF users
              *       Unlock file from general lock
              *       Release an advance lock
              *       Unlock file from UTI/EXF lock
              ?       Help
              .       Exit
              ----    --------------------------------
```

```
     Code ......... _
     File Number .. 30
     UTI/EXF Ind .. U
     Database ID .. 105    (RD-105)


Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                                           Menu
```

Option D from this menu displays the Display Locked Files screen:

```
10:57:45          *****  A D A B A S   BASIC  SERVICES  *****        1998-07-31
 DBID 105                    -  Display Locked Files   -              PACID02

 M  Fnr.   Lock Status                 M  Fnr.   Lock Status
 -  -----  -----------------------     -  -----  ------------------------
    1       Locked for ALL users
    35      Locked except for UTI
    50      Locked except for EXU/EXF
    55      Locked for ALL users
    60      Locked for ALL users













PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                 --        -        +       Menu
```

Stop User(s)

Selecting option S (stop users) from the Session Opercoms menu displays the Stop Users menu:

```
18:26:02          *****  A D A B A S   BASIC  SERVICES  *****        1999-07-29
                       -  Stop Users   -                             PACIS02


                  Code  Service
                  ----  ----------------------
                   *    Stop users using file
                   *    Stop inactive users
                   *    Stop users by jobname
                   *    Stop a selected user
                   ?    Help
```

```
                        .    Exit
                      ----  ----------------------
     Code ............. _
     File Number ...... _____
     Last Activity .... _____  (elapsed time in seconds)
     Job Name ......... _____
     Purge UQE(s) ..... N
     Selected Userid ..
     Database ID ...... 105   (RD-105)

Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help     Disp UQ   Exit      Clear UID                           Menu
```

None of the Stop Users functions are active for the demo version.

Terminate a Session Normally (ADAEND)

Selecting option T from the Session Opercoms menu invokes the Session Termination menu from which you can terminate a session normally (ADAEND).

```
11:43:00           *****  A D A B A S   BASIC  SERVICES  *****      1997-01-30
                        -  Session Termination    -             PACT002

                   Code    Service
                   ----    ----------------------------------
                    A      Normal session termination (ADAEND)
                    *      Cancel session immediately (CANCEL)
                    *      Stop session              (HALT)
                    ?      Help
                    .      Exit
                 --------  ----------------------------------
       Code ........ _
       Database ID .. 105   (RD-MPM105)
        Current nr. of users in User Queue ... 9
        Nr. of users with open transactions .. 0




  Command ==>
  PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
  Help             Exit                                      Menu
```

You are prompted to confirm your termination request before the action is taken.

37 **Database Report**

Database Report functions, which correspond to selected functions of the Adabas ADAREP utility, provide both general and specific information in either table or report format.

```
14:43:26              ***** A D A B A S  BASIC  SERVICES *****           2002-05-29
                          -  Database Report  -                           PDR0002

                    Code    Service
                    ----    ------------------------------------
                    *       List files with crit. no. of extents
                    *       Display field description table (FDT)
                    F       Display file(s)
                    G       General database layout
                    *       List VOLSER distribution of database
                    *       Display ASSO/DATA block (RABN)
                    *       Display unused storage
                    ?       Help
                    .       Exit
                    ----    ------------------------------------
      Code ......... _
      File No ...... 0_____ Password ..
      Database ID .. 1955   (WIS1955)
      VOLSER ....... _____

Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                                            Menu
```

Options available to the AOS demo version allow you to view database-level general information and tables of database files, and file-specific information for any file:

Code	Function
F	Display file(s), either a list of all files in the specified database or detailed information about a specific file.
G	Display the general layout of the specified database.

This chapter covers the following topics:

Display Files

If no particular file is specified, option F lists all files in the specified database. If a file is specified, option F provides detailed layout information for the file. Physical device and file layout information is available only for a specific file.

Display a List of Files in the Specified Database

When no file number or "0" (zero) is specified in the File No field on the Database Report menu, a list of the files in the specified database is displayed:

```
09:24:38        *****  A D A B A S   BASIC  SERVICES  *****        1997-02-20
  DBID 105                      - Display Files -                    PDRF002

 Fnr  File Name         Loaded     Top-ISN    Max-ISN    Ext. Pad %  Ind.    %Used
                                                         NUAD  A  D ACISEXU  A  D
 ---- ---------------- ---------- ---------- ---------- ---- -- -- ------- -----
    1 EMPLOYEES        1993-06-15       1110       5511 1111  3  3 NNISNNN 68 88
    2 MISCELLANEOUS    1993-06-15       1779       5511 1111  3  3 NNISNNN 32 88
    4 AUTOMOBILES      1993-06-15       1000       5511 1111  3  3 NNISNNN 34 36
    5 PERSONNEL        1993-06-15       1000       5511 1111  3  3 NNISNNN 38 52
    6 FINANCE          1993-06-15       1000       5511 1111  3  3 NNISNNN 52 52
    7 GDMUSIC          1992-05-01       3292      16535 1111  3  3 NNNSNNN 81 95
    8 SAMPC-REV311DATA 1992-05-01      44679     100593 1111  3  3 NNNSNNN 79 99
    9 RD-NAT217-FUSER  1993-09-23     163272     175005 1111  3  3 NNISNNN 76 99
   10 RD-PRD314-FDIC   1992-05-01      60016      63387 1111  3  3 NNNSNNN 73 90
   11 REV320-DBFILE    1992-05-01       4442      11023 1111 10 10 NNNSNNN 42 82
   12 REV340-DBFILE    1993-02-15      52008      63387 1111 10 10 NNNSNNN  6 13
   13 SASRM-ZAP-TEST   1992-05-01         11       1377 1111  3  3 NNNSNNN 93  4
   14 SASRM-ZAPSYS     1992-05-01          5       1377 1111  3  3 NNNSNNN 28  4
   16 SAGDT-PRD-FDIC   1992-05-01      25649      30315 1111  3  3 NNISNNN 57 85



 PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
 Help     Repos     Exit                  --       -        +        Menu
```

The PF2 (Reposition) key displays a window in which you can enter a new starting value for the file list. When you enter a file number, the Display Files list begins with that file.

The Display Files screen provides the following information for each file:

- file number and file name;

- date the file was loaded into the database;

- highest ISN currently in use in the file and the highest ISN allowed in the file;

- number of logical extents currently assigned: by Associator (N ormal index; U pper index; A ddress converter) and D ata Storage. A maximum of five logical extents may be allocated to a file.

- block padding factor percentage defined for the Associator and for Data Storage;

■ indicators as follows:

A	ADAM option: A = ADAM ISN- or descriptor-selected file; N = non-ADAM file.
C	coupled (C) or non-coupled (N) file.
I	ISNs are reusable (I) or not (N).
S	Data Storage blocks are reusable (S) or not (N).
E	data files are ciphered/encrypted (E) or not (N).
X	files are expanded (X) or normal (N).
U	USERISN option: U = option is in effect for the file; N = option is not in effect.

■ percentage of allocated space currently used by the file in the Associator and in Data Storage.

Display Information for a Specific File

When a valid system file number is specified on the Database Report menu, the following Display File Layout information is displayed for that file:

```
18:27:37          *****  A D A B A S   BASIC  SERVICES  *****          1999-01-28
  DBID 105                    - Display File Layout -                    PDRF012
  ****************
  *  File 75    *   UES-FILE
  ****************
  Records loaded ..... 1107          Date loaded .......... 1999-01-26 12:18:17
  Top ISN ............ 1107
  Max ISN expected ... 1502          Max Compr Rec Lngth .. 4816
  Minimum ISN ........ 1            Asso/Data Padding .... 3%/3%
  Size of ISN ........ 3 Bytes      Highest Index Level .. 3
  Number of Updates .. 0
  ISN Reusage ........ NO            USERISN .............. NO
  Space Reusage ...... YES           MIXDSDEV ............. NO
  ADAM File .......... NO            PGMREFRESH ........... NO
  Ciphered File ...... NO            NOACEXTENSION ........ NO
  Coupled Files ...... NONE          Universal Encoding ... YES
  Blk per DS Extent .. 0
  Blk per UI Extent .. 0
  Blk per NI Extent .. 0             Length of Owner ID ... 0



 PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
 Help                Exit      Refresh                              Menu
```

The information for the file can be refreshed by pressing PF4 .

You can display additional information about space allocations by pressing ENTER .

The Display File Layout screen displays the following information for the file:

- the file number and name;

- the number of records currently contained in the file;

- ISN information: the highest ISN currently used in the file; the highest ISN planned for the file (see the ADALOD utility's MAXISN parameter); the lowest ISN that can be assigned to a record in the file (see the ADALOD utility's MINISN parameter); whether 3- or 4-byte ISNs are used for the file; and whether ISNs can be reused.

- the total number of updates since the file was last loaded;

- other file option settings: whether Data Storage space can be reused; whether the file was loaded with the ADAM option, the cipher option, the USERISN option; whether the file is physically coupled to another file; whether Data Storage extents can be on different device types; whether the file can be refreshed using the E1 command; whether the file permits the MAXISN setting to be increased.

- the number of blocks allowed per Data Storage, upper index, and normal index extent;

- the date and time the file was last loaded;

- the maximum compressed record length permitted for the file (see the ADALOD utility's MAXRECL parameter);

- the padding factor for the Associator and for Data Storage;

- the highest index level currently active for the file;

- the total number of blocks in the file that have been changed by updates since the file was last loaded;

- the length of the owner ID for multiclient files.

- whether universal encoding support (UES) is being used.

Pressing ENTER from the initial Display File Layout screen displays the following space allocation and usage information:

```
18:33:41          *****  A D A B A S   BASIC  SERVICES  *****          1999-01-28
  DBID 105                     - Display File Layout -                  PDRF022


  File 75


      IDeviceIListI  Space allocated  I     From      To     I  Unused         I
      I Type ITypeI  Blocks   / Cyls. I     RABN     RABN     I  Blocks / Cyls.I
  -----I------I----I------------------I -------------------I----------------I
      I      I    I                   I                     I                  I
  ASSO I 3380 I AC I       3        0 I     724 -      726 I       0      0 I
      I 3380 I UI I      15        0 I     747 -      761 I       0      0 I
      I 3380 I NI I      20        0 I     727 -      746 I       0      0 I
      I 3380 I NI I      56        0 I     762 -      817 I       2      0 I
      I      I    I                   I                     I                  I
  DATA I 3380 I DS I     116        0 I     216 -      331 I      29      0 I
```

```
 PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
 Help                Exit      Refresh                                Menu
```

Display General Database Layout

Option G displays general database information on the Display General DB-Layout screen:

```
18:43:07            *****  A D A B A S   BASIC   SERVICES  *****       1999-01-28
  DBID 105                    - Display General DB-Layout -             PDRG002

  Isolated
  Database Name ............ RD-105
  Database Number .......... 105
  Database Version ......... 7.1
  Database Load Date ....... 1998-10-21 14:40:47
  System Files ............. 19 , 0 , 0 , 0 , 0 , 0 , 0 , 0
  Maximum Number of Files .. 100
  Number of Files Loaded ... 5
  Highest File Loaded ...... 75
  Trigger File Number ...... 14
  Size of RABN ............. 4 Bytes
  Current Log Tape Number .. 5
  Delta Save Facility ...... Inactive
  Recovery Aid Facility .... Inactive
  Universal Encoding Sup. .. Yes




 PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
 Help                Exit                                             Menu
```

You can display additional information about UES codes, coupling, and space allocations by pressing ENTER.

The Display General DB Layout screen displays the following information for the file:

- the name and number of the database;

- the version level of the Adabas database software;

- the date and time the database was loaded;

- the numbers of Adabas system files allocated to the database;

- the maximum number of files permitted for the database; the total number of files currently loaded; and the highest file number currently in use;

■ whether 3- or 4-byte RABNs are being used for the file;

■ the number of the most recent data protection log tape for the database;

■ whether the Adabas Delta Save Facility and/or the Adabas Recovery Aid (ADARAI) are active or inactive for the database.

■ whether universal encoding support (UES) is being used.

When universal encoding support (UES) is being used, pressing ENTER from the initial Display File Layout screen lists the current code values:

```
18:51:22          *****  A D A B A S   BASIC  SERVICES  *****         1999-01-28
  DBID 105                    - Display General DB-Layout -              PDRG002


  Universal Encoding Support Enabled


  UES Encoding Keys:


  Alpha File Encoding ......... 37
  Wide File Encoding .......... 4095
  Alpha ASCII Encoding ........ 437
  Wide User Encoding .......... 4095
```

In any case, pressing ENTER from the initial Display File Layout screen displays the following space allocation and usage information:

```
18:52:01          *****  A D A B A S   BASIC  SERVICES  *****         1999-01-28
  DBID 105                    - Display General DB-Layout -              PDRG002


      IDeviceI    Total Number of  I     Extents  in Block   I DD-Names I
      I Type I    Blocks  /   Cyls. I      From         To    I          I
  ------I------I----------------------I --------------------------I----------I
      I      I                        I                          I          I
  ASSO I 3380 I     14231        50 I       1          14231   I DDASSOR1 I
      I      I                        I                          I          I
  DATA I 3380 I      6741        50 I       1           6741   I DDDATAR1 I
      I      I                        I                          I          I
  WORK I 3380 I      3592        30 I       1           3592   I DDWORKR1 I









  PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
  Help              Exit                                            Menu
```

A   Command Log Formats

This document provides the two command log formats that Adabas supports.

The first format (invoked by **CLOGLAYOUT=4**) has been in use since Adabas version 4.

The second format (invoked by **CLOGLAYOUT=5**) is supported only in versions 5.2 and above and is the default format. It includes additional information.

The tables in this appendix refer to the following command types:

Update	A1/A4, E1/E4, N1/N2
Simple	All non-update commands with a single search argument.
Complex	All non-update commands with more than one search argument.

The command type field may contain other information that is unrelated to commands. The flags should therefore be tested with a binary mask

```
B'xxxx xusc'
```

where "u" is the update flag bit, "s" is the simple flag bit, and "c" is the complex flag bit.

CLOGLAYOUT=4

Offset				Item Length/Format/Description
Hex.	Dec.	Bytes	Format	Explanation
0	0	2	binary	record length (inclusive)
2	2	2	binary	x '0000'
4	4	1	binary	record type (see note 1):
				x '00' basic record only
				x '01' includes control block
				x '02' includes format buffer
				x '04' includes record buffer
				x '08' includes search buffer
				x '10' includes value buffer
				x '20' includes ISN buffer
				x '40' includes I/O list
5	5	1	binary	user priority (from operating system)
6	6	1	binary	command type flags (simple, complex, update):
				x '01' simple command
				x '02' complex command

Offset			Item Length/Format/Description	
Hex.	Dec.	Bytes	Format	Explanation
				x '04' update command
7	7	1	binary	number of posted ECBs in command queue
8	8	1	binary	reserved
9	9	1	binary	thread number
A	10	2	binary	number of descriptors updated
C	12	8	alphanumeric	job name
14	20	4	binary	user ID (see note 2)
18	24	4	binary	wall clock time (STCK-format) at the point the command processing is completed by the nucleus
1C	28	4	binary	duration (in units of 16 microseconds)
20	32	8 (4 * 2)	binary	ASSO/DATA/WORK I/O counts for this command (except blocked ET command I/Os to WORK)
28	40	4	binary	Additions 2 field after command execution (can contain response subcodes)
2C	44	2	alphanumeric	command option 1 and 2 fields
2E	46	2	alphanumeric	command code
30	48	4	binary	command ID
34	52	2	binary	file number
36	54	2	binary	response code
38	56	4	binary	ISN
3C	60		end of basic record	
		64	alpha./binary	Rest of control block starting with ISN lower limit
		2	binary alphanumeric	format buffer length (inclusive) format buffer
		2	binary alphanumeric	record buffer length (inclusive) record buffer
		2	binary alphanumeric	search buffer length (inclusive) search buffer
		2	binary alphanumeric	value buffer length (inclusive) value buffer
		2	binary alphanumeric	ISN buffer length (inclusive) ISN buffer
		2	binary	I/O list length (inclusive)
		(n x 4)	binary	I/O list for this command; 4 or 5 bytes are used for each I/O operation, depending on the RABN length used in the database (RABNSIZE):
				If RABNSIZE=3, byte 1 contains:

Offset		Item Length/Format/Description		
Hex.	Dec.	Bytes	Format	Explanation
				01 = Associator read
				02 = Associator write
				03 = Data read
				04 = Data write
				05 = Work read
				06 = Work write
				Bytes 2-4 contain the 3-byte RABN that is read or updated.
				If RABNSIZE = 4, byte 1 contains:
				81 = Associator read
				82 = Associator write
				83 = Data read
				84 = Data write
				85 = Work read
				86 = Work write
				Bytes 2-5 contain the 4-byte RABN that is read or updated.

📄 **Notes:**

1. The value in the record type field (offset +4) determines which parts of the record mentioned above are present.

2. For CICS, Shadow, and Westi TP monitor users, Adabas version 5 does not provide a unique ID value in this field. Users of these TP monitors must revise programs requiring a unique ID value to obtain it from the command queue element's (CQE's) CQEUID field. User exit 4 provides the CQE address at location 12(R1).

CLOGLAYOUT=5

Explanation of Log Record Types

Command log layout 5 comprises two log record types:

- the basic log record type (x'0001') is produced for all commands processed on noncluster nuclei and for those that arrive from a remote nucleus and run under internal command queue elements (ICQEs).

- the asynchronous request log record type (x'0002') is created on a nucleus that sends a command to another nucleus. This record type is used in Adabas nucleus cluster environments only.

  - Fields with Alternate Interpretations
  - Remote Nucleus ID Field
  - Command-Executing Nucleus ID Field
  - Zap AY742008: Enhanced I/O-List

Fields with Alternate Interpretations

The following fields have alternate interpretations for each log record type:

- for the basic log record type (x'0001'):

Offset			Item Length/Format/Description	
Hex.	Dec.	Bytes	Format	Explanation
46	70	2	binary	Associator I/O operation count
48	72	2	binary	Data I/O operation count
4A	74	2	binary	Work I/O operation count

- for the asynchronous request log record type (x'0002'):

Offset			Item Length/Format/Description	
Hex.	Dec.	Bytes	Format	Explanation
46	70	2	binary	number of internucleus destinations
48	72	2	binary	internucleus messaging return code
4A	74	2	binary	internucleus messaging reason code

Remote Nucleus ID Field

A field has been added at offset 4C for both log record types:

Offset		Item Length/Format/Description		
Hex.	Dec.	Bytes	Format	Explanation
4C	76	2	binary	the ID of the remote nucleus

- For the basic log record type (x'0001') where the command originates in another nucleus, this field contains the ID of the nucleus that sent the command. It remains zero for locally-executed commands and noncluster nuclei.

- For the asynchronous request log record type (x'0002'), this field contains the ID of the nucleus to which the command is sent. It is zero for implicit broadcasts (GLOBAL and SYSTEM) and contains the ID of the first destination for explicit broadcasts (list of destinations). The CLOG record contains the Adabas control block and buffers related to the first (or only) destination, and the job name and communications ID of the initiator of the request.

Command-Executing Nucleus ID Field

A field has been added at offset 58 for both log record types:

Offset		Item Length/Format/Description		
Hex.	Dec.	Bytes	Format	Explanation
58	88	2	binary	the ID of the nucleus executing the command
60	90	2	-	reserved (x'0000 0000')

For both log record types, this field provides the ID of the nucleus executing the command. In a cluster environment, the nucleus ID is either the SMPID or the PLXID; in a noncluster environment, the nucleus ID is zero (0).

Zap AY742008: Enhanced I/O-List

If you apply optional zap AY742008, the enhanced I/O-List is displayed. In this case, an I/O-List element contains not only the control byte and the RABN, but also the name of the volume where the RABN resides.

With zap AY742008, the control byte is changed. The X'40' bit is switched on. Six-byte volume names are attached after the I/O list.

For example, if RABNSIZE = 3 and byte 1 contains:

```
41 = Associator read
42 = Associator write
...
46 = Work write
```

In this case, bytes 2-4 contain the three-byte RABN that is read or updated and bytes 5-10 contain the volume name where the RABN resides.

In another example, if RABNSIZE = 4 and byte 1 contains:

```
C1 = Associator read
C2 = Associator write
...
C6 = Work write
```

Bytes 2-5 contain the four-byte RABN that is read or updated and bytes 6-11 contain the volume name where the RABN resides.

Adabas Basic Log Record Type (x'0001')

Offset		Item Length/Format/Description		
Hex.	Dec.	Bytes	Format	Explanation
0	0	2	binary	record length (inclusive)
2	2	2	-	reserved (x '0000')
4	4	2	binary	og record type:
				x '0001': Adabas basic log record
6	6	2	-	reserved (x '0000')
8	8	8	binary	time in store clock (STCK) format when request was logged
10	16	1	binary	buffer types:
				x '00' basic record only
				x'01' includes Adabas control block
				x '02' includes format buffer
				x '04' includes record buffer
				x '08' includes search buffer
				x '10' includes value buffer
				x '20' includes ISN buffer
				x '40' includes I/O list
				x '80' includes user exit B data
11	17	1	binary	dispatching priority (from operating system)

Offset			Item Length/Format/Description		
Hex.	Dec.	Bytes	Format	Explanation	
12	18	1	binary	command type (simple, complex, update - see note):	
				x '01' simple command	
				x '02' complex command	
				x '04' update command	
13	19	1	binary	number of posted ECBs in command queue	
14	20	1	binary	thread number	
15	21	1	binary	user buffer (UB) flag (see DSECT "UB")	
16	22	2	binary	number of descriptors updated	
18	24	8	alphanumeric	job name	
20	32	28	binary	communication ID	
3C	60	4	binary	elapsed time for request in command queue (in units of 16 microseconds)	
40	64	4	binary	unique command sequence number	
44	68	2	binary	database ID	
46	70	2	binary	Associator I/O operation count	
48	72	2	binary	Data I/O operation count	
4A	74	2	binary	Work I/O operation count	
4C	76	2	binary	ID of the remote nucleus that sent the command	
4E	78	1	binary	architecture byte of caller (reserved)	
4F	79	5	-	reserved (x'0000 0000 00')	
54	84	4	binary	Adabas Review	
58	88	2	binary	ID of the nucleus executing the command	
60	90	2	-	reserved (x'0000 0000')	
(Beginning of Adabas control block log area:)					
5C	92	2	-	reserved (x'0000')	
5E	94	2	alphanumeric	command code	
60	96	4	alphanumeric	command ID	
64	100	2	binary	file number	
66	102	2	binary	response code	
68	104	4	binary	ISN	
6C	108	4	binary	ISN lower limit	
70	112	4	binary	ISN quantity	
74	116	2	binary	format buffer length	
76	118	2	binary	record buffer length	
78	120	2	binary	search buffer length	

Offset		Item Length/Format/Description		
Hex.	Dec.	Bytes	Format	Explanation
7A	122	2	binary	value buffer length
7C	124	2	binary	ISN buffer length
7E	126	1	alphanumeric	command option 1
7F	127	1	alphanumeric	command option 2
80	128	8	alphanumeric	additions 1 field
88	136	4	alphanumeric	additions 2 field (returned lengths and response subcodes)
8C	140	8	alphanumeric	additions 3 field (password)
94	148	8	alphanumeric	additions 4 field (cipher code)
9C	156	8	alphanumeric	additions 5 field (global format ID/OPEN: timeout values)
A4	164	4	binary	command time in thread (in units of 16 microseconds)
A8	168	4	binary	user area
(Beginning of optional log record area:)				
AC	172	2	binary alphanumeric	format buffer length (inclusive)
AE	174	2	binary alphanumeric	record buffer length (inclusive)
B0	176	2	binary alphanumeric	search buffer length (inclusive)
B2	178	2	binary alphanumeric	value buffer length (inclusive)
B4	180	2	binary alphanumeric	ISN buffer length (inclusive)
B6...	182..	2	binary	I/O list length (inclusive)
		(n x 4)	binary	I/O list for this command; 4 or 5 bytes are used for each I/O operation, depending on the RABN length used in the database (RABNSIZE):
				If RABNSIZE = 3, byte 1 contains
				01 = Associator read
				02 = Associator write
				03 = Data read
				04 = Data write
				05 = Work read
				06 = Work write
				Bytes 2-4 contain the 3-byte RABN that is read or updated.
				If RABNSIZE = 4, byte 1 contains
				81 = Associator read

Offset				Item Length/Format/Description
Hex.	Dec.	Bytes	Format	Explanation
				82 = Associator write
				83 = Data read
				84 = Data write
				85 = Work read
				86 = Work write
				Bytes 2-5 contain the 4-byte RABN that is read or updated.

Asynchronous Request Log Record Type (x'0002')

Offset				Item Length/Format/Description
Hex.	Dec.	Bytes	Format	Explanation
0	0	2	binary	record length (inclusive)
2	2	2	-	reserved (x '0000')
4	4	2	binary	log record type:
				x '0002': asynchronous request log record
6	6	2	-	reserved (x '0000')
8	8	8	binary	time in store clock (STCK) format when request was logged
10	16	1	binary	buffer types:
				x '00' basic record only
				x'01' includes Adabas control block
				x '02' includes format buffer
				x '04' includes record buffer
				x '08' includes search buffer
				x '10' includes value buffer
				x '20' includes ISN buffer
				x '40' includes I/O list
				x '80' includes user exit B data
11	17	1	binary	dispatching priority (from operating system)
12	18	1	binary	command type (simple, complex, update - see note):
				x '01' simple command
				x '02' complex command
				x '04' update command
13	19	1	binary	number of posted ECBs in command queue
14	20	1	binary	thread number
15	21	1	binary	user buffer (UB) flag (see DSECT "UB")

Offset		Item Length/Format/Description		
Hex.	Dec.	Bytes	Format	Explanation
16	22	2	binary	number of descriptors updated
18	24	8	alphanumeric	job name
20	32	28	binary	communication ID
3C	60	4	binary	elapsed time for request in command queue (in units of 16 microseconds)
40	64	4	binary	unique command sequence number
44	68	2	binary	database ID
46	70	2	binary	number of internucleus destinations
48	72	2	binary	internucleus messaging return code
4A	74	2	binary	internucleus messaging reason code
4C	76	2	binary	ID of the remote nucleus to which the command is sent
4E	78	1	binary	architecture byte of caller (reserved)
4F	79	5	-	reserved (x'0000 0000 00')
54	84	4	binary	Adabas Review
58	88	2	binary	ID of the nucleus executing the command
60	90	2	-	reserved (x'0000 0000')
(Beginning of Adabas control block log area:)				
5C	92	2	-	reserved (x'0000')
5E	94	2	alphanumeric	command code
60	96	4	alphanumeric	command ID
64	100	2	binary	file number
66	102	2	binary	response code
68	104	4	binary	ISN
6C	108	4	binary	ISN lower limit
70	112	4	binary	ISN quantity
74	116	2	binary	format buffer length
76	118	2	binary	record buffer length
78	120	2	binary	search buffer length
7A	122	2	binary	value buffer length
7C	124	2	binary	ISN buffer length
7E	126	1	alphanumeric	command option 1
7F	127	1	alphanumeric	command option 2
80	128	8	alphanumeric	additions 1 field
88	136	4	alphanumeric	additions 2 field (returned lengths and response subcodes)
8C	140	8	alphanumeric	additions 3 field (password)

Offset		Item Length/Format/Description		
Hex.	Dec.	Bytes	Format	Explanation
94	148	8	alphanumeric	additions 4 field (cipher code)
9C	156	8	alphanumeric	additions 5 field (global format ID/OPEN: timeout values)
A4	164	4	binary	command time in thread (in units of 16 microseconds)
A8	168	4	binary	user area
(Beginning of optional log record area:)				
AC	172	2	binary alphanumeric	format buffer length (inclusive)
AE	174	2	binary alphanumeric	record buffer length (inclusive)
B0	176	2	binary alphanumeric	search buffer length (inclusive)
B2	178	2	binary alphanumeric	value buffer length (inclusive)
B4	180	2	binary alphanumeric	ISN buffer length (inclusive)
B6...	182..	2	binary	I/O list length (inclusive)
		(n x 4)	binary	I/O list for this command; 4 or 5 bytes are used for each I/O operation, depending on the RABN length used in the database (RABNSIZE):
				If RABNSIZE = 3, byte 1 contains:
				01 = Associator read
				02 = Associator write
				03 = Data read
				04 = Data write
				05 = Work read
				06 = Work write
				Bytes 2-4 contain the 3-byte RABN that is read or updated.
				If RABNSIZE = 4, byte 1 contains:
				81 = Associator read
				82 = Associator write
				83 = Data read
				84 = Data write
				85 = Work read
				86 = Work write
				Bytes 2-5 contain the 4-byte RABN that is read or updated.

B   Supplied UES Encodings

The tables in this document list the encodings available with universal encoding support (UES) when executing Adabas utilities or issuing Adabas commands. Encodings represent single-byte character sets (Latin-1 or not) or double-/multiple-character sets. If encodings

- have a character set in common with other encodings so that conversion between them is accomplished without loss of data, they are "interoperable".
- are not interoperable, they are "coexistent" with other encodings.

Columns used in the following tables are described as follows:

Key	ADAECS key number in decimal and hexadecimal		
CS	character set as identified by IBM		
CP	code page usually identical to the "key". It is *not* identical when ■ CP does not fit into the ECS key number range 1-4095; or ■ the encoding is constructed from two or more code pages.		
Size	F	encoding uses all allocated graphical character space	
	M	maximum for given ESID	
	S	subset	
ESID	encoding scheme identifier with the following entries:		
	1100	EBCDIC fixed 1	
	1301	EBCDIC mixed DBCS modal	
	2100	IBM PC data fixed 1	
	2200	IBM PC data DBCS only	
	2300	IBM PC data mixed DBCS nonmodal	
	4100	ISO-8 fixed 1	
	4105	ISO-8 fixed 1 graphic characters in the range x'80'- x`9F' reserved for control codes	
Fill	the hexadecimal value representing the fill character used in the encoding.		
Sub	the hexadecimal value representing the substitution character used in the encoding.		

Encodings are categorized as follows:

Interoperable Encodings

Single-Byte Character Sets (Latin-1)

Key		CS	CP	F/M/S Size	ESID	Fill	Sub	Description
Dec	Hex							
37	025	697	37	F 190	1100	40	3F	CECP: USA, Canada (ESA*), Netherlands, Portugal, Brazil, Australia, New Zealand
273	111	697	273	F 190	1100	40	3F	CECP: Austria, Germany, de_deu
277	115	697	277	F 190	1100	40	3F	CECP: Denmark, Norway
278	116	697	278	F 190	1100	40	3F	CECP: Finland, Sweden
280	118	697	280	F 190	1100	40	3F	CECP: Italy, it_ita
284	11C	697	284	F 190	1100	40	3F	CECP: Spain, Latin America (Spanish), es_esp
285	11D	697	285	F 190	1100	40	3F	CECP: United Kingdom, en_gbr
297	129	697	297	F 190	1100	40	3F	CECP: France, fr_fra
500	1F4	697	500	F 190	1100	40	3F	CECP: Belgium, Canada (AS/400*), Switzerland, international Latin-1
819	333	697	819	F 190	4100	20	1A	ISO 8859-1: Latin alphabet, Latin-1
871	367	697	871	F 190	1100	40	3F	CECP: Iceland, is_ISL
923	39B	**	923	F 190	4100	20	1A	ISO 8859-15 with euro sign
924	39C	**	924	F 190	1100	40	3F	IBM EBCDIC with same char. set as 923.
1140	474	**	1140	F 190	1100	40	3F	IBM EBCDIC 37 with euro sign
1141	475	**	1141	F 190	1100	40	3F	IBM EBCDIC 273 with euro sign
1142	476	**	1142	F 190	1100	40	3F	IBM EBCDIC 277 with euro sign
1143	477	**	1143	F 190	1100	40	3F	IBM EBCDIC 278 with euro sign
1144	478	**	1144	F 190	1100	40	3F	IBM EBCDIC 280 with euro sign
1145	479	**	1145	F 190	1100	40	3F	IBM EBCDIC 284 with euro sign
1146	47A	**	1146	F 190	1100	40	3F	IBM EBCDIC 285 with euro sign
1147	47B	**	1147	F 190	1100	40	3F	IBM EBCDIC 297 with euro sign
1148	47C	**	1148	F 190	1100	40	3F	IBM EBCDIC 285 with euro sign
1149	47D	**	1149	F 190	1100	40	3F	IBM EBCDIC 871 with euro sign
*3946	F6A	697	850	S 190	2100	20	7F	PC data-190: Latin alphabet, Latin-1 (CCSID=4946)

* This code page is not yet available but may be at a later time.

** This information is not yet available.

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

Key Dec	Hex	CS	CP	F/M/S Size	ESID	Fill	Sub	Description
420	1A4	235	420	M 181	1100	40	3F	IBM, Arabic (all presentation shapes)
424	1A8	941	424	M 152	1100	40	3F	IBM, Hebrew, iw_isr
813	32D	925	813	M 183	4100	20	1A	ISO 8859-7: Greek/Latin, el_GRC
870	366	959	870	F 190	1100	40	3F	IBM, Latin-2
875	36B	925	875	M 184	1100	40	3F	IBM, Greek, el_GRC
912	390	959	912	F 190	4100	20	1A	ISO 8859-2: Latin-2
914	392		914	F 190	4100	20	1A	ISO 8859-4
915	393	1150	915	F 190	4100	20	1A	ISO 8859-5: Cyrillic, 8-bit
916	394	941	916	M 152	4100	20	1A	ISO 8859-8: Hebrew, iw_ISR
918	396	1160	918	F 190	1100	40	3F	IBM, Urdu
919	397		919	F190	4100	20	1A	ISO 8859-10
920	398	1152	920	F 190	4100	20	1A	ISO 8859-9 Latin-5 (ECMA-128, Turkey TS-5881), tr_TUR
921	399	1305	921	F 190	4100	20	1A	IBM, Baltic, 8-bit
922	39A	1307	922	F 190	4100	20	1A	IBM, Estonia, 8-bit
1006	3EE	1160	1006	F 190	4100	20	1A	IBM, Urdu
1025	401	1150	1025	F 190	1100	40	3F	IBM, Cyrillic multilingual
1026	402	1152	1026	F 190	1100	40	3F	IBM, Turkey Latin-5, tr_TUR
1112	458	1307	1112	F 190	1100	40	3F	IBM, Baltic, multilingual (EBCDIC)
1122	462	1307	1122	F 190	1100	40	3F	IBM, Estonia (EBCDIC)

Double- and Multiple-Byte Character Sets

Key Dec	Hex	CS	CP	F/M/S Size	ESID	Fill	Sub	Description
290	122	1172	290	M 162	1100	40	3F	Japanese Katakana host extended SBCS, ja_JPN
836	344	1174	836	M 99	1100	40	3F	Simplified Chinese host extended SBCS (EBCDIC), zh_CHN
1027	403	1172	1027	M 162	1100	40	3F	IBM, Japanese Latin host extended SBCS, ja_JPN
1041	411	1172	1041	S 162	2100	20	1A	IBM, Japanese PC data extended SBCS, ja_JPN
1043	413	1175	1043	M 97	2100	20	1A	IBM, traditional Chinese PC data extended SBCS, zh_TWN
1115	45B	1174	1115	M 99	2100	20	1A	IBM, simplified Chinese PC data single-byte (IBM GB) including 5 SAA SB characters, zh_CHN

Key		CS	CP	F/M/S Size	ESID	Fill	Sub	Description
Dec	Hex							
1380	564	937	1380	M 9355	2200	A1A1	FEFE	IBM, simplified Chinese DBCS PC (IBM GB) including 1880 UDC and 31 IBM-selected, zh_CHN
1381	565	1174937	11151380	M 9454	2300	20 A1A1	1A FEFE	IBM, simplified Chinese PC data mixed (IBM GB) including 1880 UDC, 31 IBM-selected and 5 SAA SB characters, zh_CHN
3026	BD2	1172370	290 300	S 9306	1301	40 4040	3F FEFE	IBM CCSID 5026, Japanese Katakana-Kanji host mixed including 1880 UDC, extended SBCS, ja_JPN
3035	BDB	1172 370	1027 300	S 9306	1301	40 4040	3F FEFE	IBM CCSID 5035, Japanese Latin-Kanji host mixed including 1880 UDC, extended SBCS, ja_JPN
3396	D44	370	300	S 9144	1200	4040	FEFE	IBM, CCSID 4396, Japanese host double-byte including 1880 UDC, ja_JPN
3709	E7D	1175	3709	S 97	1100	40	3F	IBM CCSID 28709: traditional Chinese host extended SBCS (EBCDIC), zh_TWN
4091	FFB			F 63456		0020	001A	UTF-8: with user/compatibility area; without surrogates

Coexistent Encodings

Single-Byte Character Sets

Key		CS	CP	F/M/S Size	ESID	Fill	Sub	Description
Dec	Hex							
3	3	103	367	F 94	5100	20	1A	US-ASCII
6	6			F 190	4100	20	1A	ISO 8859-3, Latin-3, Afrikaans, Catalan, Dutch, English, German, Italian, Maltese, Spanish, Turkish
9	9			M 145	4100	20	1A	ISO 8859-6, Arabic
437	1B5	697	437	F 222	2100	20	1A	Microsoft MSDOS US, en_USA
720				F 223	2100	20	1F	Microsoft Windows OEM Codepage 720, Arabic
737				F 223	2100	20	1F	Microsoft Windows OEM Codepage 737, Greek
775				F 223	2100	20	1F	Microsoft Windows OEM Codepage 775, Baltic
850	350	1106	950	F 223	2100	20	1A	Microsoft multilingual code page 850, Latin-1
852				F 223	2100	20	1F	Microsoft Windows OEM Codepage 852, cp852_DOSLating2
857				F 223	2100	20	1F	Microsoft Windows OEM Codepage 857, Turkish

Key		CS	CP	F/M/S Size	ESID	Fill	Sub	Description
Dec	Hex							
858	350	1106	950	F 222	2100	20	1A	Microsoft multilingual code page 850, Latin-1, euro-ready
862				F 223	2100	20	1F	Microsoft Windows OEM Codepage 862, Hebrew
866				F 223	2100	20	1F	Microsoft Windows OEM Codepage 866, Cyrillic
874				F 223	2100	20	1F	Microsoft Windows OEM Codepage 874, Thai
1047		697		F 190	1100	40	3F	IBM Latin-1 Open Systems EBCDIC
1250	4E2	1400	1250	M 217	4105	20	1A	Microsoft, Windows ANSI, Latin-2, euro-ready
1251	4E3	1401	1251	M 220	4105	20	1A	Microsoft, Windows ANSI, Cyrillic, euro-ready
1252	4E4	1402	1252	M 215	4105	20	1A	Microsoft, Windows ANSI, Latin-1, euro-ready
1253	4E5	1403	1253	M 206	4105	20	1A	Microsoft, Windows ANSI, Greek, euro-ready
1254	4E6	1404	1254	M 215	4105	20	1A	Microsoft, Windows ANSI, Turkey, euro-ready
1255	4E7	1405	1255	M 194	4105	20	1A	Microsoft, Windows ANSI, Hebrew, euro-ready
1256	4E8	1406	1256	M 214	4105	20	1A	Microsoft, Windows ANSI, Arabic, euro-ready
1257	4E9	1407	1257	M 203	4105	20	1A	Microsoft, Windows ANSI, Baltic rim, euro-ready
2084		948	878	F 222	4105	20	1F	KO18-R
2087		813	775	S 222	2100	20	1F	MS-DOS Baltic PC-Data
2258		1408	1258	S 212	4105	20	1F	Microsoft, Windows-1258, Vietnam
3585	E01	646		S 95	1100	40	3F	BS2000 DF.03 International CCSN:EDF03IRV
3586	E02	265		S 95	1100	40	3F	BS2000 DF.03 CCSN:EDF03DR Austria, Germany (EBCDIC) - with Dollar
3587	E03			S 95	1100	40	3F	BS2000 DF.03 CCSN´:EDF03DRI Austria, Germany (EBCDIC) - with International Currency Sign
3588	E04	697		F 190	1100	40	3F	BS2000 DF.04-DR V LATIN 1, CCSN:EDF04DR Austria, Germany (EBCDIC)
3589	E05	697		F 190	1100	40	3F	BS2000 DF.04-1 LATIN 1, CCSN:EDF041IRV International (EBDIC)
4092	FFC			3	-	20	A1	Software AG, used for ECS internally
4093	FFD				4100	20	1A	Software AG, old TS default, ASCII
4094	FFE	-	-		1100	40	3F	Software AG, old TS default, EBCDIC
4095	FFF			-	7200	0020	001A	Unicode

Double- and Multiple-Byte Character Sets

Key		CS	CP	F/M/S Size	ESID	Fill	Sub	Description
Dec	Hex							
18	12	103 1061 1121 1062	367 952 896 953	S 13102	4403	20 8140	1A	IBM CCSID 1350, EUC-JP, ja_JPN composed of US-ASCII, JIS-X-0208, HW Katakana, and JIS-X-0212-90
36	24				5404	20	1A	ISO-2022-JP, US-ASCII, JIS-Roman, JIS-0208-1983, JIS_C_6226-1978
300	12C	1001	300	M 11634	1200	4040	FEFE	IBM Japanese Latin host double-byte including 4370 UDC, ja_JPN
301	12D	370	301	M 9144	2200	8140	FCFC	IBM Japanese PC double-byte including 1880 UDC, SJIS, ja_JPN
835	343	935	835	M 20263	1200	4040	FEFE	IBM traditional Chinese host double-byte including 6204 UDC (EBCDIC) zh_TWN
837	345	937	837	M 9355	1200	4040	FEFE	Simplified Chinese host double-byte including 1880 UDC (EBCDIC), zh_CHN
927	39F	935	927	M 20263	2200	8140	FCFC	IBM traditional Chinese PC data double-byte including 6204 UDC, zh_TWN
932	3A4	1122370	897 301	M 9301	2300	20 8140	1A	Microsoft, JIS Roman, JIS-X-208, half-width Katakana, ja_JPN
935	3A7	1174 937	836 837	M 9454	1301	40 4040	3F FEFE	Simplified Chinese host mixed including 1880 UDC, extended SBCS (EBCDIC), zh_CHN
936	3A8	1185 937	1185 937	M 9449	2300	20 1A1A	1A	Microsoft, GB Roman, GB 2312-80, zh_CHN
937	3A9	1175 935	903 937	S 20360	1301	40 4040	3F FEFE	IBM traditional Chinese host mixed including 6204 UDC, extended SBCS (EBCDIC), zh_TWN
942	3AE	1172 370	1041 301	M 9306	2300	20 8140	1A FCFC	IBM, Japanese PC data mixed including 1880 UDC, extended SBCS, ja_JPN
948	3B4	1175 935	948	M 20360	2300	20 8140	1A FCFC	IBM, traditional Chinese PC data mixed including 6204 UDC, extended SBCS, zh_TWN
949	3B5	1278 1050	949	M 10197	2300	20 A1A1	1A AFFE	IBM, Korean IBM KS code - PC data mixed including 1880 UDC, ko_KOR
950	3B6	103 935	950	M 20357	2300	20 A140	1A	Microsoft, Big Five, zh_TWN
951	3B7	1050	951	M 10103	2200	A1A1	AFFE	IBM, Korean IBM KS code - PC data double-byte including 1880 UDC, ko_KOR
3001		10001	3001			20 8140	1A	MAC Japanese, JIS-Roman, JIS-X-208, HW Katakana, ja_JPN

Key		CS	CP	F/M/S Size	ESID	Fill	Sub	Description
Dec	Hex							
4037		1175 935		S 20360	1301	40 4040	3F FEFE	Same as EDD 937 Chinese EBCDIC, with code point X'5F' mapped to Unicode U+005E and X'B0' to U+00AC

38 AFPLOOK

The Adabas Fastpath command analysis sampler (AFPLOOK) can be used to determine where the best results may be expected from Adabas Fastpath by reporting on the command constructs that qualify for Adabas Fastpath.

The sampler can be controlled and viewed online using SYSAFP and, in addition, prints the results at normal database termination.

> **Note:** For those Adabas customers who do not have an Adabas Fastpath license, the AFPLOOK sampler along with a demo version of SYSAFP is distributed as part of the Adabas release tape. In this form, by default, AFPLOOK will sample a full database session.

Enabling AFPLOOK

AFPLOOK is enabled using the following ADARUN command:

```
ADARUN FASTPATH=YES
```

Operational Defaults

AFPLOOK is set up with certain operational defaults that control the amount of memory used during command analysis by restricting

- the maximum number of files sampled; and
- the number of concurrent users.

If any parameter is exceeded, AFPLOOK tries to ignore the excess while still reporting maximum information. In this way, AFPLOOK audits a general sampling of the database command workload to determine Fastpath optimization parameters. The operational defaults can be modified for site requirements as described in the section **APLOOK Parameters.**

AFPLOOK Parameters

This section describes the AFPLOOK parameters, which are used to define the boundaries of the sample and limit the amount of memory required.

- Maximum Files
- Command/Descriptors per File
- Maximum Concurrent Users
- Maximum CIDs per User

- Maximum Commands Processed
- Job Name
- Selected Files

Maximum Files

The maximum number of files to be sampled.

Once the maximum number of files is put in the analysis table, no additional files are sampled; however, additional files show in a command count so that it can be determined whether or not this parameter should be increased for subsequent executions.

Default: 64

Command/Descriptors per File

The maximum number of command/descriptor entries per file.

In conjunction with the `Maximum Files` parameter, this parameter restricts the amount of memory used. If the maximum entries is reached for a file, the last entry is converted into a general accumulator. Note that only one entry is required for the Adabas command types `L1`, `L2`, `S8`, and `S9`.

Default: 32

Maximum Concurrent Users

The maximum size for the table of concurrent users.

If all the user areas are being used at one time, a new request is satisfied by releasing the 'oldest usage' user area. The number of times this reusage occurs is noted and printed in the summary. When a sample contains a high percentage of reusage, this parameter should be adjusted.

Default: 100

Maximum CIDs per User

The maximum concurrent Adabas Command IDs (CIDs) sampled for each user processed.

In conjunction with the Maximum Concurrent Users parameter, this parameter restricts the amount of memory used. Commands for Command IDs that exceed this maximum are ignored and reported as rejected. When a sample contains a high percentage of rejections, this parameter should be adjusted.

Default: 10

Maximum Commands Processed

The maximum number of commands to be sampled.

Default: No limit

Job Name

Used to restrict sampling to a particular job name. One or more asterisks (*) can be used in the job name as a wild card character so that the sample can select all jobs that match the name ignoring the character positions occupied by an asterisk (*).

Default: None

Selected Files

Used to restrict sampling to specific files.

This option may be useful where the maximum files overflowed, or file activity is known and detailed analysis is required.

Default: All files

AFPLOOK Report

This section describes the types of information available on the AFPLOOK report:

- File Summary
- Potential Optimization Summary
- Sample Command Analysis
- Report Parameters

File Summary

This section of the report provides a summary of the file commands.

```
--- -- ----  ----------          --  ----------  ---------
FNR CC DESC  DIRECT ACC          RC  SEQUENTIAL  SEQUENCES
--- -- ----  ----------          --  ----------  ---------
 20 L1  --            1
    L2  --                                    4          4
    L3  CC            1
    L9  AA            1
    L9  BB            2
```

```
    L9  CC              2                           1         1
    S1  AA                                          3         1
    TOTALS              7                           8             21(18%)


                                      EXCLUDED COMMANDS:          2
                                      ALREADY PREFETCHED:         3


   (UPDATES 2,INSERTS 1,DELETES 1)  (MAX.RBL DA 0,SEQ 32)
-------------------------------------------------------------------
```

Column	Explanation
FNR	Adabas file number.
CC	Adabas command code.
DESC	Primary descriptor.
DIRECT ACC	Maximum number of direct access commands that can be optimized.
RC	Maximum number of RC commands that can be optimized.
SEQUENTIAL	Maximum number of sequence commands that can be optimized.
SEQUENCES	Number of sequences that caused the number of sequential commands. The sequence factor for optimization may be calculated from these two numbers.

The rightmost number shows the total sampled commands for the file together with the percentage relative to all the sampled file commands. On a large report, this number can be used to determine quickly which files should be considered for optimization.

Commands that have been ignored for the file are also listed along with the reason for exclusion.

The final line shows the update commands as well as the maximum record buffer lengths found for direct access or sequential commands that can be optimized.

Potential Optimization Summary

This section of the report summarizes the total commands sampled for all files and expresses this as a percentage of all commands seen. Excluded commands are similarly reported.

```
----------------------- POTENTIAL OPTIMIZATION SUMMARY ------------------
        SAMPLED COMMANDS                     MAXIMUM OPTIMIZATION
SAMPLED FILE COMMANDS      116 ( 77%) <------ SEQUENTIAL:        55 (47%)
                                             DIRECT ACCESS:     32 (27%)
                                                     RCS:        4 ( 3%)
EXCLUDED COMMANDS           33 ( 22%)
TOTALS                     149 (100%)                          91 (61%)
-----------------------------------------------------------------------------
```

The maximum optimization numbers are an estimation of potential optimization. The sequential commands, direct access, and RC totals are expressed as a percentage of the total sampled file commands. The total is expressed as a percentage of all commands.

These numbers indicate the estimated *potential* optimization using Adabas Fastpath. The actual optimization will depend on various factors unique to each user site. Contact Software AG for assistance when interpreting samples.

Sample Command Analysis

This section of the report provides command analysis information.

```
--------------------------------------------------------------------------
                     COMMAND ANALYSIS
REJECTED COMMANDS
        MAX. USERS EXCEEDED:           0
        MAX. CIDS  EXCEEDED:           0
        MAX. FILES EXCEEDED:           0            0 (   0%)
EXCLUDED COMMANDS
        BAD COMMANDS:                  4
        NON-FILE COMMANDS:             7
        NON-FILE RCS:                  2
        EXCLUDED FILE COMMANDS:        8
        UPDATE COMMANDS:               4
        ALREADY PREFETCHED:            8           33 (  22%)
SAMPLED FILE COMMANDS                             116 (  77%)
ALL COMMANDS SEEN                                 149 (100%)
--------------------------------------------------------------------------
```

The numbers shown:

- illustrate the type of commands processed, and

- put the previous section into perspective.

Rejected commands are categorized by users, CIDs, and files exceeded. If the total percentage is high, estimates reported elsewhere may not give an accurate assessment.

Excluded commands are split into the following categories:

Category	Explanation
Bad commands	Unexpected Adabas response codes.
Non-file commands	Commands that cannot be attributed to a file; for example, `OP`,`CL`,`ET`,`C1`,`RE`. Plus file commands `HI`,`LF`,`RI`.
Non-file RCs	All `RC` commands plus any `RC` for which the CID is not stored by AFPLOOK.
Excluded file commands	`L4`,`L5`,`L6`,`S4`,`S5`.
Update commands	`A1`,`A4`,`E1`,`E4`,`N1`.
Already prefetched	Any command that could qualify for sequential optimization that has prefetch or multifetch already set.

Report Parameters

This section of the report

■ shows the important parameters used to produce the report; and

■ gives an indication of the parameters needed.

```
-----------------------------------------------------------------
     PARAMETERS USED
     MAX. FILES:    64  FILES NEEDED:       5
      ..MAX. DE:    32  OVERFLOWS:          0
     MAX. USERS:   100  HIGH USERS:        15
      ..MAX.CID:    10  HIGH CIDS:          4
     * REUSED USER AREA OCCURRENCES:       0
     MAX.RECORDS: NO LIMIT
-----------------------------------------------------------------
```

39 AVILOOK

The Adabas Vista command analysis sampler (AVILOOK) can be used to determine those files which may benefit from Adabas Vista's Partitioning option by reporting on the command constructs used to access the file.

The sampler is controlled and viewed online using SYSAVI and, in addition, prints the results at normal database termination.

> **Note:** For those Adabas customers who do not have an Adabas Vista license, the AVILOOK sampler along with a demo version of SYSAVI is distributed as part of the Adabas release tape.

Enabling AVILOOK

AVILOOK is enabled using the following ADARUN command:

```
ADARUN VISTA=YES
```

SYSAVI – Selecting AVILOOK

▶ **To select AVILOOK**

■   Selecting service 4 from the SYSAVI main menu displays the AVILOOK menu options:

```
+-----------------------------------------------------------------+
|                     Code    Service                             |
|                     ----    ---------------                     |
|                       1     File Maintenance                    |
|                       .     Exit                                |
|                     ----    ---------------                     |
|           Code..: _                                             |
|                                                                 |
|           Database ID..: _____                                  |
|                                                                 |
|            Nucleus ID..: _____                                  |
|                                                                 |
|   System Coordinator Node.: _____                               |
|                                                                 |
+-----------------------------------------------------------------+
```

SYSAVI – Using AVILOOK

This section describes how to use AVILOOK.

- AVILOOK File Maintenance
- AVILOOK File Statistics
- AVILOOK Add Files

AVILOOK File Maintenance

The File Maintenance screen lists the files that are already defined to AVILOOK for a specified database number. The database name is also displayed.

▶ **To display the File Maintenance screen from the AVILOOK menu**

1   Select service option 1.

2   Specify the Adabas database number for the database for which you wish to run AVILOOK.

   The database must be running with the `ADARUN VISTA=YES` parameter.

3   (Optional) For a cluster database, specify a Nucleus ID.

   If the database number you specify is a cluster database, you have the option to specify the Nucleus ID of the cluster nucleus you wish to monitor.

```
+----------------------------------------------------+
|  13:31:07     Select Cluster Members  2002-11-09  |
|                                         V1NUCSM1   |
|  DBID:   231                                        |
|                                                    |
|   C    Nuc    C    Nuc    C    Nuc    C    Nuc     |
|   _     1     _     2                              |
|                                                    |
|                                                    |
|                                                    |
|  Mark to select                                    |
|  Command ==>                                       |
|           PF3 Exit     PF4 Refr                    |
|                                                    |
+----------------------------------------------------+
```

You may select the appropriate nucleus from this list.

If you choose not to specify a Nucleus ID or you specify a value of 0, you are required to specify the Node ID of the local Adabas System Coordinator. A window is displayed listing the nuclei that are currently active in the cluster.

If the job within which you are using SYSAVI is defined to a System Coordinator group, the Node ID of the local Adabas System Coordinator is automatically set up.

```
DBID: 231    (TEST-V7-DB)

C    File    Command Limit        Commands        Started          Status
_      12               0               0                          Paused
_       2               0            5768   1999-09-24 09:09:20    Active




Mark with (A)ctivate,(P)ause,(R)eset,(S)tatistics,(X)Delete

 Command ==>
```

Press PF4 to refresh the command count.

▶ **To select a file entry**

■   Enter one of the following options in column C next to the file to be selected:

a	activate; start gathering statistics
p	pause; stop gathering statistics
r	reset statistics to zero
s	display the current statistics for the file
x	delete the file from the file list

AVILOOK File Statistics

▶ **To display the File Statistics screen from the AVILOOK menu**

■ Enter option "s" in column C next to the file entry:

```
DBID: 231   (TEST-V7-DB)
File: 2                                        Started: 1999-09-24 09:09:20
                                               Paused :


     CC     Desc    Command Count         CC     Desc    Command Count
     L3      AA            2836
     S1      AB            1324
     L3      BC              24
     L9      S1              26


 Other Commands not listed above: 1558


 Command ==>
```

This screen shows statistics on the command constructs used to access the file (for example, S1, L3 and L9 commands). The statistics are displayed in descending order by command code (CC) and Adabas two-character field name (Desc).

In this example, there are 2836 accesses to file 2 using an L3 command with the Adabas field AA as the primary sequence field. Such a file, where the predominant access is by a single key, may benefit from being partitioned using the Adabas field AA as the Adabas Vista partitioning field.

AVILOOK

AVILOOK Add Files

▶ **To add a new AVILOOK file**

1   Press PF10 from the AVILOOK File Maintenance screen.

```
+------------------------------------------------------------------------+
|  08:11:07                    AVILOOK Add File                 1999-09-24 |
|                                                                V14100M2  |
|                                                                          |
|          File _____     Status (A/P)  P       (Active/Paused)           |
|                         Command Limit 0       (0=No Limit)              |
|                                                                          |
|                    PF3 Exit    PF5 Update                               |
|                                                                          |
+------------------------------------------------------------------------+
```

2   Enter the file number.

3   Indicate whether you want AVILOOK to start gathering statistics immediately (Active status) or you want to define the file now and activate it at a later time (Pause status).

4   (Optional) Predefine the maximum number of commands the active file can process before it automatically reverts to pause status.

5   Press PF5 to add the file.

📄   **Note:** Files defined (and activated) are only sampled for the duration of the current database session.

Index

A

Adabas Cluster Services
   monitor a session online, 206
Adabas Online System
   demo version
      error messages for, 204
      invoking, 202
      obtaining help information, 204
      overview, 198, 202
      range of options, 199
      specifying a database for, 203
      using program function (PF) keys, 203
Adabas Parallel Services
   monitor a session online, 206
ADAMXY module, 170
ADARAI utility
   concept of generation, 97
ADARUN
   display parameters, 207
ADASMXIT, 179
Associator
   blocks reserved by Adabas, 124
   device allocation, 139
   space calculation
      automatic, 124
Autobackout
   function of routine, 91

C

Checkpoints
   list current, 213
   types of, 214
Command log
   device allocation, 139
Condition description block (CDB), 180

D

Data compression
   options
      default, 76
      fixed-storage (FI), 76
      null-value suppression, 76
Data Storage
   device allocation, 139
   space calculation

automatic, 124
Database
   design, 55
   online maintenance menu, 221
   online report, 232
DBA
   database planning, 42
   documenting
      the database, 22
   maintaining user ID and password information, 27
   training responsibilities, 32
   using a data dictionary, 24
Descriptor
   prefixed with multiclient owner IDs, 104

E

ETID
   conversion to ownerid by ADALOD, 106

F

Field definition table
   define for a new file
      using AOS demo version, 218
Files
   display locked, 228
   online maintenance menu, 217

I

Improved data recording capability
   use recommendation, 141

M

MAXISN
   ADALOD parameter, 147
Multiclient files
   converting from single client, 100
   description of, 100
   owner ID
      relationship to user ID, ETID, 100
Multiprocessing
   monitoring sessions, 206

P

PIN modules

ADAMXY, 170
installing, 171
list of, 171
PIN routine
processing, 170
user exit, 179
PIN routines
defined, 169
PINAUTOR
activating, 172
installing, 172
user exit with, 181
PINOPRSP
activating, 172
installing, 172
Predict
Adabas data dictionary
overview, 24
Protection log
device allocation, 139

R

RABNs
device space required by, 145
Recovery log
defining, 96
device allocation, 139

S

Session
display status, 210
monitor online, 206
Sort
device allocation, 139
Space
estimating for database as a whole, 125
estimating for each file, 125
System
display status, 210

T

Temp
device allocation, 139
Threads
display usage statistics, 210

U

User
stop operation for
using Basic Services, 228
User exits
ADASMXIT, 179
handling errors in, 168
PIN routine, 179

W

Work area
device allocation, 139